HTML EXECUTABLE 2025

USER GUIDE

Make secure ebooks & apps

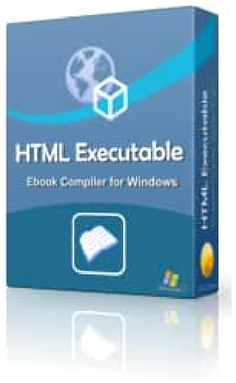
G.D G. Software

Table of Contents

1. Welcome to HTML Executable

This is the online documentation for HTML Executable.

Description



HTML Executable™ is a versatile HTML to EXE software compiler. It lets you turn HTML, PDF, and DOCX documents, or simple and complex websites, into stand-alone Windows applications for the desktop called **digital publications or ebooks**. It is compatible with all recent Windows versions, including Windows 11.

Why compile your website or webpage into an executable file? Perhaps you want to showcase your work without any risk of someone reading your HTML code or stealing images. Maybe you need to display a website to someone who doesn't have a browser, or you want to ensure people see the website exactly as intended. You might also want to develop desktop applications, create digital information products, archive your websites, build and sell impressive ebooks, or even construct modern help systems as a powerful alternative to older formats like Microsoft HTML Help (CHM).
If any of these scenarios apply to you, HTML Executable offers a comprehensive solution.

Use HTML Executable to create fully customized ebooks, presentations, digital publications, paywalled content, Rich Internet Applications, interactive demos, offline documentation, and compact archives - all designed for efficient, secure, and user-friendly distribution or backup of your websites.

Full customization of your ebooks

Nearly every aspect of the ebook application can be customized according to your needs: apply skins to the user interface; choose GUI items; display splash screens or custom messages; integrate a stand-alone PDF viewer; add a table of contents, a search engine, and a favorite manager; restrict access to pages; create trial ebooks and define user rights; use software activation to protect your digital property; use scripting to extend the behavior of your ebook; include an automated web update when you release a new version; protect your ebook with dongles; make portable versions that run on USB drives; change the icon and version information of your .exe files and digitally sign them; and much more!

You can compile almost any website containing HTML, JavaScript, PDF files, Word DOCX, images, video, audio, XML, and more. HTML Executable is the **perfect package for anyone who needs to publish secure materials electronically, make digital information products, and earn money from ebooks.**

If you want to run PHP files in compiled ebooks, please consider our other product: ExeOutput for PHP.

How to Use This Documentation

There are three main ways to navigate this documentation:

Browse through the different topics using the **Contents tree** on the left side to discover HTML Executable's features.

Press F1 in the software, it will open for you the topic related to the current page.

Use the **documentation's search engine** to find information on a specific issue.

Getting started with HTML Executable

Start a new project

Frequently Asked Questions

If you have **questions**, feedback, or bugs to report, you can visit our forum or contact us by **e-mail**: info@htmlexeNOSPAM.com.

For further information about HTML Executable, visit our website at https://www.htmlexe.com.

2. Overview and Getting Started

2.1. Getting Started

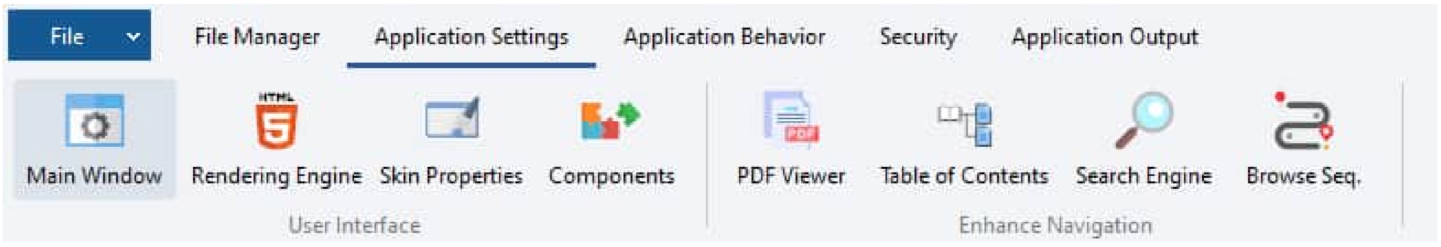
What HTML Executable Basically Does

HTML Executable is a software that compiles websites into standalone Windows applications (.exe files). These applications embed the website into a custom web browser, enabling end users to browse the website as if they were using their regular web browser, but without requiring an internet connection or a web server. Importantly, source files such as HTML pages, JavaScript files, images, etc., are **securely embedded within the application and are not accessible to the end user**, preventing them from being copied. In addition to this, HTML Executable offers **content protection and licensing features** to further secure the packaged website, so that you can even earn money from your websites, ebooks, and much more.

Navigating the Interface

In HTML Executable, a "**publication**" or "**ebook**" refers to a **compiled website**. We will use these terms to designate the application running your website, created by you with HTML Executable.

HTML Executable uses ribbons: all related features are grouped into pages belonging to one of the five following ribbons or tabs: File Manager, Application Settings, Application Behavior, Security, and Application Output.



- Under "**File Manager**", you can manage files that will be compiled into your ebook or publication. It works like Windows Explorer, with a tree displaying the folders on the left and the files inside the selected folder on the right. You can add, remove, or edit files and configure their properties.
- The "**Application Settings**", "**Application Behavior**", and "**Security**" ribbons are the heart of HTML Executable: they contain all the features that let you customize the behavior and appearance of your ebook.
- Finally, with the "**Application Output**" ribbon, you can access options related to how HTML Executable will output the publication .exe file.

To navigate between the ribbons, click on the desired tab at the top of the main window.

Standard and additional commands, such as Load/Save Project and Build options, are available by clicking the **File** Menu button: File

Shortcuts are available in the main toolbar: New Project, Load Project, Save Project, Compile Publication, and Run Publication.

Starting Your First Project

Each time you open HTML Executable, it displays the welcome page where you can choose to create a new project or load an existing one.

At any time, if you have questions, feel free to consult the FAQ section, this documentation, the www.htmlexe.com website, or **post your questions to the** user forum, where you can also share your feedback.

Always begin a new publication by creating a project to save your settings, so you don't have to configure them again each time.

You can also start a new project by importing an HTML Help file (.chm) using the free CHM To Exe add-on: select "File|Convert HTML Help..." in HTML Executable.

Or, if you have an ebook in EPUB format, use our free add-on EPub to APP to import it into HTML Executable.

See a video tutorial online about making ebooks from PDF files with HTML Executable.

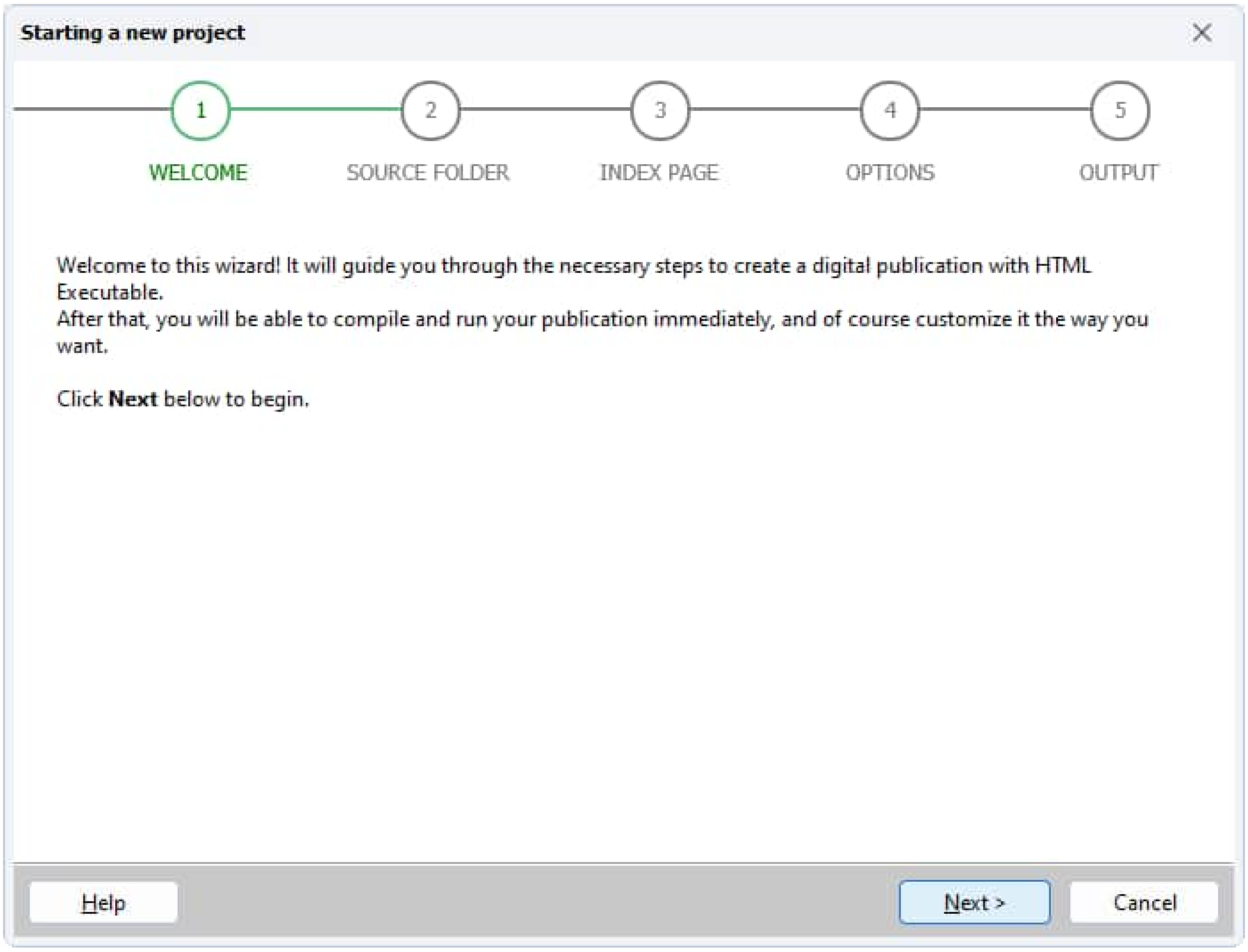
Creating a project

Frequently Asked Questions

2.2. Starting a new project

Starting a new ebook or publication project with HTML Executable involves several quick and easy steps.

To create a project file, select **New Project** in the Application menu after clicking File (or click **Start a new publication project** in the toolbar) and this window will appear:



Welcome to this wizard! It will guide you through the necessary steps to create a digital publication with HTML Executable.
After that, you will be able to compile and run your publication immediately, and of course customize it the way you want.

Click **Next** below to begin.

This window is called the **New Publication wizard**: it will assist you in the creation of your new project and it gives you advice about the different steps you need to follow.

HTML Executable stores all of its settings (the project) in a file called **project file** (file extension: .hepx). You can use the Open/Save buttons in the main window's bar to open or save project files at any time. But before that, you need to create a new project.

There are four remaining steps that we will describe below.

1) Selecting the source folder of your website

The **source folder** is the folder that contains all files that compose your website. HTML Executable will consider it as the **root of your website**: it will become the root of your publication / ebook, and all the files inside this folder and its subdirectories (if any) will be automatically added to the publication when the project is being created. You can, of course, add/remove files later if you wish.

The **directory structure** of your website is stored in the publication: the **source folder** is set as the root of the publication/website. File paths will be stored relative to this source folder so links between HTML pages will work exactly as if you were navigating through a website on a server (see virtual paths). You can consider a compiled publication as a server.

2) Choose the index page of your publication

The **index page** is the first page that is displayed to your end users when the publication is run. It is considered the Home page (it will be displayed when end users click the Home button).

HTML Executable lists all files available in the source folder and **you have to select which HTML page** (or PDF file) you want to set as the index page. An HTML file inside a subfolder is accepted.

It is possible to choose a PDF document as the homepage, but this is recommended only if you want to convert a single PDF file to a secure ebook ("PDF to EXE").

3) Options for your ebook

HTML Executable allows you to completely customize the interface of your publication and ebook application. This step lets you choose the GUI theme you want for it. You can change it later.

- Browser with classic toolbar and menubar (classical look)
- Browser with ribbon and application menu button (more modern look)
- Browser only: no other pre-defined UI item (for instance, you already have a UI defined in your HTML app).

Moreover, you can enable the secure built-in PDF viewer or not (this can be enabled/disabled later). Same for DOCX viewer.

4) Last step: output settings

You have

1. to specify **the path to the publication / ebook .exe file** that will be output by HTML Executable. It must be a full path including directory, filename and extension. You can select it by clicking the Browse button. It is recommended that you **do not put your output .exe** file in the source folder, but rather in a different directory. **Ensure that the destination directory is writeable**.

2. to **give a title to your publication**. The title will appear on all dialog and message boxes displayed to end users as well as in the Windows task bar. HTML Executable should have parsed your index page and read the value of the `<TITLE>` HTML tag if any. You can of course choose a different title.

IMPORTANT: if your title contains a quote (`"`) character, be sure to replace it by two quotes `""` in order to avoid script compilation errors.

Finally, click **Finish** to create your project.

HTML Executable will then prepare a blank project, configure all settings by default and automatically add files from the source folder. It generally takes a few seconds depending on the number of files you have in your source folder.

When it is ready, the "Main Window" page is displayed: you are ready to edit your project and you can **also immediately compile it if you wish by clicking** at the top (or press **F9**).

How to Customize the Main Window

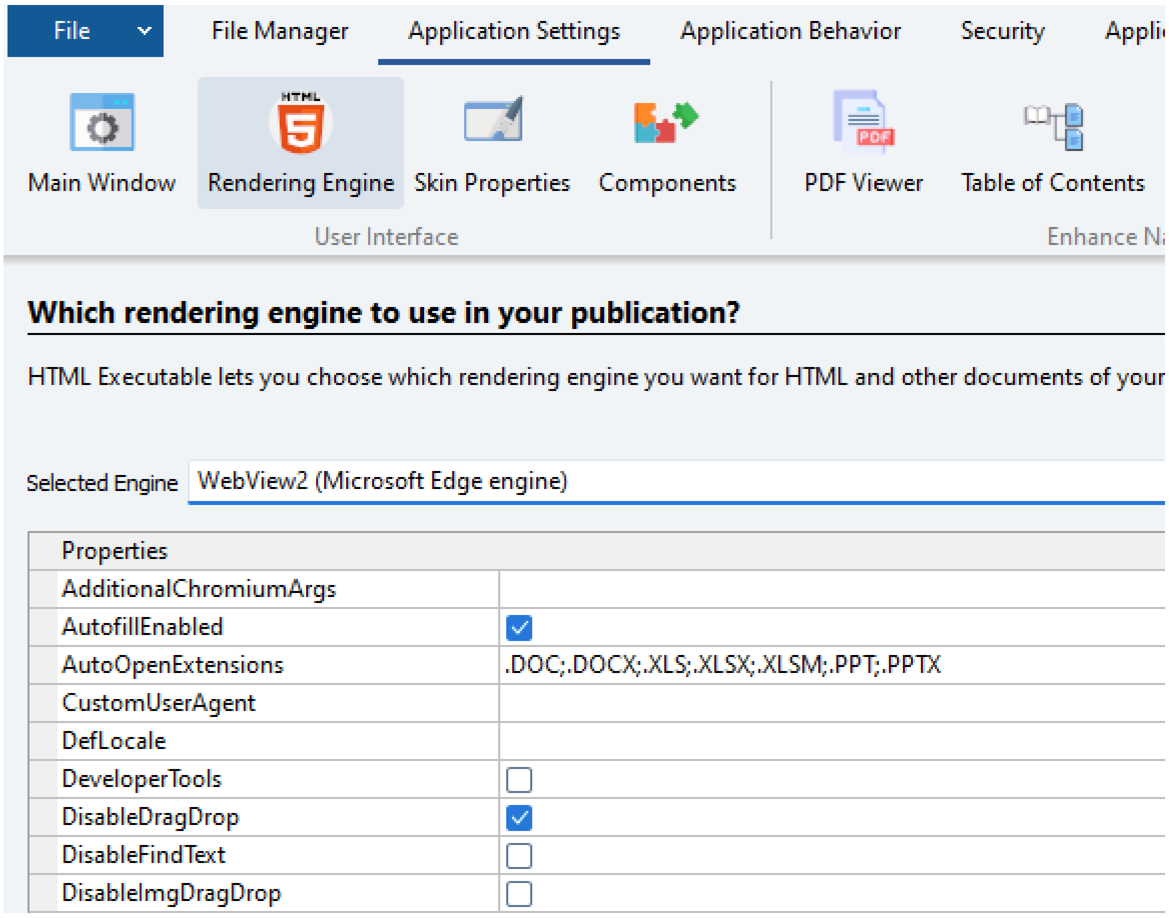
Using the file manager

3. Create Ebooks And Publications

3.1. Choosing The Rendering Engine

HTML Executable offers HTML Executable offers **two rendering engines** for your ebooks and publications: the Chromium Embedded Framework (CEF) and WebView2. Each engine has unique advantages and disadvantages. Understanding the differences between these two engines can help you select the one that's most suitable for your needs. Both offer full support for HTML5, CSS3, JavaScript, media files, and so on.

HTML Executable lets you change the rendering engine whenever you like. We made the different resources used by the software compatible between the two engines.



Which rendering engine to use in your publication?

HTML Executable lets you choose which rendering engine you want for HTML and other documents of your

Selected Engine: WebView2 (Microsoft Edge engine)

Properties	
AdditionalChromiumArgs	
AutofillEnabled	✓
AutoOpenExtensions	.DOC;.DOCX;.XLS;.XLSX;.XLSM;.PPT;.PPTX
CustomUserAgent	
DefLocale	
DeveloperTools	☐
DisableDragDrop	✓
DisableFindText	☐
DisableImgDragDrop	☐

Some Hints

CEF can be a preferable choice for older Windows versions, such as Windows 7, where the WebView2 runtime might not be installed by default. WebView2 is perfectly suited for Windows 11. Furthermore, if your website or ebook contains MP4 videos with certain proprietary codecs, WebView2 is likely a more reliable choice. You should test your website with HTML Executable to determine which engine works best for you.

Chromium Embedded Framework (CEF)

CEF is a simple framework for embedding Chromium-based browsers in other applications. It's the same rendering engine used in Google Chrome, which means that it offers high performance and compatibility with modern web technologies.

Advantages of CEF

- Excellent performance and compatibility with modern web technologies.
- Does not rely on system updates or external dependencies for its core rendering.
- Creates self-contained ebooks and applications (self-contained, requiring no additional downloads).
- A single EXE file for your application allows for easy distribution.

Disadvantages of CEF

- The CEF engine can result in larger application EXE file sizes, leading to ebooks or publications that are several tens of megabytes in size.
- Does not include proprietary media codecs (MP4 videos won't be played unless you use the dedicated CEF build that offers MP4 support but requires specific licensing).

WebView2

WebView2 is a free component from Microsoft that uses the same Chromium-based engine found in the Edge browser. It's a part of Windows 11 and is also available for previous versions of Windows through Windows Update or a stand-alone installer available for free.

Advantages of WebView2

- Excellent performance and compatibility with modern web technologies.
- As a component often pre-installed with Windows or easily updatable, it simplifies deployment as you don't need to bundle Chromium dependencies.
- It can result in smaller EXE file sizes compared to CEF, as the entire framework is not bundled with your application.
- Regularly updated by Microsoft to incorporate new HTML5 features.
- A single EXE file for your application allows for easy distribution.

Disadvantages of WebView2

- It relies on system updates and might not be available on all systems, especially older ones, or may require users to install/update it.
- If WebView2 is not present, HTML Executable will prompt end users to install the component and provide instructions.

Changing the Rendering Engine

To select the rendering engine in HTML Executable:

1. Navigate to Application Settings / Rendering Engine.
2. Choose your preferred engine from the "Selected Engine" dropdown list.

In this page, you can also configure various properties for the selected engine. For example, in both cases, you can enable the Developer Tools by ticking the "DeveloperTools" option. This will allow users to access the Developer Tools within the ebook by right-clicking on the webpage they wish to inspect.

Remember to save your project settings after making any changes to the rendering engine or its properties.

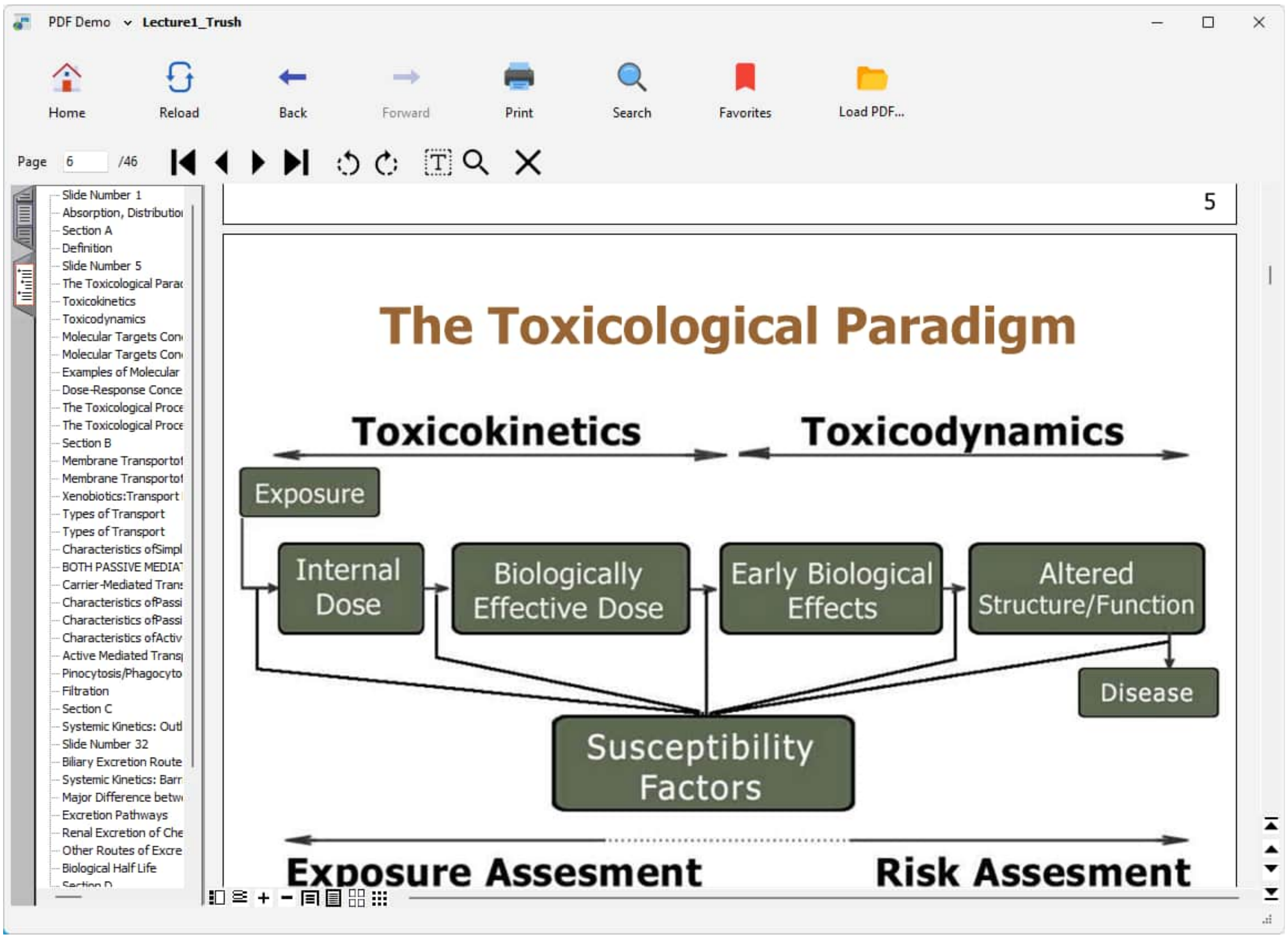
💡 Choosing the right rendering engine for your project depends on your specific needs and the nature of your ebook or publication. Consider the trade-offs between file size and system dependencies when making your decision.

3.2. Displaying PDF and Securing PDF as Ebooks

Besides HTML pages, PDF documents are also completely supported by HTML Executable. This means that, like HTML files, PDF files can be included in your ebooks or applications. Users of the software can view PDF files without the use of any extra plugins or software because HTML Executable offers a stand-alone PDF viewer. Additionally, HTML Executable may offer your PDF content the same strong protection it does for HTML, restricting user permissions (such as copy and print), blocking snapshots, and protecting the source files from copying.

Using the HTML Executable's built-in PDF viewer lets you **keep your PDF documents as secure as possible**: they are not unpacked to the hard disk, so they cannot be copied by your users. It is also not possible to use the "Save" feature to save the PDF file on the hard disk. Printing is optional and can be forbidden. Moreover, printing with a PDF printer from the built-in PDF viewer is not allowed.

Learn all features and hints about the built-in PDF Viewer



Watch our video tutorial about how to convert a PDF into an ebook with HTML Executable. Moreover, we added some additional customization steps like: disabling Print, Print Screen..., adding a password protection (and you can even go further thanks to the security features of HTML Executable), integrating a splash screen, changing the ebook window's look, creating a custom icon for your ebook EXE file thanks to our icon software GConvert - and much more.

Displaying PDF documents in your ebook or publication

To **enable the built-in PDF viewer**, go to Application Settings => PDF Viewer and turn on: "Use the built-in PDF viewer in this publication".

Then, PDF files are displayed as if they were HTML pages. For instance, when an end user clicks a link to a PDF file, the application will show the PDF directly.

If the built-in PDF viewer is not used, the default PDF reader provided by the WebView2 or Chromium engines will be used.

In cases you are unable to display PDF documents correctly, add the "_heopenit" target to the anchor tag as shown in the following HTML code:
```
<a href="mydoc.pdf" target="_heopenit">Open my PDF document</a>
```

This will cause the publication to open the "mydoc.pdf" file in the default PDF reader.

HTML Executable also supports viewing and securing DOCX files.

3.2.1. Features of the Stand-Alone PDF Viewer

HTML Executable lets you integrate a **stand-alone PDF viewer** in your publications and ebooks. Your end users can view and read PDF documents compiled in your publication without the need of Adobe Reader or any other PDF reader.

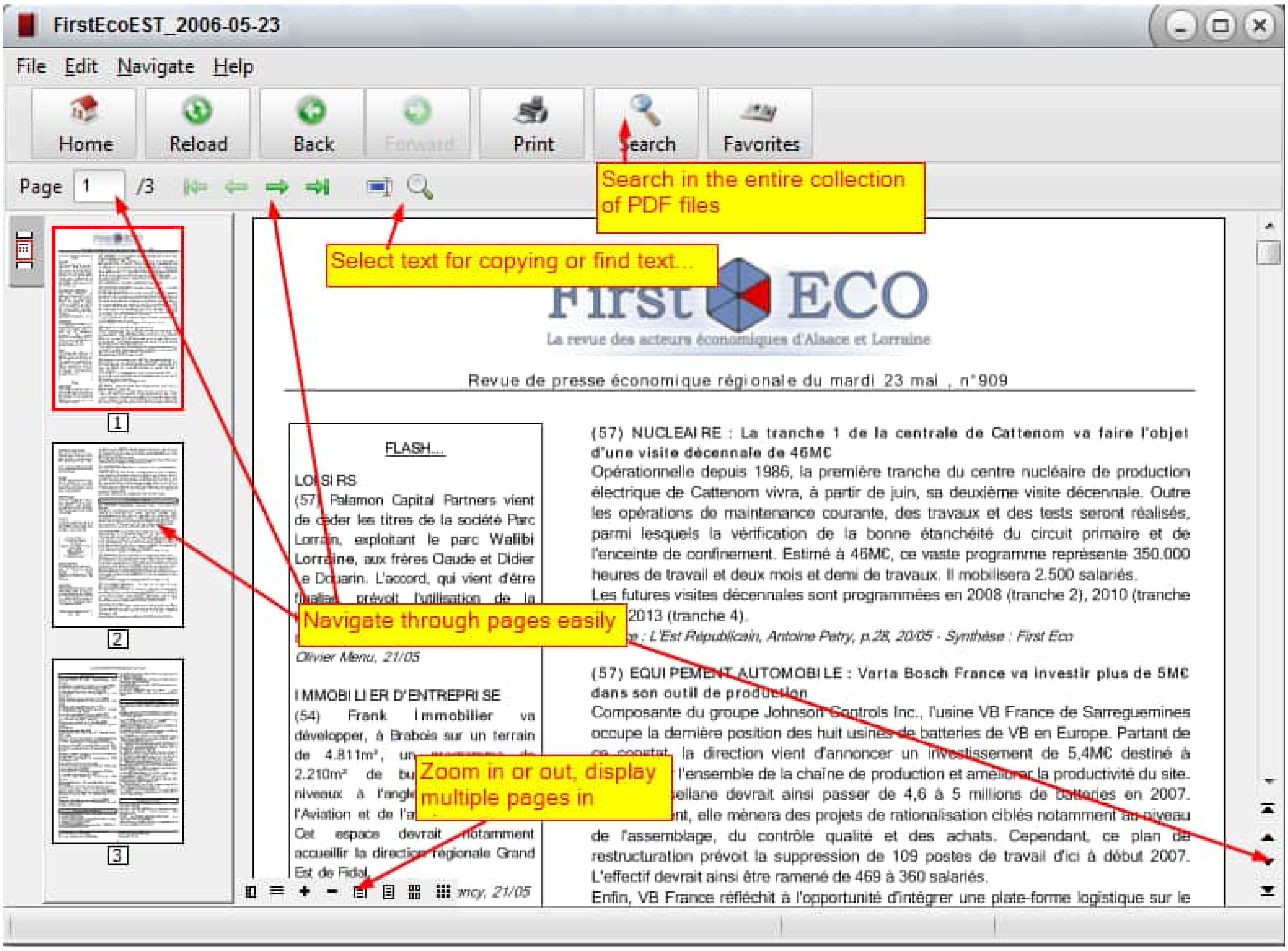
About the Built-in PDF Viewer

The built-in PDF engine **handles almost all PDF documents fine**: PDF files created by popular printer drivers, Adobe Acrobat, Microsoft Office or OpenOffice should be displayed and printed fine. However, PDF files created by imaging and advanced publishing tools may not be properly displayed. If this is the case, printing your PDF document using a printer driver such as PDF Creator may help.

Please keep in mind that **the PDF engine is small and it cannot compete with professional reader tools** like Adobe Reader. The primary goal of the PDF engine is to display simple documents. Anyway, if you get troubles with a PDF file, you can **submit it** to our technical support team for review. We do not guarantee however that we can come up with a solution in all cases. However, the defect PDF file may be used to improve the engine.

Features available with the Built-in PDF Viewer

HTML Executable comes with a **sample named "PDF Viewer" that illustrates the following features**. You can compile this sample either by going to the "HTML Executable Samples PDF Sample" folder on your hard drive, or choose "Samples => PDF Viewer" when you start HTML Executable.


FirstEcoEST_2006-05-23
File
Edit
Navigate
Help
Home
Reload
Back
Forward
Print
Search
Favorites
Page 1 /3
Search in the entire collection of PDF files
Select text for copying or find text...
Navigate through pages easily
Zoom in or out, display multiple pages in
First ECO
La revue des acteurs économiques d'Alsace et Lorraine

Revue de presse économique régionale du mardi 23 mai , n°909

FLASH...

LOISIRS
(57) Palamon Capital Partners vient de céder les titres de la société Parc Lorrain, exploitant le parc **Walibi Lorraine**, aux frères Claude et Didier Le Douarin. L'accord, qui vient d'être finalisé, prévoit l'utilisation de la

Olivier Menu, 21/05

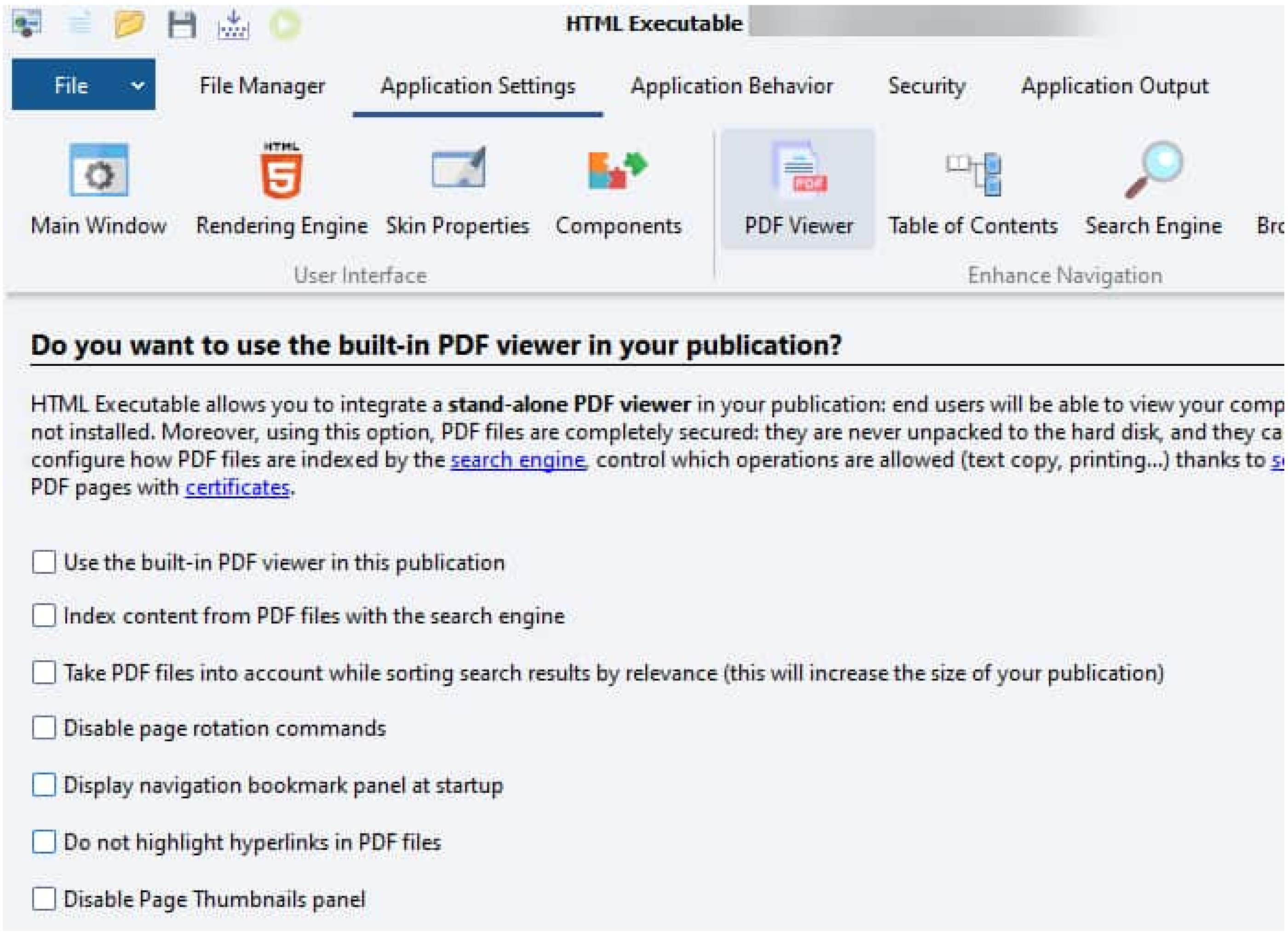
IMMOBILIER D'ENTREPRISE
(54) Frank Immobilier va développer, à Brabois sur un terrain de 4.811m², un [illegible] de 2.210m² de bu[illegible] niveaux à l'angle [illegible] l'Aviation et de l'a[illegible] Cet espace devrait notamment accueillir la direction régionale Grand Est de Fidal.

[illegible]ncy, 21/05

(57) NUCLEAIRE : La tranche 1 de la centrale de Cattenom va faire l'objet d'une visite décennale de 46M€
Opérationnelle depuis 1985, la première tranche du centre nucléaire de production électrique de Cattenom vivra, à partir de juin, sa deuxième visite décennale. Outre les opérations de maintenance courante, des travaux et des tests seront réalisés, parmi lesquels la vérification de la bonne étanchéité du circuit primaire et de l'enceinte de confinement. Estimé à 46M€, ce vaste programme représente 350.000 heures de travail et deux mois et demi de travaux. Il mobilisera 2.500 salariés.
Les futures visites décennales sont programmées en 2008 (tranche 2), 2010 (tranche [illegible] 2013 (tranche 4).
[illegible]e : L'Est Républicain, Antoine Petry, p.28, 20/05 - Synthèse : First Eco

(57) EQUIPEMENT AUTOMOBILE : Varta Bosch France va investir plus de 5M€ dans son outil de production
Composante du groupe Johnson Controls Inc., l'usine VB France de Sarreguemines occupe la dernière position des huit usines de batteries de VB en Europe. Partant de ce constat, la direction vient d'annoncer un investissement de 5,4M€ destiné à [illegible] l'ensemble de la chaîne de production et améliorer la productivité du site. [illegible]ellane devrait ainsi passer de 4,6 à 5 millions de batteries en 2007. [illegible]nt, elle mènera des projets de rationalisation ciblés notamment au niveau de l'assemblage, du contrôle qualité et des achats. Cependant, ce plan de restructuration prévoit la suppression de 109 postes de travail d'ici à début 2007. L'effectif devrait ainsi être ramené de 469 à 360 salariés.
Enfin, VB France réfléchit à l'opportunité d'intégrer une plate-forme logistique sur le

You may customize the behavior of the built-in PDF Viewer on the Application Settings => PDF Viewer page:



Do you want to use the built-in PDF viewer in your publication?

HTML Executable allows you to integrate a **stand-alone PDF viewer** in your publication: end users will be able to view your comp
not installed. Moreover, using this option, PDF files are completely secured: they are never unpacked to the hard disk, and they ca
configure how PDF files are indexed by the search engine, control which operations are allowed (text copy, printing...) thanks to s
PDF pages with certificates.

- [ ] Use the built-in PDF viewer in this publication
- [ ] Index content from PDF files with the search engine
- [ ] Take PDF files into account while sorting search results by relevance (this will increase the size of your publication)
- [ ] Disable page rotation commands
- [ ] Display navigation bookmark panel at startup
- [ ] Do not highlight hyperlinks in PDF files
- [ ] Disable Page Thumbnails panel

Thumbnails, bookmarks, text selection, page printing, page navigation, page rotation, zoom in and out functions are available.

Note: you can **disable the "page rotation**" feature. Page rotation is not compatible with text selection. You can also **disable the View Commands panel**.

It is possible to enable the **display of the navigation bookmark panel** at startup: when the PDF is loaded, its bookmarks are automatically listed. You can disable this feature at any time thanks to the HEShowPDFBookmarks global variable. You can set the with of the panel thanks to **Navigation Panel Initial Width** (in pixels).

Thumbnails are enabled by default. If you want to disable them, enable "**Disable Page Thumbnails panel**".

The PDF viewer can **remember last current page of PDF files**: when a user navigates from a PDF to another document, the PDF viewer stores the page number of the PDF. If the user comes back to the latter, the last page is restored. A menu option is also made available so that end users can disable this behavior if they want.

Non-Latin characters are supported.

PDFs are indexed by the search engine like HTML pages and included in search results if you enable "**Index content from PDF files with the search engine**" option in the Application Settings => PDF Viewer page. Specific PDF files can be excluded from the search thanks to File Properties - Security. It is possible to index PDF files even if the built-in PDF viewer engine is disabled. In that case, the default PDF reader will be used to display documents when a user clicks a search result.

However, after a search process, results from PDF documents will appear at the end of the list. If you want results from PDF documents to be treated as HTML ones when sorting by relevance, you need to turn this other option on: "**Take PDF files into account while sorting search results by relevance**". But note that this will automatically **make your publication bigger** (more or less depending on the size of your PDFs), since HTML Executable must convert PDF documents to raw text data and store this data into the publication. By default, this option is turned off to keep publications small.

Security profiles can be applied to PDF documents too: you can restrict access to PDF documents, define user rights like text copying, page printing...

The PDF engine displays the document title (read from PDF metadata) in the title bar and in search results.

Hyperlinks are supported by the PDF viewer. By default, these hyperlinks are highlighted (their background is grey) to inform the user they can be clicked (no hand cursor is displayed because the engine does not handle it). You can prevent this highlighting by checking the "**Do not highlight hyperlinks in PDF files**" option.

PDF files can be used in the Table of Contents.

Note: it is not possible to highlight a searched item in a PDF file (without HTML) and NOT allow the text to be copied using HTML Executable.

See PDF Viewer Advanced Features

3.2.2. Advanced Features for the PDF Viewer

For advanced users, here are some additional features provided by the built-in PDF viewer of HTML Executable.

Opening PDF to specific pages or bookmark

You can link to specific pages by number or named locations:

For a specific page by number:

```
<a href="file.pdf#page=3">Link text</a>
```

For a named location (destination):

```
<a href="file.pdf#nameddest=TOC">Link text</a>
```

Loading external PDF files with HEScript

You can **load external PDF files** directly in the publication thanks to the HEScript command named "LoadPDFFromFile". Example:

```
procedure OpenPDFDialog; var Path: String;
begin
Path := OpenFileDialog("Choose the PDF to open", ".pdf", ".pdf", "PDF Files
(*.pdf)", "");
if Path = "" then exit;
LoadPDFFromFile(Path);
end;
```

Sending commands to the PDF viewer engine

When the PDF viewer engine loads a PDF document, it triggers the OnPDFDisplay event. You can then pass commands thanks to the built-in HEScript functions named PDFViewerCommand and PDFViewerCommandStr.

List of available commands (and description / parameter value for PDFViewerCommand)

Command ID	Description
54	Select the paper color. Param is a RGB value.
141	Choose the renderer for screen display. The default is Param=1 which selects the GDI+ Renderer, Para better scaled images (antialias), but implements only basic clipping. See code example below.
30	Show printer selection and setup dialog. Param = 0
32	Show the printer dialog. Param = 0
22	Go to the page number specified by Param. The first page has number 0.

The following script example lets you switch to the AGG renderer (insert it in UserMain)

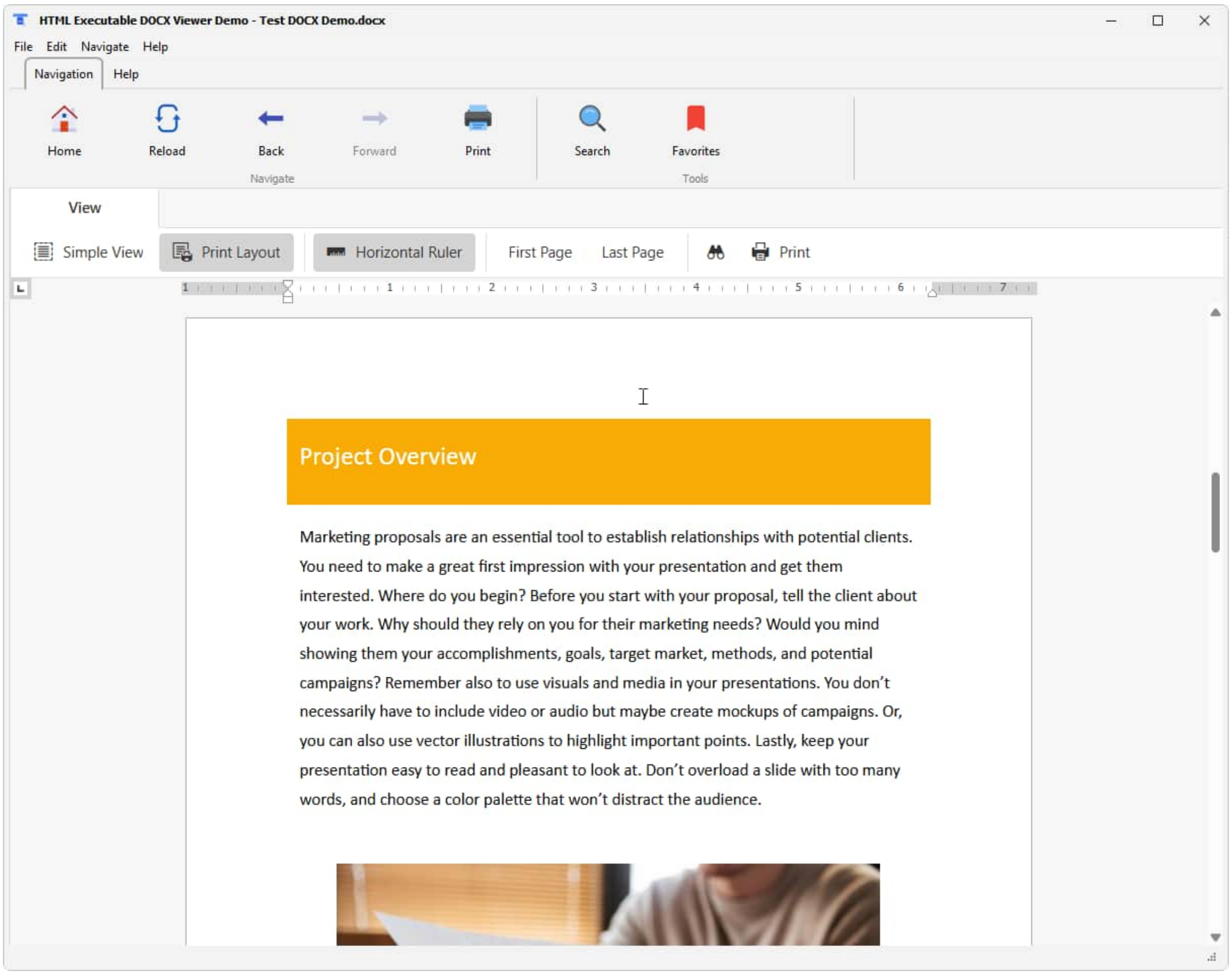
```
procedure OnPDFDisplay(PDFPath: String);
begin
PDFViewerCommand(141, 0);
end;
```

3.3. Displaying DOCX Files and Securing DOCX as Ebooks

In addition to HTML pages and PDF documents, HTML Executable has robust support for Microsoft Word DOCX files. This means DOCX files can be seamlessly included in your ebooks or applications. End users can view DOCX files directly within the publication thanks to its built-in **DOCX viewer, even if Microsoft Word is not installed on their computer**. Furthermore, HTML Executable extends its strong content protection capabilities to your DOCX content, allowing you to restrict user permissions (such as copy and print), block screenshots, and safeguard the source files from unauthorized copying.

Utilizing the HTML Executable's **built-in DOCX viewer ensures your Word documents remain as secure as possible**: they are not unpacked to the hard disk, preventing users from copying them. The "Save" feature is disabled for these embedded documents. Printing can be optionally allowed or forbidden, and security profiles can be applied to control specific actions like text selection or copying.

Learn all features and hints about the built-in Word (DOCX) Viewer


HTML Executable DOCX Viewer Demo - Test DOCX Demo.docx
File Edit Navigate Help
Navigation Help
Home Reload Back Forward Print Search Favorites
Navigate Tools
View
Simple View Print Layout Horizontal Ruler First Page Last Page Print

Project Overview

Marketing proposals are an essential tool to establish relationships with potential clients. You need to make a great first impression with your presentation and get them interested. Where do you begin? Before you start with your proposal, tell the client about your work. Why should they rely on you for their marketing needs? Would you mind showing them your accomplishments, goals, target market, methods, and potential campaigns? Remember also to use visuals and media in your presentations. You don't necessarily have to include video or audio but maybe create mockups of campaigns. Or, you can also use vector illustrations to highlight important points. Lastly, keep your presentation easy to read and pleasant to look at. Don't overload a slide with too many words, and choose a color palette that won't distract the audience.

Displaying DOCX documents in your ebook or publication

To **enable the built-in DOCX viewer**, go to Application Settings => Word Viewer and turn on: "**Use the built-in Word Viewer in this publication**".

Once enabled, DOCX files are displayed similarly to HTML pages or PDF documents. For instance, when an end user clicks a link to a DOCX file compiled in your publication, the application will display the DOCX content directly.

If the built-in DOCX viewer is not used (option is disabled), and a user attempts to open a DOCX file, HTML Executable will attempt to open it using the system's default application for DOCX files (usually Microsoft Word). This requires Word to be installed on the user's computer and the file will be temporarily extracted, which is less secure. The "_heopenit" target can also be used for this purpose:

HTML
```html
<a href="mydoc.docx" target="_heopenit">Open my Word document</a>
```
HTML

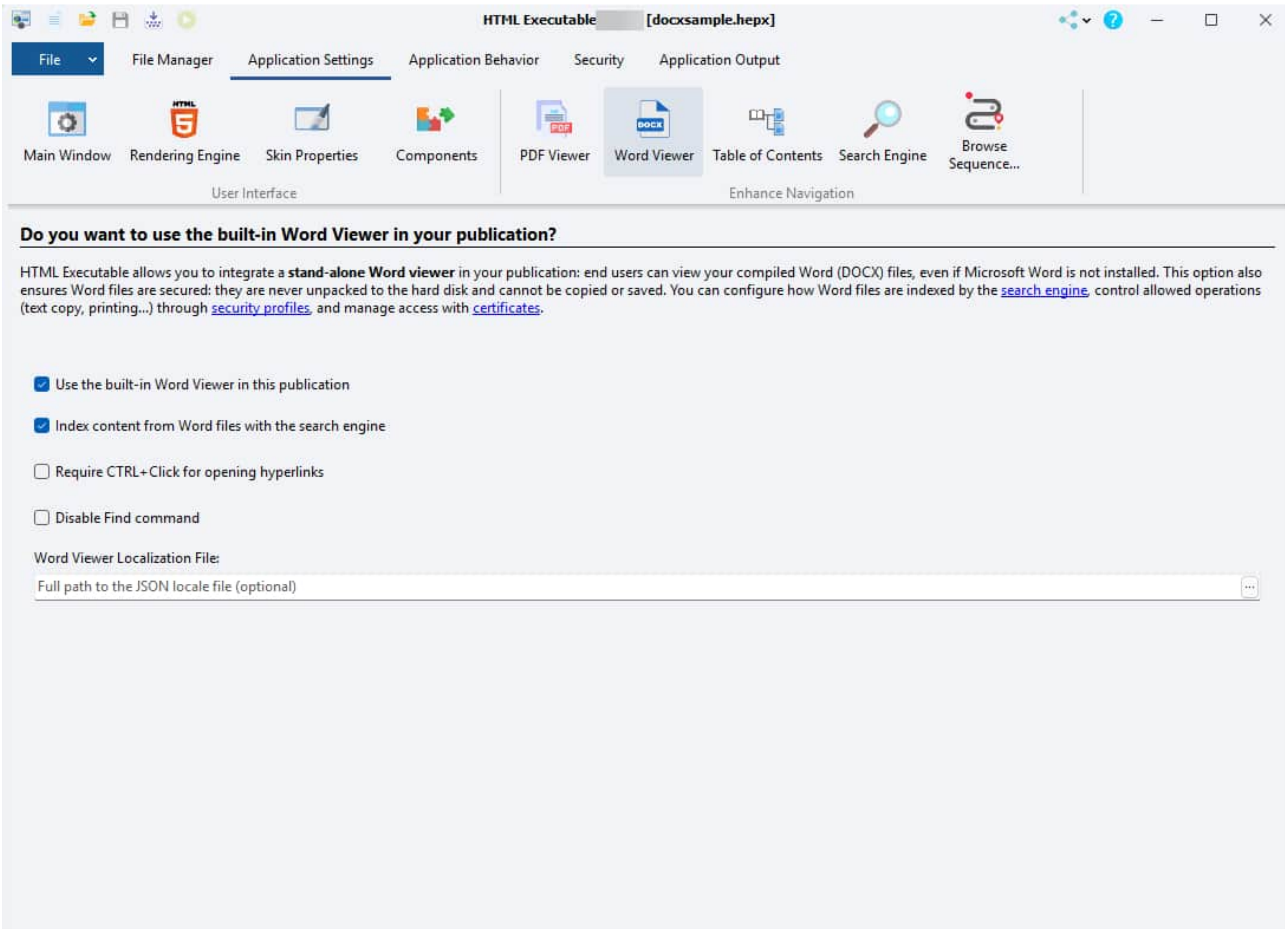
3.3.1. Features of the Stand-Alone Word (DOCX) Viewer

HTML Executable offers a **built-in viewer for Microsoft Word (.docx) files**. This allows your end users to view DOCX documents compiled into your publication without needing Microsoft Word installed on their system. The viewer is designed to be secure and highly integrated with HTML Executable's features.

About the Built-in DOCX Viewer

The built-in DOCX engine aims to render most DOCX documents accurately, including formatting, images, and tables. As with any complex file format, very intricate or unusually formatted DOCX files might display minor differences compared to Microsoft Word. However, for standard documentation, reports, and manuals, the viewer provides a reliable and secure way to present content.

You can customize the behavior of the built-in DOCX Viewer on the Application Settings => Word Viewer page:



Do you want to use the built-in Word Viewer in your publication?

HTML Executable allows you to integrate a **stand-alone Word viewer** in your publication: end users can view your compiled Word (DOCX) files, even if Microsoft Word is not installed. This option also ensures Word files are secured: they are never unpacked to the hard disk and cannot be copied or saved. You can configure how Word files are indexed by the search engine, control allowed operations (text copy, printing...) through security profiles, and manage access with certificates.

- [x] Use the built-in Word Viewer in this publication
- [x] Index content from Word files with the search engine
- [ ] Require CTRL+Click for opening hyperlinks
- [ ] Disable Find command

Word Viewer Localization File:

Full path to the JSON locale file (optional)

Key Features

No Microsoft Word Required: End users can view DOCX files even if they don't have Word or Office installed.

Secure Viewing: DOCX files are never unpacked to the hard disk, preventing unauthorized copying or saving by end users.

User Interface Elements: The viewer provides a familiar interface with options like: Print Layout View, Horizontal Ruler (toggle), Navigation: First Page, Last Page, Page Up, Page Down. Print (if enabled) and Zoom controls.

Search Integration: Option to index content from DOCX files with the publication's search engine. The "Find" command within the viewer can also be enabled/disabled.

Security Profiles: Apply security profiles to DOCX documents to control user rights, such as disabling text selection, copying, printing, or the "Find" command.

Hyperlink Control: An option to "Require CTRL+Click for opening hyperlinks" can be enabled to prevent accidental navigation.

Localization: The DOCX viewer's interface (e.g., tooltips, dialogs) can be localized using a specified JSON locale file. You can find a French locale sample in the "Resources\WordView\locales" subfolder of your HTML Executable installation.

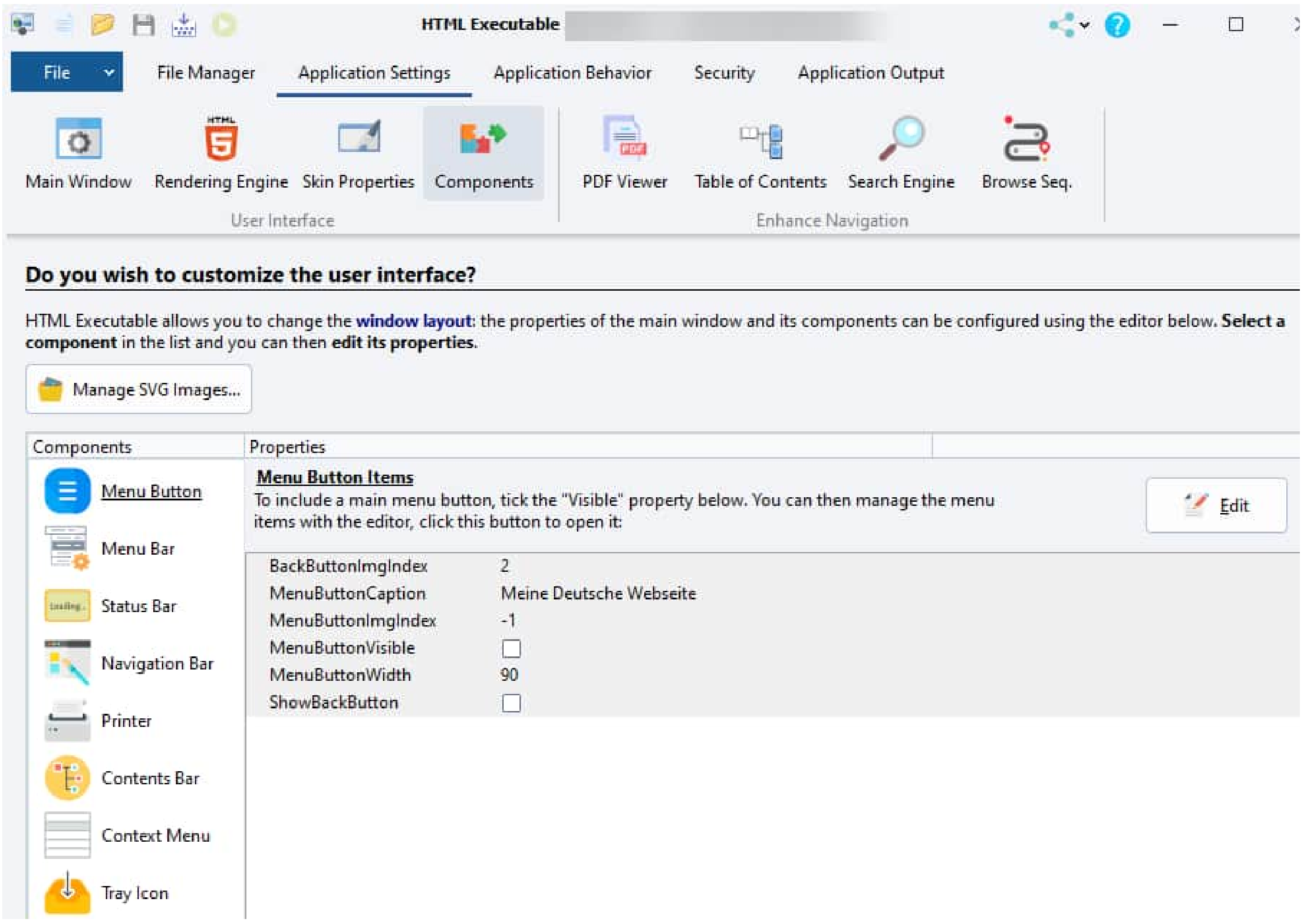
Screenshot Protection: Like other content in HTML Executable, you can enable measures to prevent screenshots of sensitive Word documents using the Content Protection features.

Table of Contents Integration: DOCX files can be included as entries in the publication's Table of Contents.

3.4. Designing User Interface

HTML Executable lets you **create modern and complex user interfaces for your ebooks and HTML applications with its visual editor**. To edit the interface of your application, go to **Application Settings => Components**.

For instance, you can create and display a Table of Contents for your ebook to improve navigation, add more commands to the navigation bar or use your own Application Menu button.



Do you wish to customize the user interface?

HTML Executable allows you to change the **window layout**: the properties of the main window and its components can be configured using the editor below. **Select a component** in the list and you can then **edit its properties**.

Manage SVG Images...

Menu Button Items
To include a main menu button, tick the "Visible" property below. You can then manage the menu items with the editor, click this button to open it:

BackButtonImgIndex	2
MenuButtonCaption	Meine Deutsche Webseite
MenuButtonImgIndex	-1
MenuButtonVisible	☐
MenuButtonWidth	90
ShowBackButton	☐

You have the list name **Components** and sometimes a **properties editor** on the right side.

Components

This list contains all of the components of the user interface that can be customized.

The following components are available for you to customize. Please refer to the corresponding topic for further help:

- Menu Bar
- Status Bar

For instance, if you do not want to use a component in your ebook, you can turn its **Visible** property to False.

Each component has its own properties and can be configured with the properties editor. To edit the properties of a component, just **select it in the list**.

See also: Manage SVG Images (common images in SVG format for your HTML Executable ebook or application project).

Edit component properties

The **properties editor** lets you modify the properties of the selected component: it works like the one you can find in Paquet Builder. If you are not familiar with this grid editor, see below for a brief description about how it works.

All changes made are directly saved!

Modify UI components and controls at runtime

HTML Executable supports changing control properties at runtime.

How does the Properties Editor work?

The Properties Editor enables you to set properties for the selected component. By setting properties you are defining the state of a component. If the Properties are arranged by name, the first column lists the names of the selected action's properties: If a plus sign (+) appears beside a property name, this can be clicked to display the sub-properties of that property. These can be a list of possible values when the property represents a set of flags, or sub properties if the property represents an object (the value column gives the name of the object, enclosed in parentheses). Similarly, if a minus sign (-) appears, this can be clicked to collapse the sub properties. When a property has focus, you can also use the keyboard + and - keys to expand or collapse properties.

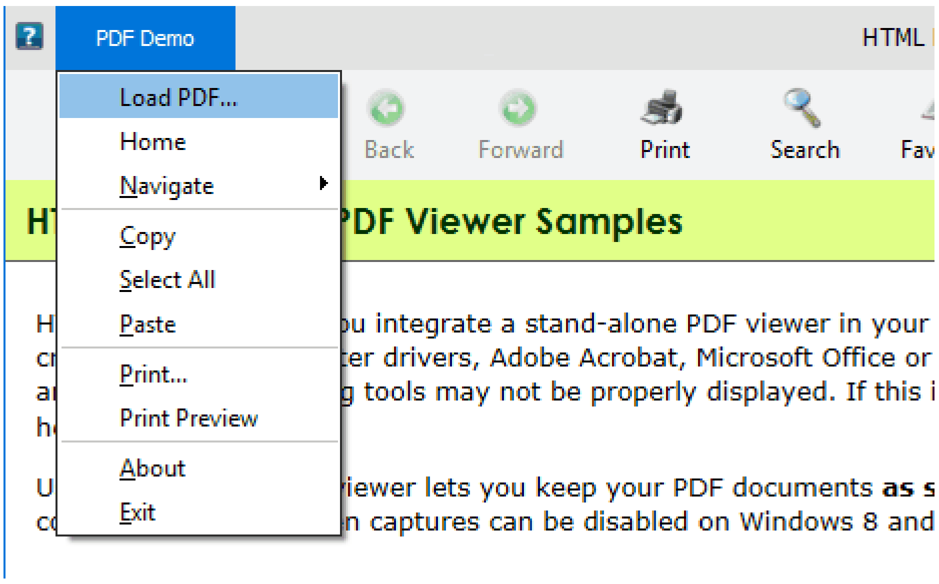
The second column displays the property values: When the property is selected, the value changes to an edit control where you can type a new value. If the value can be set using a dialog, a small button appears when the property is selected. Click this button to display a dialog where you can set the property.

If the value is an enumerated type, a drop-down button appears when the property is selected. Click this button to display a drop-down list that you can use to set the property.

3.4.1. All UI Components

3.4.1.1. Menu Button

The "Menu Button" component displays a top-left App Menu Button that should replace the menu bar. It lets your end users access various commands such as page printing, navigation tools or help information of your ebook.



On the screenshot above, the app menu button is named "PDF Demo". This can be changed thanks to the **MenuButtonCaption** property below.

To enable/disable the menu button, set **MenuButtonVisible** to false.

{{% notice warning %}} IMPORTANT: some skins **DO NOT support displaying the menu button**, so even if MenuButtonVisible is set to true, the menu button will not be rendered. This is not a bug but a lack of the skin you selected. {{% /notice %}}

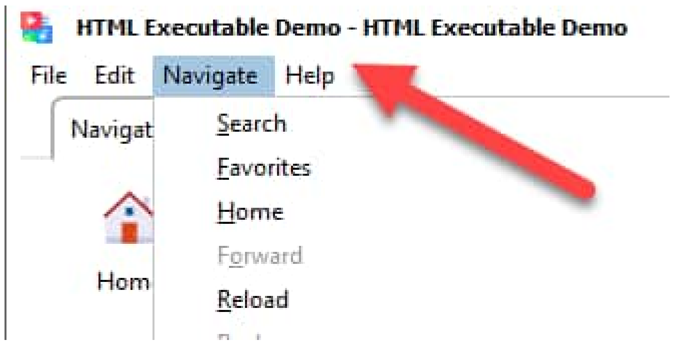
You can add **your own menu items** to the menu displayed by the App Menu button thanks to the menu editor (click **Edit**).

List of properties:

BackButtonImgIndex: number of the image in the SVG list to be used as the icon for the Navigate Back button (see ShowBackButton also). The icon should represent a left arrow.
MenuButtonCaption: text to be displayed on the app menu button.
MenuButtonImgIndex: number of the image in the SVG list to be used as the icon for the Menu button (leave to -1 for no icon).
MenuButtonVisible: if set to false, the app menu button will be removed.
MenuButtonWidth: width of the app menu button in pixels.
ShowBackButton: adds a Navigate Back quick button as in recent Windows versions.

3.4.1.2. Menu Bar

The "menu bar" component displays the **standard GUI menus** at the top side of the main window. It lets your end users access various commands such as page printing, navigation tools or help information of your ebook.



To hide the menu bar, set Visible to false.

You can add **your own menu items** to the menu bar thanks to the menu editor (click **Edit**).

List of properties:

NoFavorites: if True, the publication / ebook will not manage any user favorites. The Favorites panel is not included nor the Favorites button.
Visible: if set to false, the menu bar will be removed.

3.4.1.3. Status Bar

The "status bar" component displays a panel at the bottom of the main window. It **displays information for your end users**, in a similar way to Web browsers. When an end user moves the cursor on a hyper link, the URL may appear in the status bar. To hide the status bar, set Visible to false.

List of properties

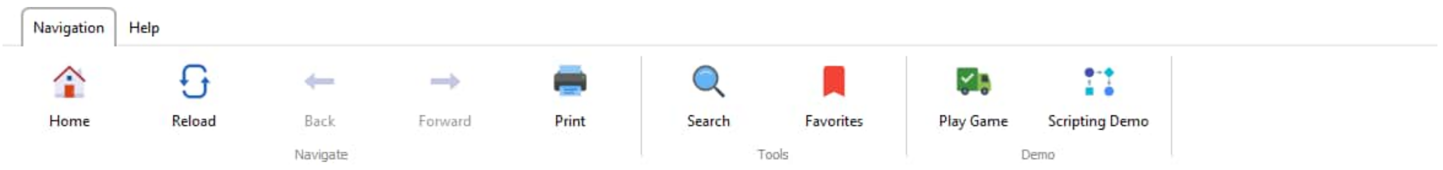
AutoShowURLs: determine whether the publication should display the URL of a link which is under the mouse pointer. Set it to False if you want to keep your internal URLs secret.
DefaultText: the text to be displayed by default when no other information needs to be indicated.
HideProgressBar: disable the progress bar used for large files or long downloads.
Visible: if set to false, the status bar is removed.

Notes

It is recommended that a status bar be included in your publications.

3.4.1.4. Navigation Bar

In HTML Executable, a **Navigation Bar** is a customizable element of the user interface where you can add panels, buttons to access commands, design ribbons similar to the ones used in Microsoft Office, to create simple or complex interfaces tailored to your needs.



You have the flexibility to design your Navigation Bar in a way that best suits your publication or eBook, whether that be a minimalist design with just a few key buttons or a more extensive set up with various panels and features.

A dedicated editor is available for creating and editing your Navigation Bar. This editor allows you to drag-and-drop different components such as buttons and panels to customize the Navigation Bar according to your requirements. More information on how to use this editor is available.

Generally, if you have a navigation bar in your ebook or HTML application, you can at least have these buttons on it:

Home: return to the contents page.
Reload: refresh the current page.
Back: return to the previous loaded page.
Forward: go to the next page (in the case end users have already pressed Back).
Print: displays the page printer dialog.
Search: displays the search panel.

Contents: displays or hides the table of contents (TOC) panel.
Favorites: displays or hides the Favorites panel.

To hide the navigation bar, set its Visible property to false.

Buttons can bear images (icons). To manage these images (in SVG format), use the "**Manage SVG images**" button. Thus, you can import your own icon images in SVG format to create your own personalized buttons.

List of properties

ImageHeight: you can set the height in pixels for the SVG images used on buttons. By default, the height is set to 32 pixels. All buttons share the same image height.
ImageWidth: you can set the width in pixels for the SVG images used on buttons. By default, the width is set to 32 pixels. All buttons share the same image width.
Visible: if set to false, the tool bar is removed.

3.4.1.5. Printer

The "printer" component sets properties for the printer and handles how HTML pages of your ebook / publication are printed.

Allowing or disallowing pages to be printed is not set with this component, but with security profiles.

List of properties

EnablePrintPreview: if true, printing the current webpage will show the Print Preview window first.
FooterText, HeaderText: sets the text to be displayed at the bottom/top of each printed HTML page.
OpenPDFAfterPrint: if set to True, the application will open the newly created PDF file when the end user selected Print to PDF.
PrintLandScape: determines how the page is printed.
SelectionOnly: prints the selection only.
ShowHeaderFooter: indicates whether header and footer should be printed or not.

3.4.1.6. Contents Bar

Adding a table of contents (TOC) to your publication or ebook provides end users **with a hierarchical view of the content**. Users click a topic listed in the table of contents, and are taken directly to the information they are looking for. To enable it, set **Visible to true**.

To manage the items of your Table Of Contents, use the TOC editor (Application Settings => Table Of Contents).

List of properties

AutoExpand: Set `AutoExpand` to `True` to cause the selected item to expand (its children will be visible) and the unselected items to collapse.
AutoFindTopic: If `True`, the ebook / publication will always try to locate the current displayed HTML page in the TOC and make it as selected. Automatically set to `True` if a Browse Sequence is used.
CloseIfSearchPanelOpens: If `True`, the TOC is closed when the search panel is shown and vice versa. This mimics the default behavior of CHM files.
DefaultTarget: You can specify in which frame the documents should open, otherwise if you leave this property blank, the frame system will be ignored. HTML Executable-special targets may be used.
HideSelection: If `True`, the current selected node is hidden when the TOC does not hold the focus.
InitialWidth: The initial width (in pixels) of the TOC. Use this option to set the size so no horizontal scrollbar is displayed.
IsSticky: If `True`, the TOC will always remain visible. End users cannot close it (the Contents button is in this case unnecessary).
ItemHeight: By default, set to 18px (one line). If you would like your TOC to have multiline items, please increase the value.
ShowAtStartup: Displays the TOC tree at startup or not. End users can then use the Table of Contents button (see above) to display it.
ShowButtons: If `ShowButtons` is `True`, a button will appear to the left of each parent item. The user can click the button to expand or collapse the child items as an alternative to double-clicking the parent item.
ShowLines: If `ShowLines` is `True`, lines linking child nodes to their parent nodes are displayed. Nodes at the root of the hierarchy are not automatically linked. To link nodes at the root, the `ShowRoot` property must also be set to `True`.
ShowOnRightSide: If `True`, the contents bar will be displayed on the right side of the window (instead of the left one by default) and other navigation panels will be displayed on the left side.
ShowRoot: To show lines connecting top-level nodes to a single root, set the tree view's `ShowRoot` and `ShowLines` properties to `True`.
Visible: If set to `False`, the contents bar feature will be removed. Do not forget to set this crucial property to `True` if you create a TOC in the TOC editor.

3.4.1.7. Context Menu

The "context menu" component controls the menu that pops up when clicking with the right-hand mouse button. It gives end users of your ebook access to the most-used commands.

You can add your own menu items to the context menu thanks to the menu editor (click **Edit**).

List of properties

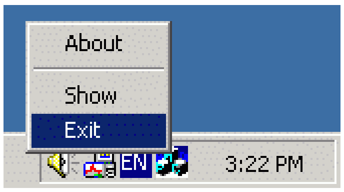
DisableContextMenu: set it to True to remove the context menu.
DisableOpenLinkNewWindow: lets you disable the display of the "Open Link In New Window" menu item when users right-click on links in your webpages.
EnableShowSourceCode: may be useful for debugging purposes. The "Show Source Code" menu item will be displayed if this option is true, and this menu item lets you inspect the full source code of the currently displayed HTML page directly in the "Developer Tools".

Notes

To disable the context menu, you can also use Security Profiles instead of the property above. Moreover, specific commands (like copy to clipboard) can be disabled with Security Profiles too.

3.4.1.8. Tray Icon

Instead of having a button in the taskbar, you may prefer your ebook or HTML application to **show a small tray icon:**



If so, turn on "**Enable Tray Icon**". The small icon displayed is the one of the publication EXE file.

The tray icon optionally displays a popup menu when end users right-click on the tray icon.

Note: in Windows, all tray icons may not always be visible by default. This is a normal behavior and end users can adjust tray icon bar settings to modify it.

Warning: when end users minimize the main window to the tray, all **secondary windows** like pop-ups are always **closed and not restored.**

Default Hint Text

This is the text displayed in the hint when hovering over the tray icon.

Show Minimized on Start

When this option is enabled, the application starts minimized: the main window is not displayed, only its tray icon is visible.

The Close button Minimizes the Main Window

By default, when you hit the Close system button, the application exits. If you want to avoid your application from exiting, use this option. It will force the main window to minimize to the tray when the close button is hit.

In that case, ensure to **give end users an option to exit your publication** (like a menu command in the tray's popup menu). Otherwise, they will need to end the application manually with the task manager.

Always minimize to tray

If you enable this option, the main window is always minimized to the tray, not in the taskbar. Otherwise, a taskbar button will appear when the main window is minimized.

Edit popup menu

Lets you manage the entries for the tray menu. This menu appears when end users right-click on the tray icon. It should at least contain two commands: Restore and Exit. These commands are automatically created by HTML

Executable for each new project and it is recommended that you leave them.

Using the menu editor

3.4.2. Using The Menu Editor

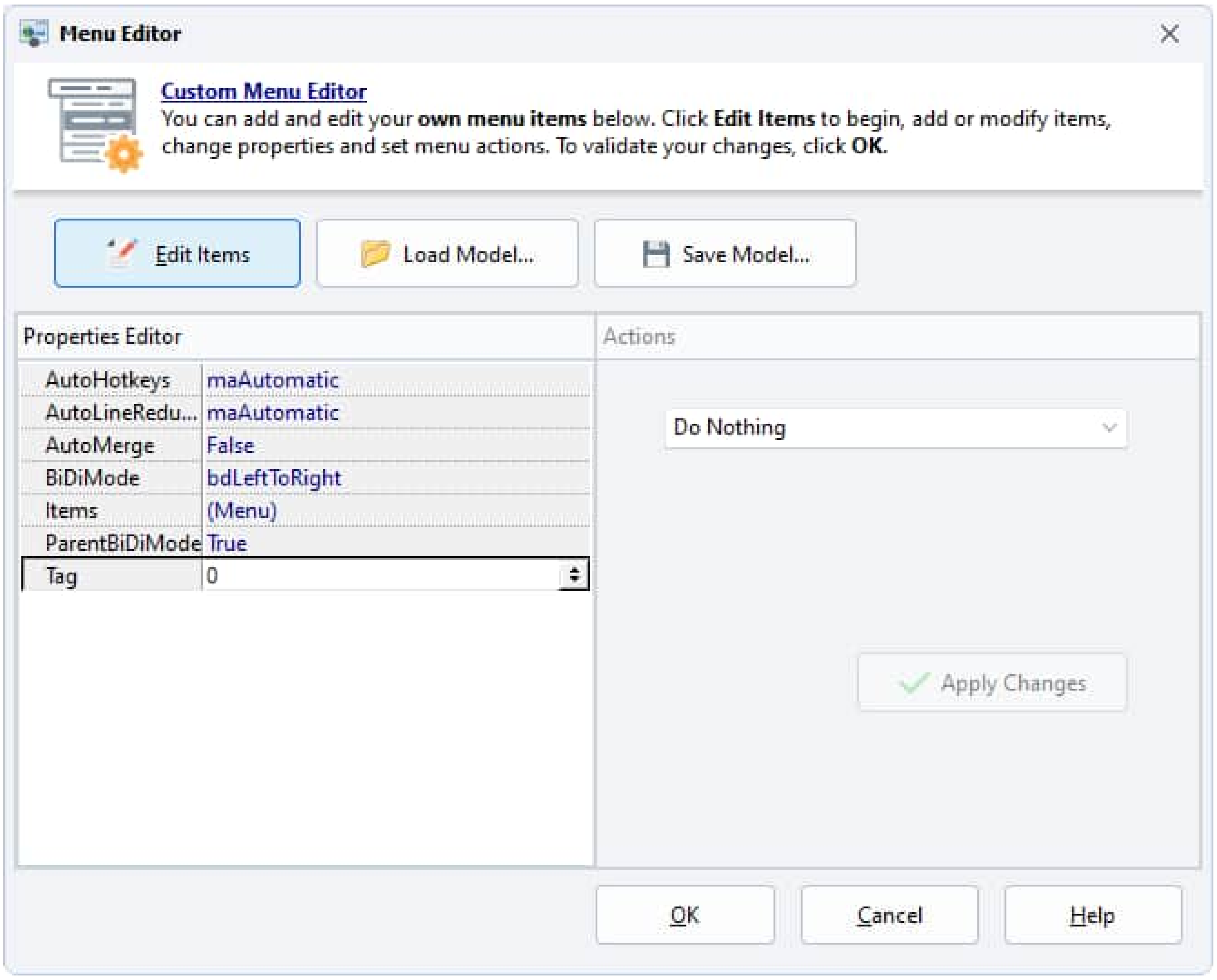
The Menu Editor in HTML Executable is a powerful feature that allows you to manage and customize the menus in your eBook or HTML application. With it, you can access **menu items** of the menu bar, the app menu button, the context menu, the tray icon, and **associate commands with them**.

Accessing the Menu Editor

The **menu bar** displays the standard GUI menus at the top side of the main window (while the App Menu button displays the menu when clicked). By default, a publication contains four pre-defined menu items: *File, Edition, Navigate and Help* with various menu commands that let your end users navigate through your web pages, select and copy text parts, print pages... The properties of these four menu items can be modified thanks to the Menu Bar component. To edit **custom items**, go to the Application Settings => Components page, select "**Menu Bar**" or '**Menu Button**" and click the "**Edit**" button.

The **context menu** pops up when clicking with the right-hand mouse button. It contains common navigation and edition commands like Back, Next, Copy to clipboard... and it can be customized according to your needs with the Context Menu component. Finally, a pop-up menu can be displayed when the end user right-clicks the tray icon if the latter is enabled. You can change the items by clicking "**Edit popup menu**" in the Tray Icon tab.

Menu Editor Overview

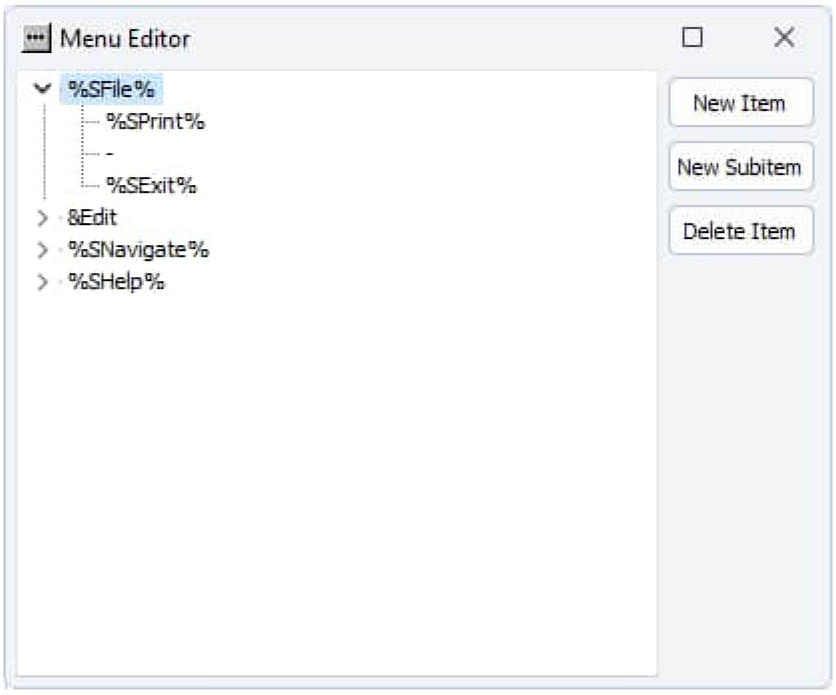

Menu Editor

Custom Menu Editor

You can add and edit your **own menu items** below. Click **Edit Items** to begin, add or modify items, change properties and set menu actions. To validate your changes, click **OK**.

AutoHotkeys	maAutomatic
AutoLineRedu...	maAutomatic
AutoMerge	False
BiDiMode	bdLeftToRight
Items	(Menu)
ParentBiDiMode	True
Tag	0

To access the menu items, click the `**Edit Items**` button or double click on the `Items` property. This will open a sub-window that displays the menu tree, allowing you to manage all menu items:



The menu tree lists all of the menu items. The menu tree is hierarchical. This means that parent menus can have child items. Each item in the tree represents a menu item in your eBook or application.

In case of the main menu bar, the four predefined menus (File, Edition, Navigate and Help) are already in the list: you can select them in order to add new menu children.

In HTML Executable, text enclosed within percentage signs, such as `%SFile%`, are referred to as resource strings. These strings are placeholders that will be replaced by the corresponding localized text defined in the Localization page. For instance, `%SFile%` will be replaced by the text associated with the identifier `SFile` in the list of resource strings. This allows for the proper localization of your application or eBook based on the user's language settings.

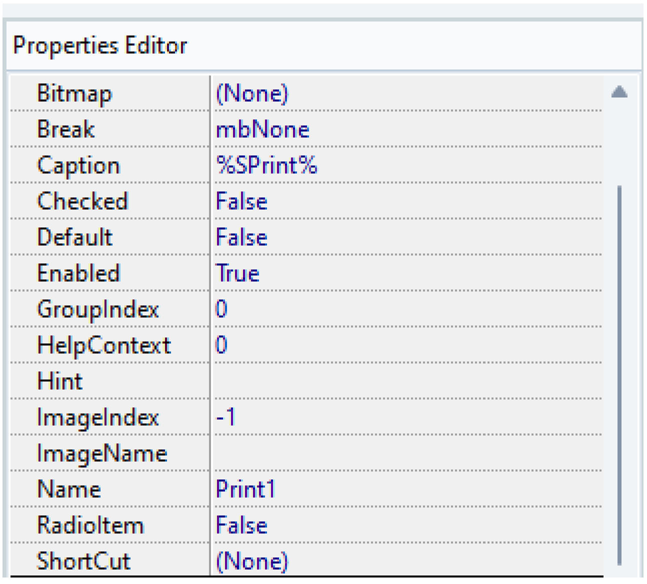
To add a new menu item, click **New Item** to add a top menu item or click **New Subitem**, the new menu item will become a child of the selected item (in the menu tree).

To **edit a menu item**, select it in the menu tree and you can then modify its properties (which are shown in the Properties editor).

To **delete a menu item**, select it in the menu tree and you can then remove it.

You can reorder the menu items using **drag/drop operations**.

Menu Item Properties and Actions



Properties Editor

Bitmap	(None)
Break	mbNone
Caption	%SPrint%
Checked	False
Default	False
Enabled	True
GroupIndex	0
HelpContext	0
Hint	
ImageIndex	-1
ImageName	
Name	Print1
RadioItem	False
ShortCut	(None)

While you can modify all properties, the most important ones are:

Caption: This is the text that will appear on the menu item.
ImageIndex: This property allows you to set an image for the menu item (defined in the SVG image list).

In addition, under the `Actions` section, you can choose the action that will be executed by the eBook or application when the menu item is selected by the user.

Learn more about all actions available

Remember to click `Apply changes` after editing a menu item to save the changes.

Menu separator

Specify a **hyphen character** (-) as the value of Caption for the menu item to indicate that the menu item is a separator.

Separators should have their action set to "Do nothing".

Saving and Loading Menu Templates

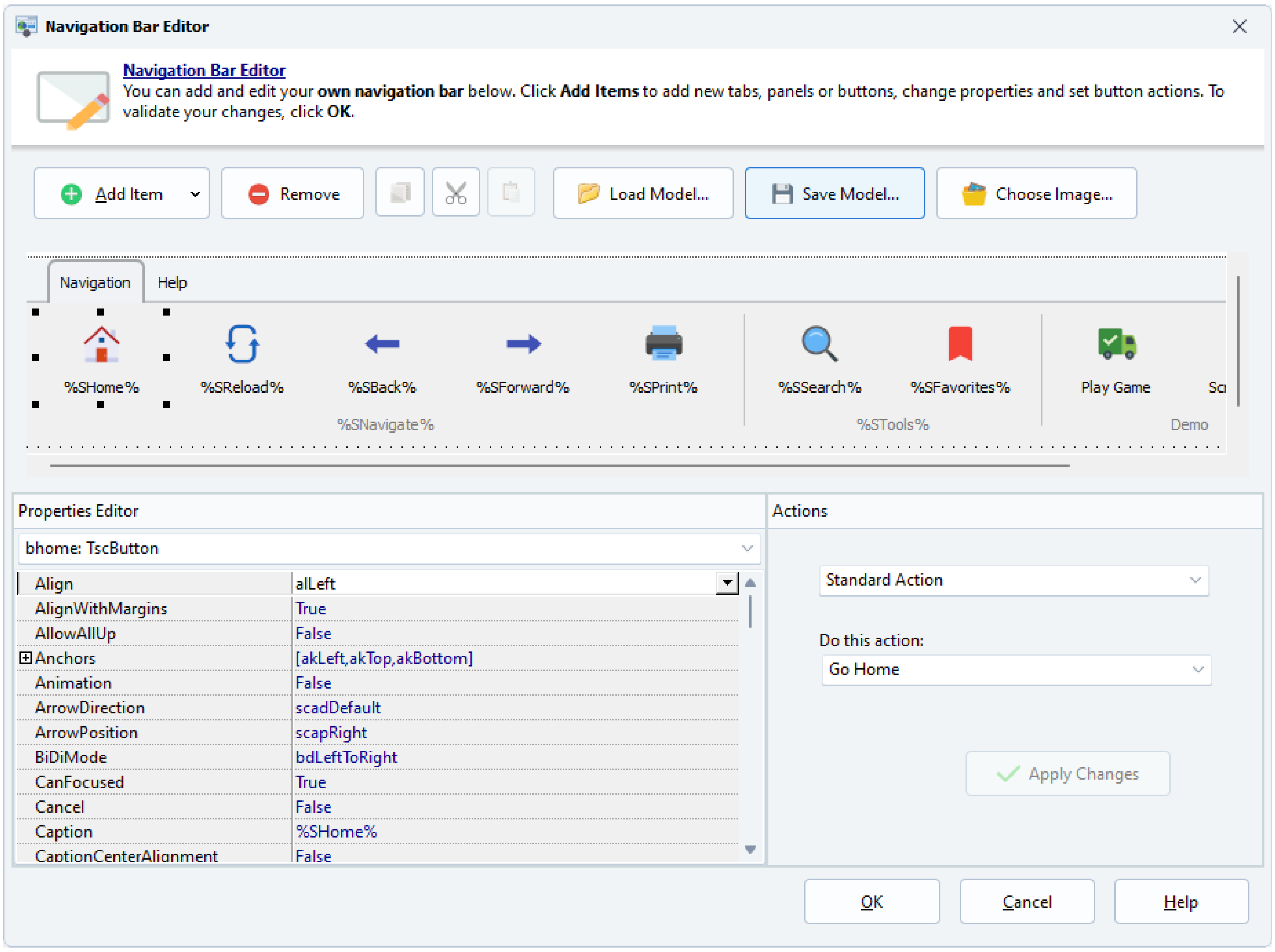
You can save your menu configuration as a template for future use. To do this, click on the `Save model` button. To load a previously saved template, click on the `Load model` button.

3.4.2.1. Video Tutorial: Add A Custom Menu

Watch a video tutorial below about adding a menu item with the Menu Editor in HTML Executable and associating an action.

3.4.3. Navigation Bar Editor

The Navigation Bar Editor is a versatile tool designed to provide you with control over the navigation bar's functionality and aesthetic of your ebook or HTML application. This tool makes the navigation bar highly customizable, enabling users to modify button panels and even design a visual ribbon.


Navigation Bar Editor

Navigation Bar Editor
You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set button actions. To validate your changes, click **OK**.

Properties Editor

bhome: TscButton

Align	alLeft
AlignWithMargins	True
AllowAllUp	False
⊞Anchors	[akLeft,akTop,akBottom]
Animation	False
ArrowDirection	scadDefault
ArrowPosition	scapRight
BiDiMode	bdLeftToRight
CanFocused	True
Cancel	False
Caption	%SHome%
CaptionCenterAlignment	False

Actions

Standard Action

Do this action:
Go Home

Apply Changes

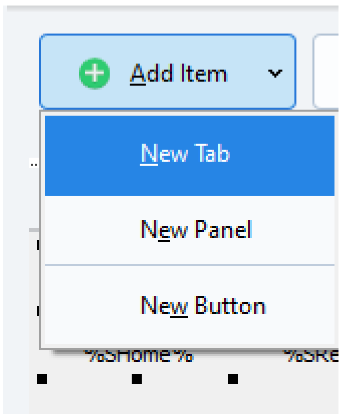
Visual Editor

The Navigation Bar editor is similar to Embarcadero Delphi and C++ Builder Form Designer. On the top side, you have **different tools: add item, remove, copy/cut/paste**. On the left bottom side, you can **edit properties** of the selected control(s) with the Properties Editor. On the bottom right side, you can **associate actions to the selected control**.

Each control has its own unique name.

Recommended: Learn how to use the Navigation Bar Editor with our tutorial

Add a new control



Choose the type of control you want and place it onto the form. You can resize the control with the grips. Note that the Tab and Panel controls are containers and can own other controls called children.

The best way to create new controls is to copy and paste existing ones. To quickly modify an existing UI model, use the **Copy/Cut/Paste buttons**:

Edit Properties

The Properties Editor shows all properties of a control. It works the same way as the Properties editor described here. You don't have to modify all properties. Only some of them are useful such as Caption, Align, Name, Enabled, Visible...

Align controls for resizing

To create UI controls that resize properly, use the following control properties (see more info):

Align
AlignWithMargins
Anchors
Margins
Padding

Remove a control

Select the control(s) you want to delete and click the Delete button (or press the DEL key).

Import/Export a model

Several ready-to-use (and ready to modify) models are shipped with HTML Executable. You'll find them in the **`UIModels`** subfolder of your HTML Executable location. Model files are given the `.heuim` extension. You can export your own models too.

Control Actions

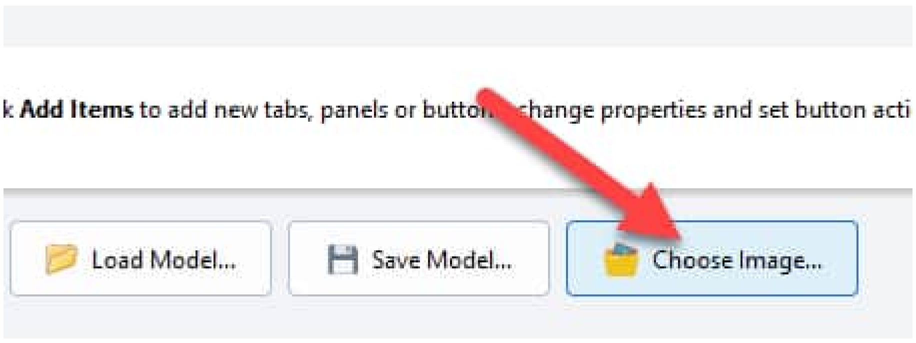
When you place buttons on the UI form, you must associate an action that will be performed when the user clicks the button.

Remember to click `Apply changes` after setting an action to save the changes.

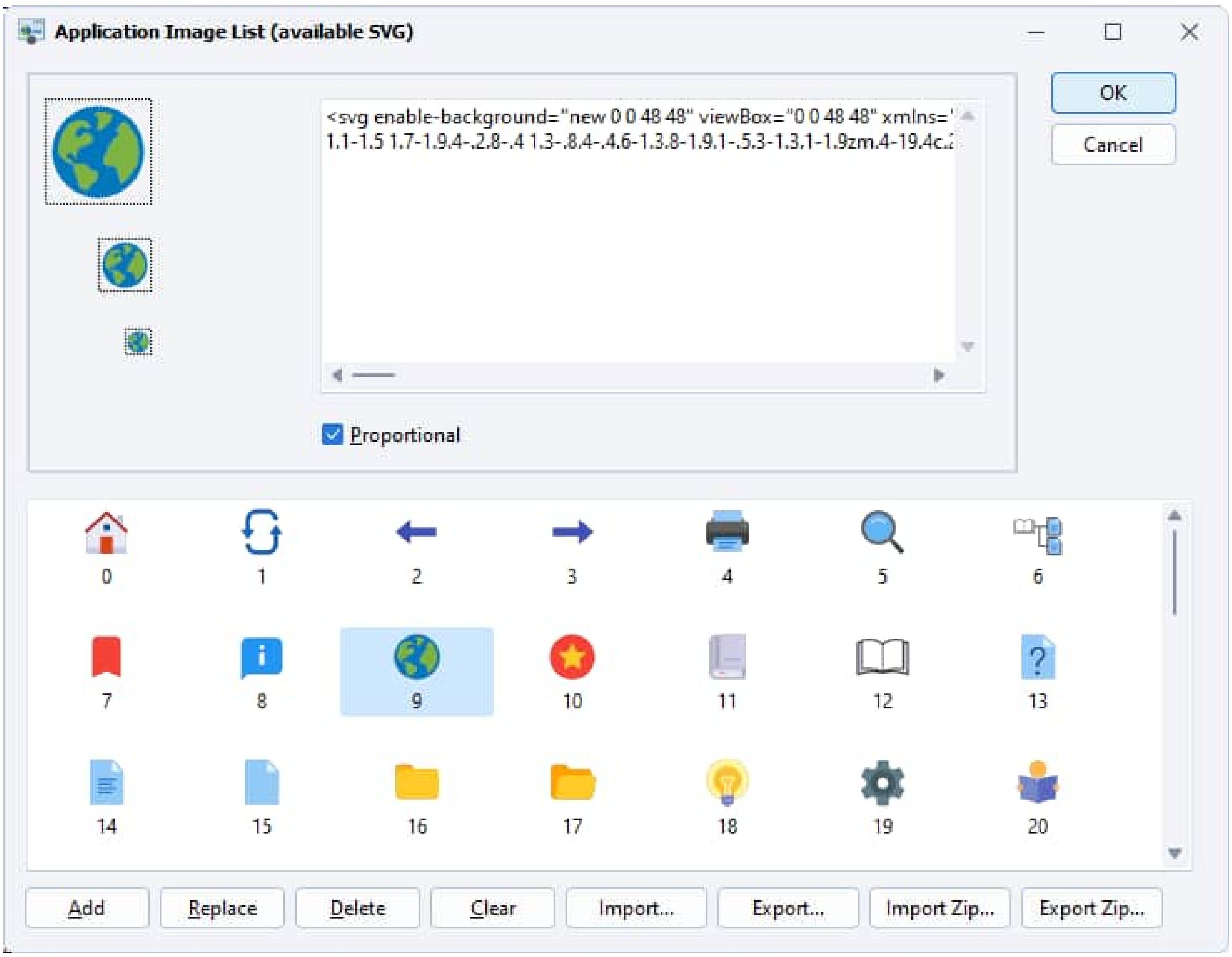
[See all available control actions]

Choose an image for a button

Button images are stored in svg format in a list common to your point project. You can manage this list using the SVG image manager accessible by clicking on **Choose image**. The SVG Image Manager will be shown. To select the image displayed on a currently edited button, you need to set the image index property to the number corresponding to the image in the list. However, for greater convenience, it's easier to click on the Choose image button, which allows you to visually select the image directly without having to enter a number.



k Add Items to add new tabs, panels or button change properties and set button acti
Load Model...
Save Model...
Choose Image...



```
<svg enable-background="new 0 0 48 48" viewBox="0 0 48 48" xmlns='
1.1-1.5 1.7-1.9.4-.2.8-.4 1.3-.8.4-.4.6-1.3.8-1.9.1-.5.3-1.3.1-1.9zm.4-19.4c.
```

Modify control at runtime

It's possible to change the control's properties at runtime.

Refer to the dedicated topic "How To Modify Controls At Runtime".

3.4.3.1. Tutorial: how to make a ribbon or a navigation toolbar

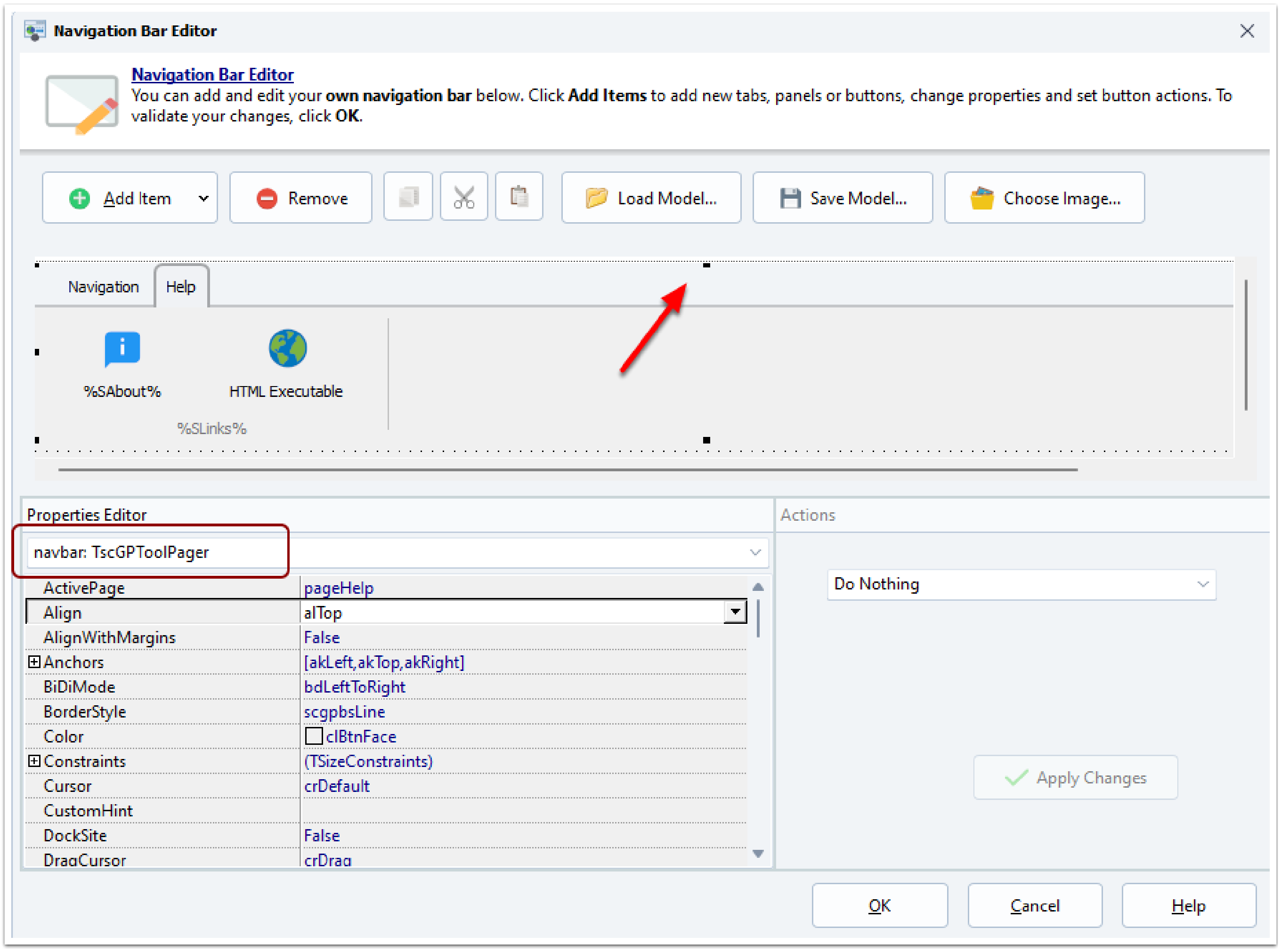
This tutorial introduces you to the concept of creating a ribbon using HTML Executable's Navigation Bar Editor. If you are familiar with Microsoft Office, you would already have a fair understanding of what a ribbon is. In the context of Microsoft Office, a ribbon is a tool that houses the command buttons and icons in a series of tabs at the top of a window. It provides a rich command presentation and replaces traditional menus with tabs and buttons.

In HTML Executable, you can create a similar structure for your ebook or HTML application using the Navigation Bar Editor. This tool allows you to organize your user interface into different tabs, each containing panels and buttons for various commands. This structure can greatly enhance the user experience, making it easy to find and use different features within your ebook or publication.

In the coming sections, we will delve deeper into the process of creating a ribbon using HTML Executable's Navigation Bar Editor. We'll cover the basics of setting up tabs, adding panels, and populating those panels with buttons for various commands.

How To Manage Tabs

To manage the different tabs in your ribbon, select the ribbon in the editor by clicking on it or choose the "navbar" control in the object list of the Properties editor


Navigation Bar Editor

Navigation Bar Editor
You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set button actions. To validate your changes, click **OK**.

Properties Editor

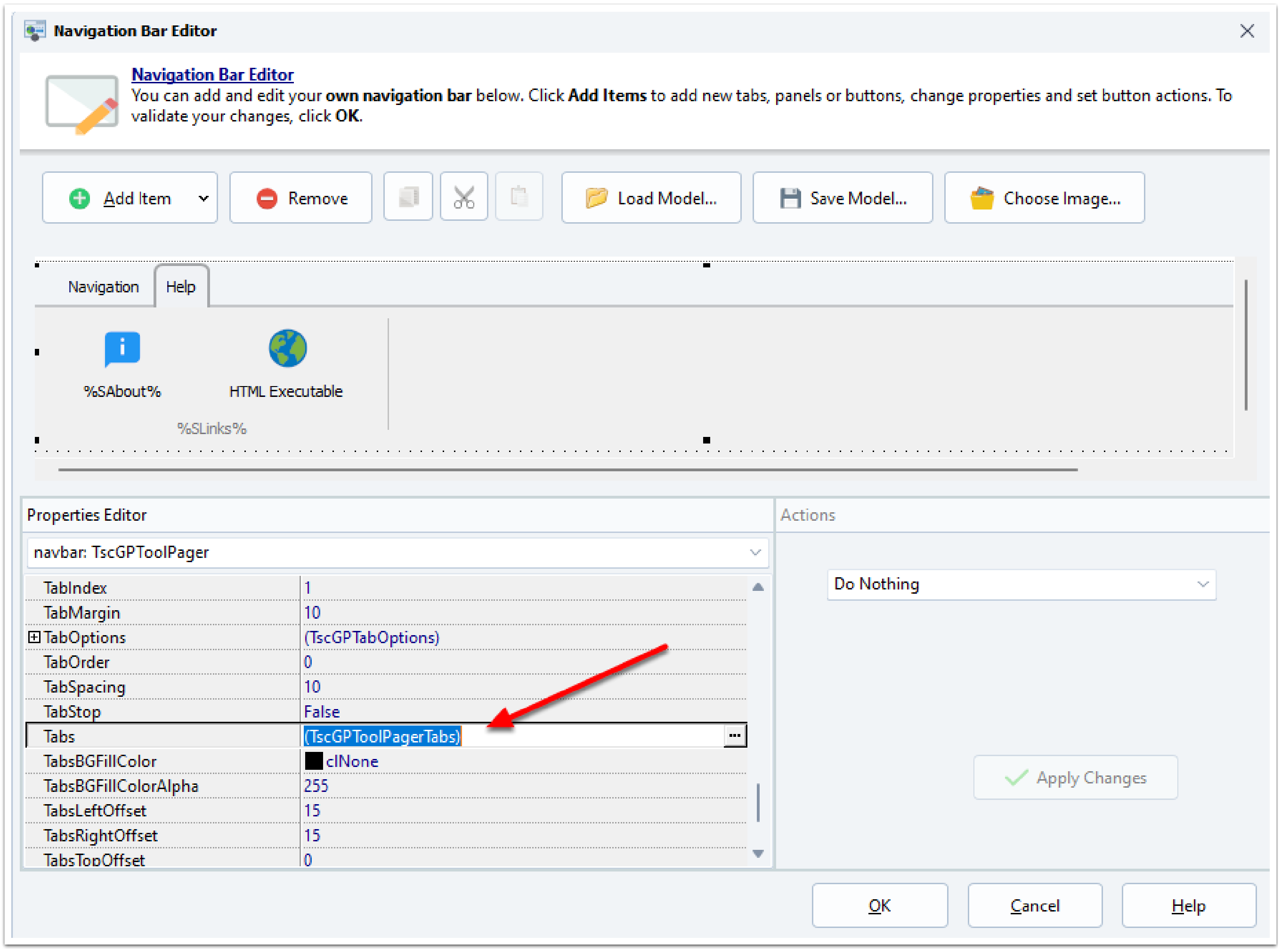
navbar: TscGPToolPager

ActivePage	pageHelp
Align	alTop
AlignWithMargins	False
⊞Anchors	[akLeft,akTop,akRight]
BiDiMode	bdLeftToRight
BorderStyle	scgpbsLine
Color	☐clBtnFace
⊞Constraints	(TSizeConstraints)
Cursor	crDefault
CustomHint	
DockSite	False
DragCursor	crDrag

Actions

Do Nothing

Double-click on the "Tabs" property or click the '...' button

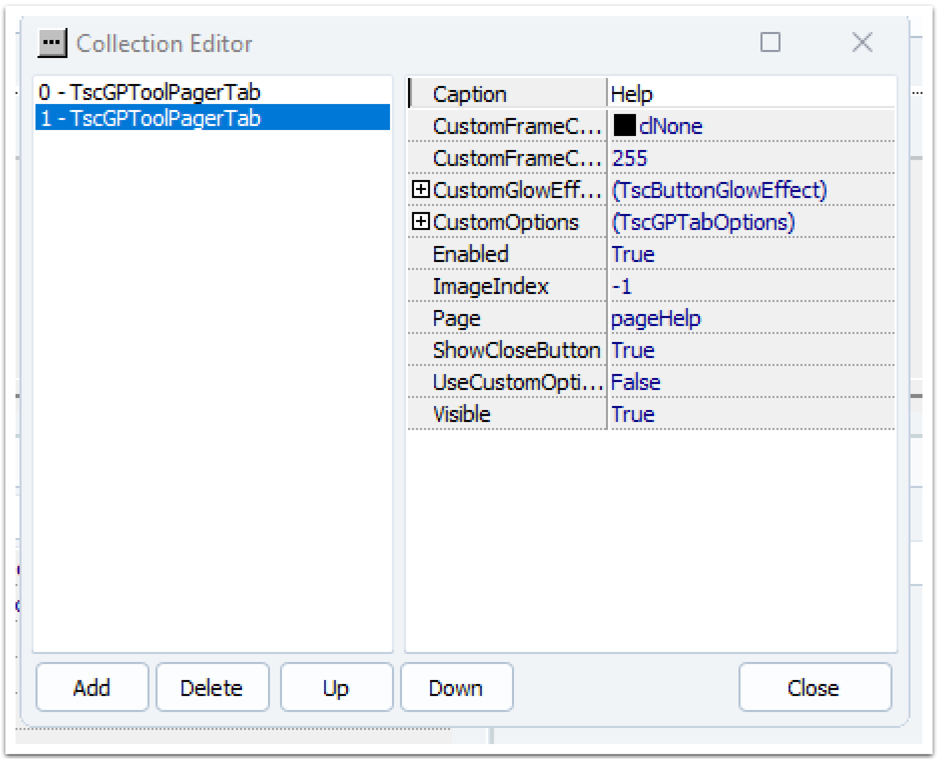


Navigation Bar Editor
You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set button actions. To validate your changes, click **OK**.

navbar: TscGPToolPager

TabIndex	1
TabMargin	10
⊞TabOptions	(TscGPTabOptions)
TabOrder	0
TabSpacing	10
TabStop	False
Tabs	(TscGPToolPagerTabs)
TabsBGFillColor	■ clNone
TabsBGFillColorAlpha	255
TabsLeftOffset	15
TabsRightOffset	15
TabsTopOffset	0

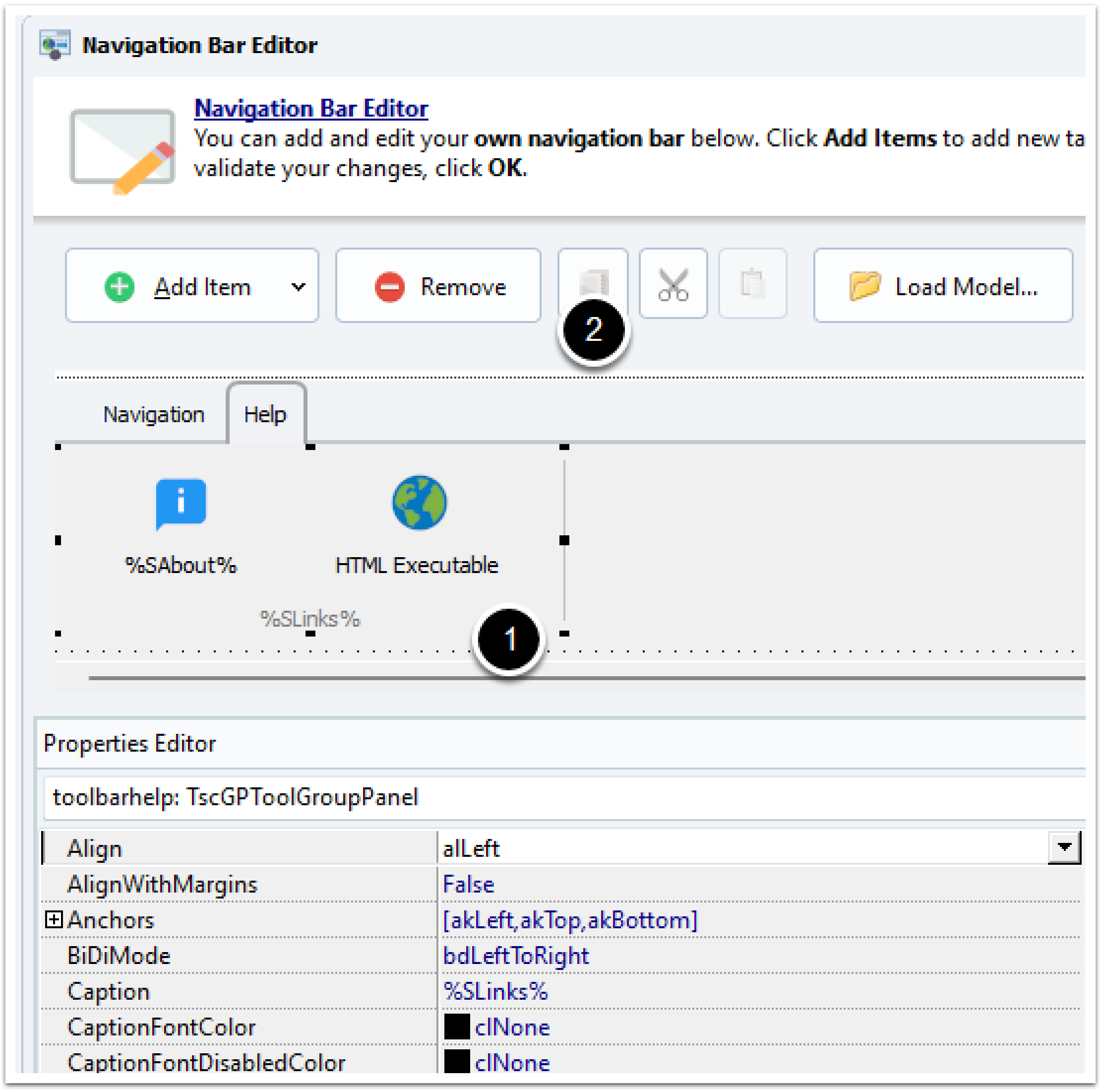
The collection editor appears, allowing you to add, delete or move the various tabs in your ribbon. Use the Caption property to change the title of your tabs.



0 - TscGPToolPagerTab
1 - TscGPToolPagerTab

Caption	Help
CustomFrameC...	■ clNone
CustomFrameC...	255
⊞CustomGlowEff...	(TscButtonGlowEffect)
⊞CustomOptions	(TscGPTabOptions)
Enabled	True
ImageIndex	-1
Page	pageHelp
ShowCloseButton	True
UseCustomOpti...	False
Visible	True

Add a new toolbar to a tab in the ribbon


Navigation Bar Editor

Navigation Bar Editor
You can add and edit your **own navigation bar** below. Click **Add Items** to add new ta
validate your changes, click **OK**.

Properties Editor

toolbarhelp: TscGPToolGroupPanel

Align	alLeft
AlignWithMargins	False
⊞Anchors	[akLeft,akTop,akBottom]
BiDiMode	bdLeftToRight
Caption	%SLinks%
CaptionFontColor	■ clNone
CaptionFontDisabledColor	■ clNone

The easiest way to do this is cloning an existing toolbar. HTML Executable ships with ready-to-import models, just use them as a start.

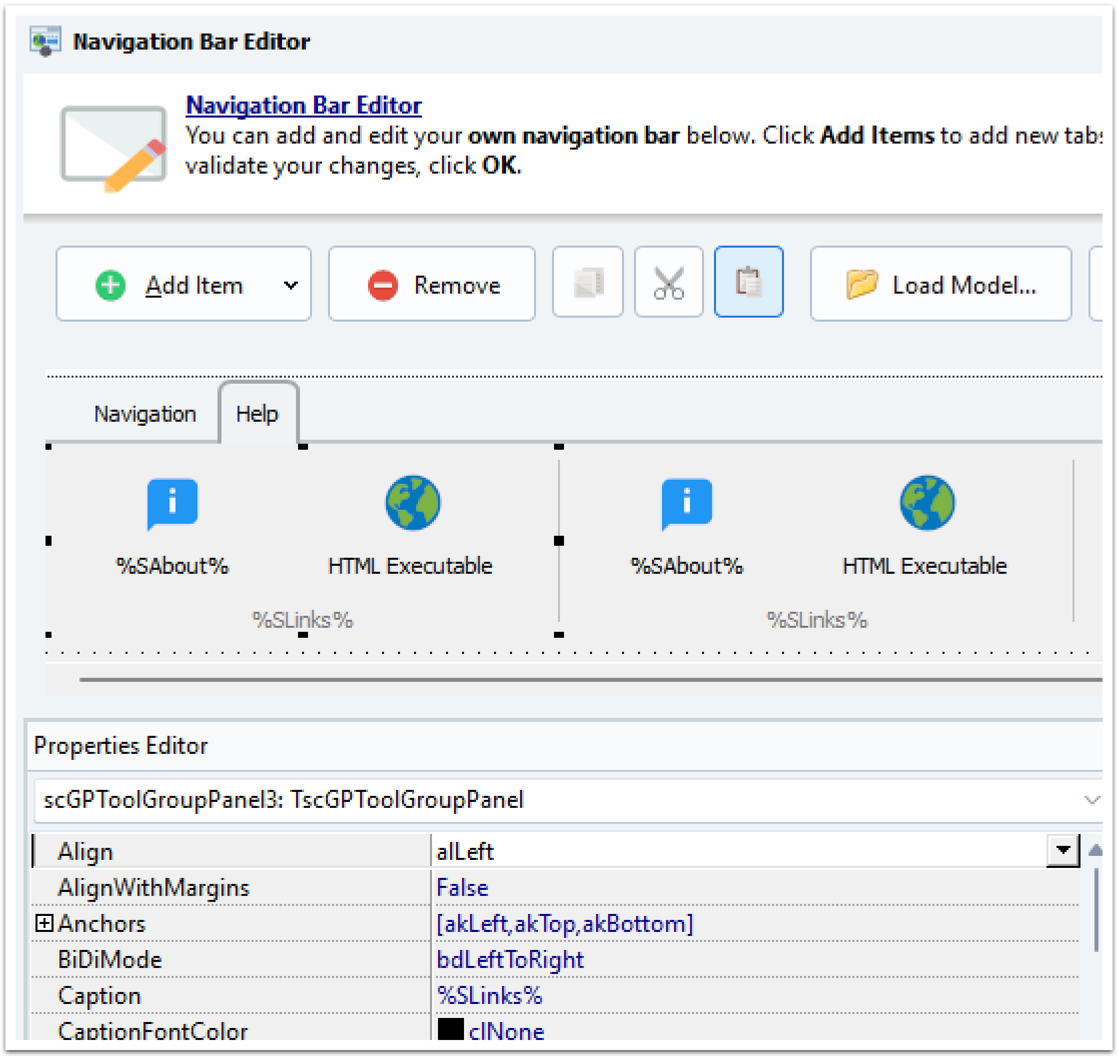
1. Select the toolbar you want to clone

2. Click "Copy"

3. Deselect the control and choose the destination tab where you want to place the new toolbar.

4. Click "Paste"

The new toolbar can be moved but ensure the "Align" property remains to "alLeft".



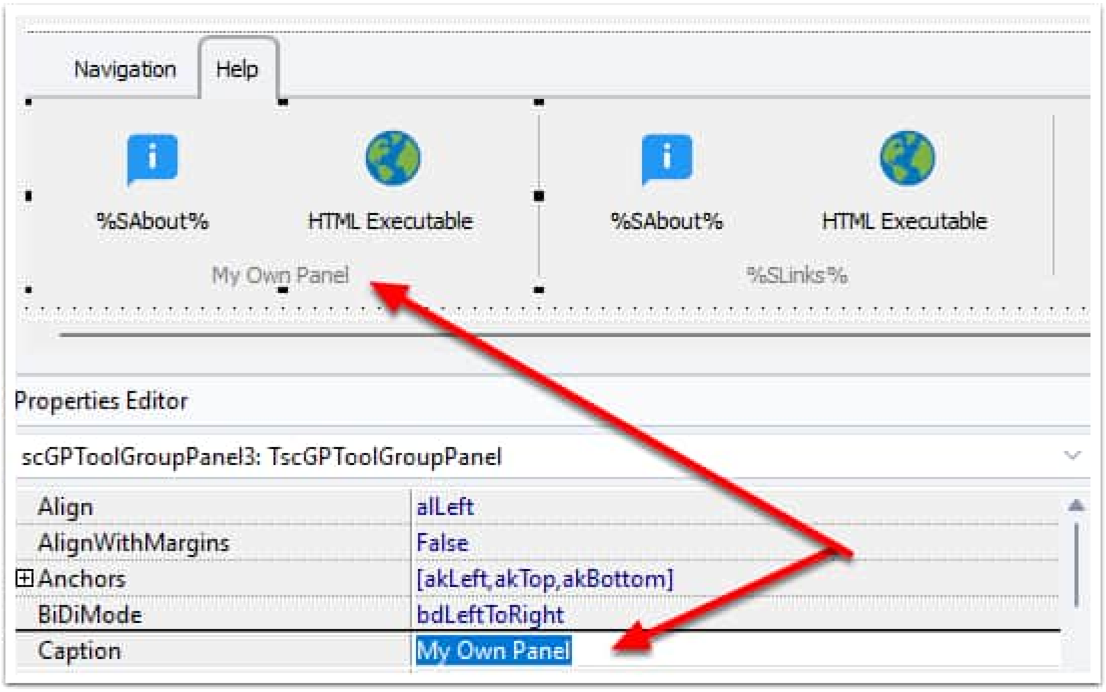
Navigation Bar Editor

You can add and edit your **own navigation bar** below. Click **Add Items** to add new tab
validate your changes, click **OK**.

scGPToolGroupPanel3: TscGPToolGroupPanel

Align	alLeft
AlignWithMargins	False
⊞Anchors	[akLeft,akTop,akBottom]
BiDiMode	bdLeftToRight
Caption	%SLinks%
CaptionFontColor	■ clNone

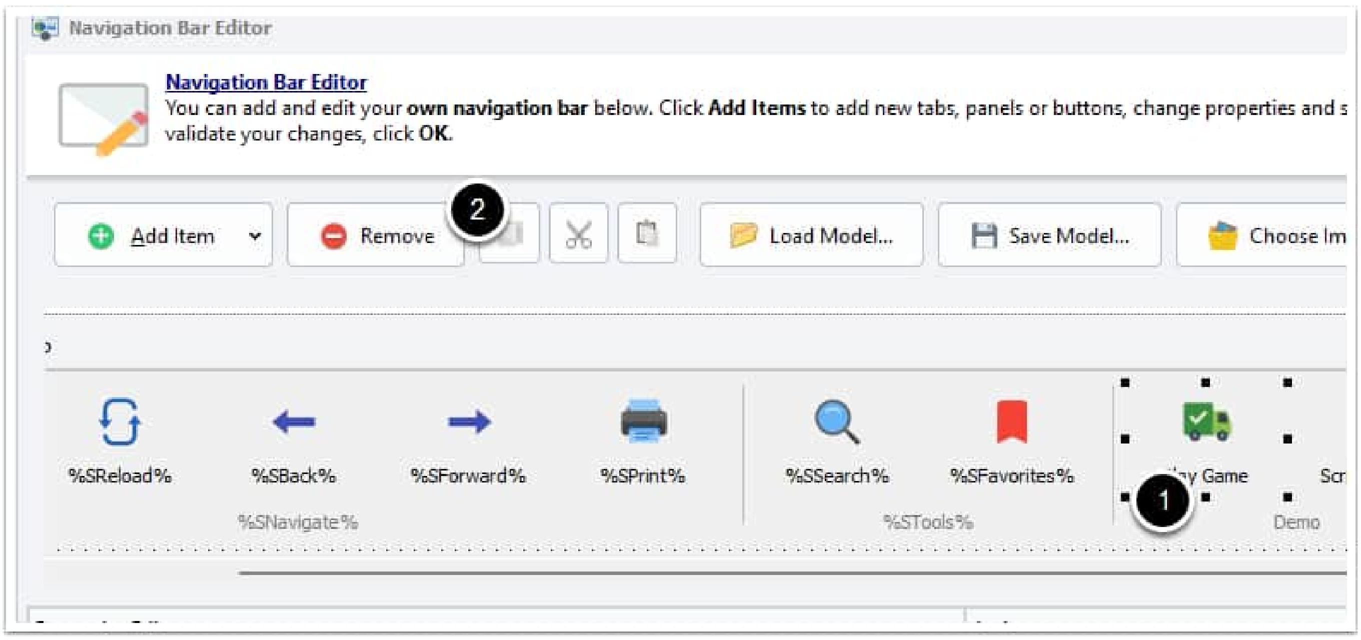
You can also change its title with the "Caption" property.



scGPToolGroupPanel3: TscGPToolGroupPanel

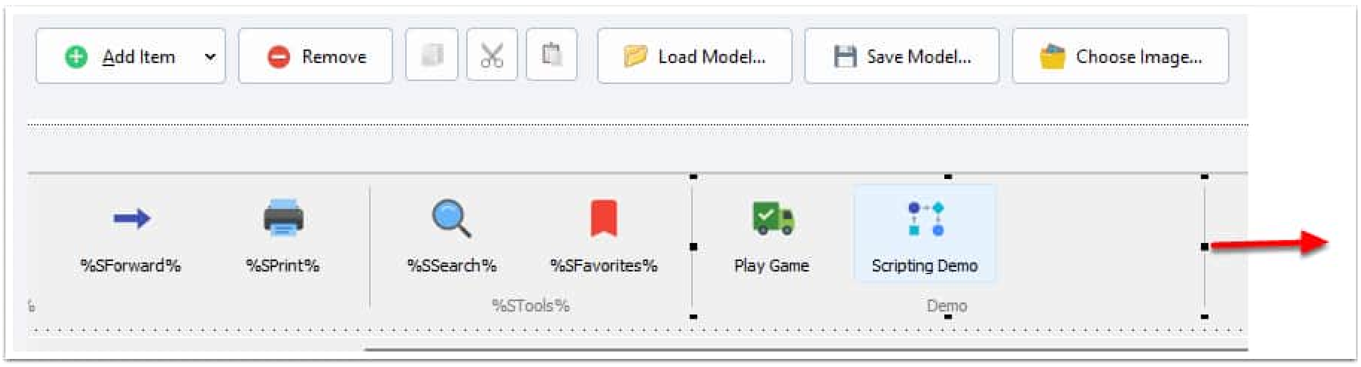
Align	alLeft
AlignWithMargins	False
⊞Anchors	[akLeft,akTop,akBottom]
BiDiMode	bdLeftToRight
Caption	My Own Panel

Add a new button to the toolbar and optionally a separator

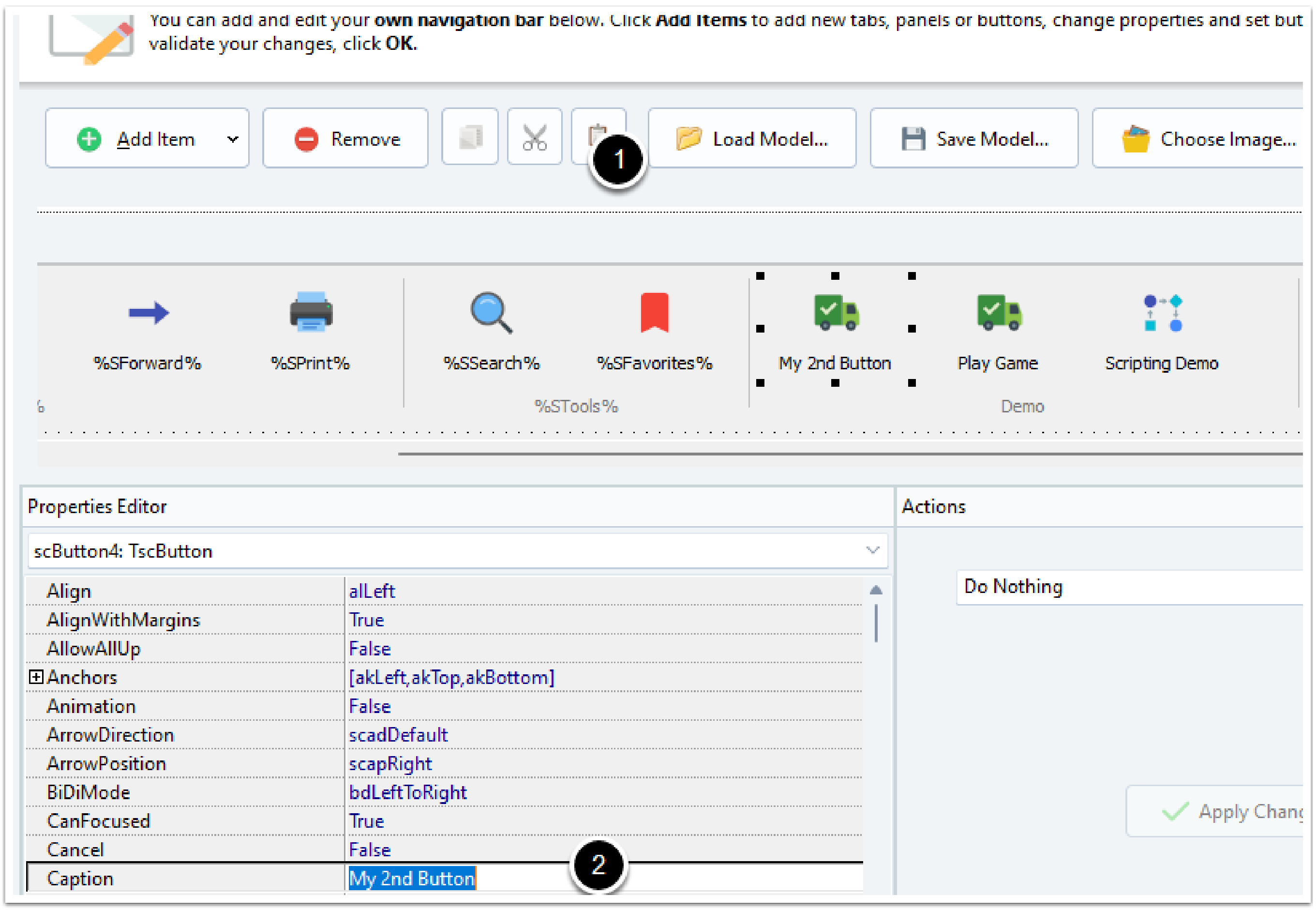


To add a new button to the toolbar, we clone an existing one. Select the original button to clone (1) and click Copy to clipboard (2)

Select the panel where you want to place the new button and increase its width with the selectors:



Click Paste (1) to paste the clone, and you can then edit the button's properties. For instance, its Caption (2):


You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set but
validate your changes, click **OK**.

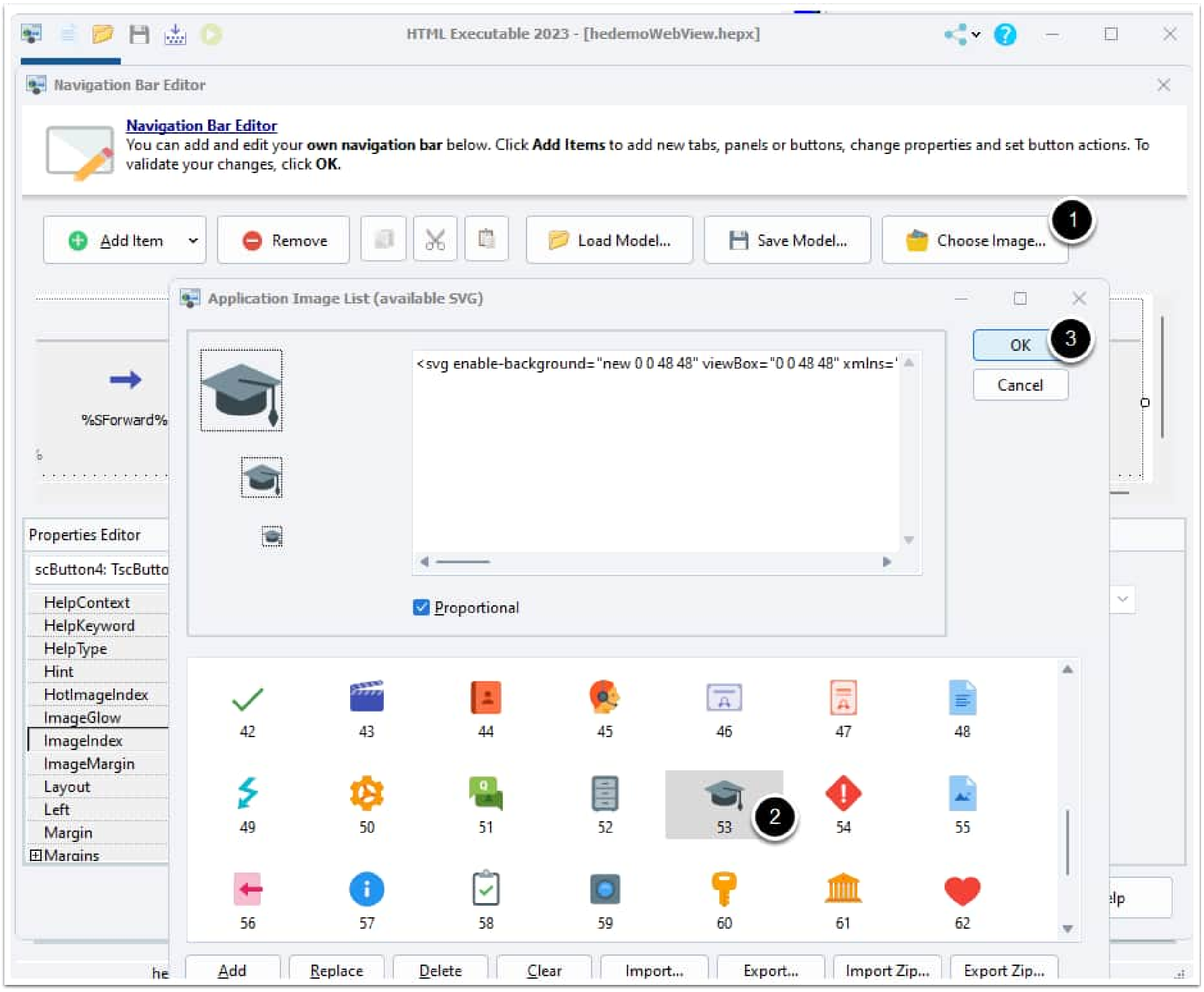
Properties Editor

scButton4: TscButton

Align	alLeft
AlignWithMargins	True
AllowAllUp	False
⊞Anchors	[akLeft,akTop,akBottom]
Animation	False
ArrowDirection	scadDefault
ArrowPosition	scapRight
BiDiMode	bdLeftToRight
CanFocused	True
Cancel	False
Caption	My 2nd Button

Modify the button's icon

To change the button's icon, click "Choose Image" and select the icon (SVG format) you want. You can import new SVG by clicking Add or use existing ones already available in your HTML Executable project.

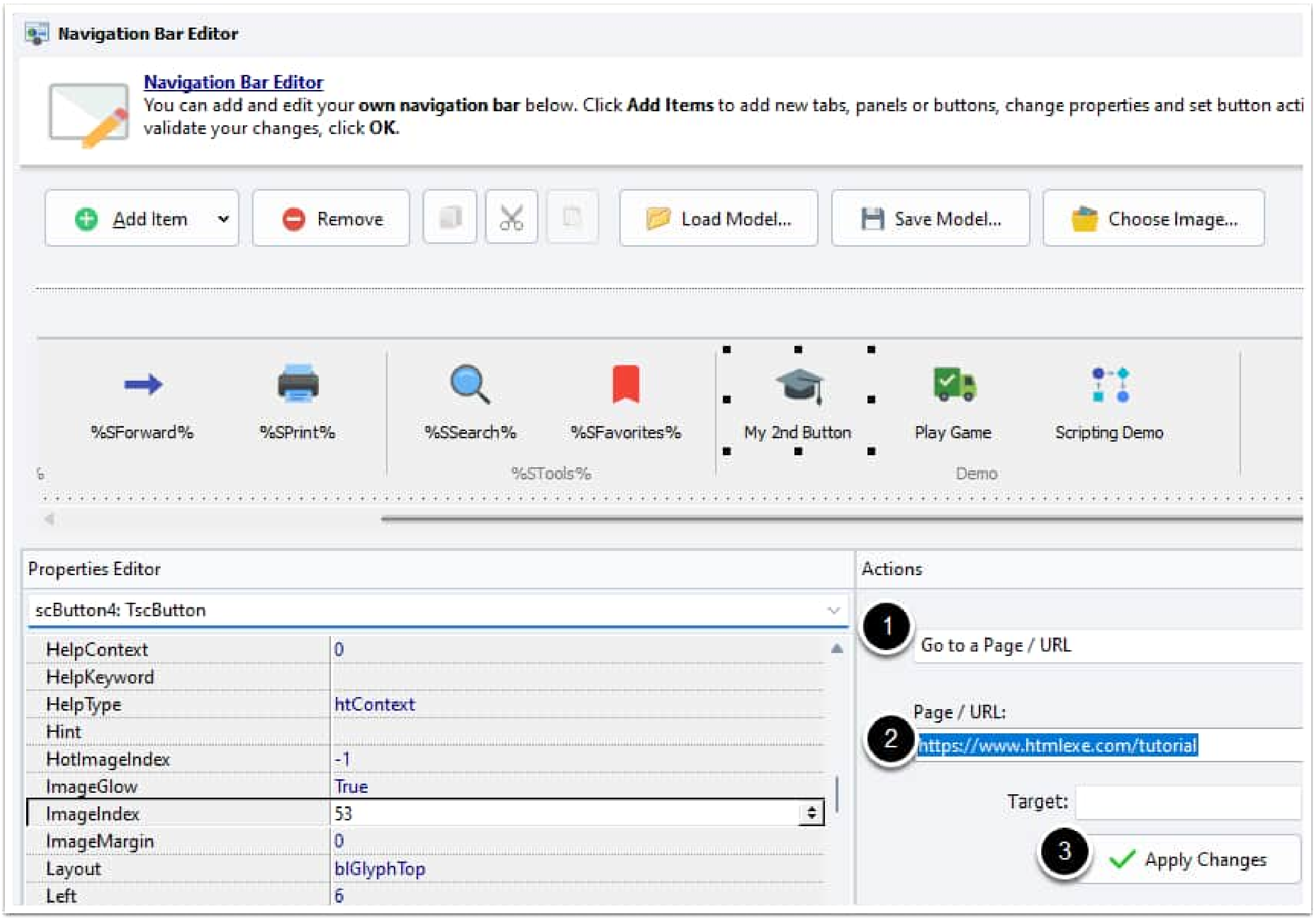


Navigation Bar Editor
You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set button actions. To validate your changes, click **OK**.

`<svg enable-background="new 0 0 48 48" viewBox="0 0 48 48" xmlns='`

Properties Editor
scButton4: TscButto
HelpContext
HelpKeyword
HelpType
Hint
HotImageIndex
ImageGlow
ImageIndex
ImageMargin
Layout
Left
Margin
Margins

You then need to associate an action to be executed when the user clicks the button. To do this, select the action to be carried out (1), then enter any parameters required for the action, such as which page to go to (2), and don't forget to click Apply Changes to save (3).



Navigation Bar Editor

Navigation Bar Editor

You can add and edit your **own navigation bar** below. Click **Add Items** to add new tabs, panels or buttons, change properties and set button acti
validate your changes, click **OK**.

Properties Editor

scButton4: TscButton

HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
HotImageIndex	-1
ImageGlow	True
ImageIndex	53
ImageMargin	0
Layout	blGlyphTop
Left	6

Actions

Go to a Page / URL

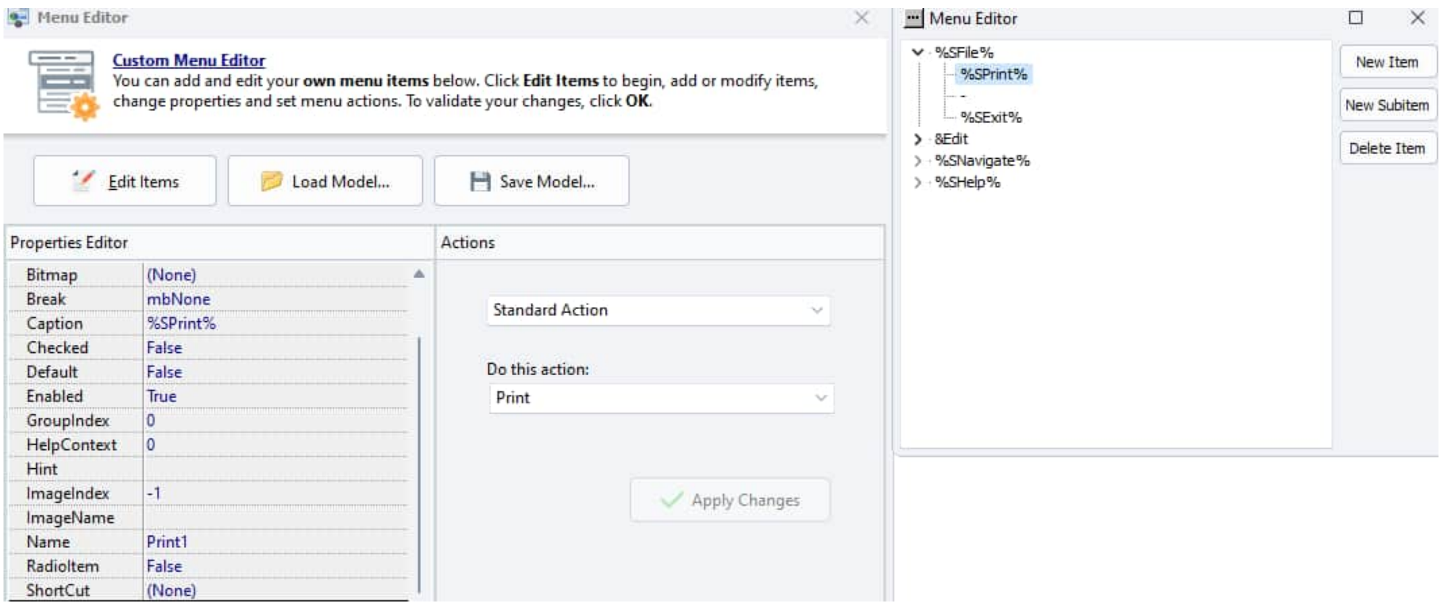
Page / URL: https://www.htmlexe.com/tutorial

Target:

Apply Changes

3.4.4. Menu and Button Actions

In the menu editor and/or button editor, when you create menu items, buttons, you must associate an action that will be performed when the user clicks the button/menu item:



Menu Editor

Custom Menu Editor

You can add and edit your **own menu items** below. Click **Edit Items** to begin, add or modify items, change properties and set menu actions. To validate your changes, click **OK**.

Properties Editor

Bitmap	(None)
Break	mbNone
Caption	%SPrint%
Checked	False
Default	False
Enabled	True
GroupIndex	0
HelpContext	0
Hint	
ImageIndex	-1
ImageName	
Name	Print1
RadioItem	False
ShortCut	(None)

Actions

Standard Action

Do this action: Print

Apply Changes

How to define an action

Select the component (button, image or menu item) to make the Actions panel active.
Choose the action you want to be executed: `Standard Action`, `Go to a page/url`, `Execute an HEScript Function`, and finally `Execute JavaScript Code`.
Be sure to click `Apply Changes` to save your settings before selecting another control or exiting the editor.

Available Actions

Standard Action

Lets you choose a pre-defined action in the list:

Go Home
Back
Forward
Refresh
Print
Find
Select All
Copy
Cut
Paste
Zoom In
Zoom Out
Reset Zoom
About
Exit Application
Print to PDF
Show/Hide Favorites
Show/Hide TOC
Show/Hide Search
Go to web homepage
Remember last viewed page (special)
Unlock this application
Deactivate this application
Check for updates

Go to a page/url

Lets you open the page or the URL specified in the application's browser. You can optionally indicate a target: `_heexternal` (open the URL in the external default web browser) or `_henewinstance` (start a new instance of the application and open the URL). Finally, if you use frames, the destination frame's name can be indicated.

This is not the way to open a popup. See below.

Do not enter something else for frame if there is no frame in your current webpage. Otherwise, nothing will happen.

Execute an HEScript Function

Lets you associate an HEScript script procedure or function to the button/menu item/timer.

Syntax: `[scriptname].[functionname]`

Execute JavaScript Code

This will run the specified JavaScript code. It must be real JS code and not just the name of a JS function. The code is executed in the context of the current HTML page.

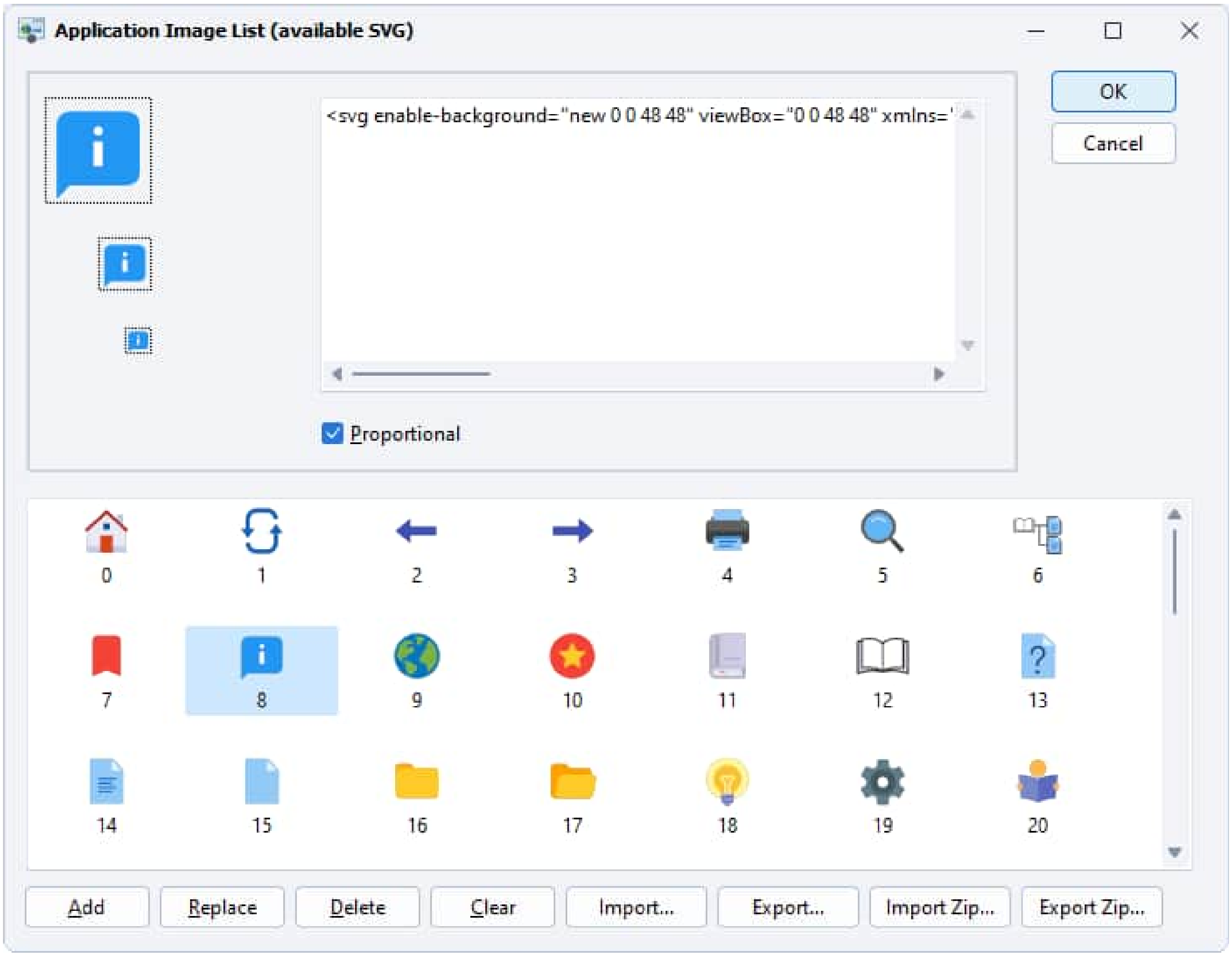
Open a page in a popup

To open a new window or popup, JavaScript code must be used. Choose **Execute JavaScript Code** and type the following code for instance:

```
window.open('https://www.htmlexe.com','preview','width=1000,height=700');
```

3.4.5. Manage SVG Images

The HTML Executable SVG Image Manager is a versatile tool for managing all SVG images and icons that can be used in different parts of a compiled HTML application or ebook. This includes, but is not limited to, buttons on ribbons or toolbar icons (Navigation Bar) or items in the Table of Contents. You can access it thanks to the **Manage SVG Images** button in Application Settings => Components.



Why Use SVG Format?

Scalable Vector Graphics (SVG) is a widely adopted image format due to its versatility and scalability. SVG images can be automatically resized by the HTML Executable application or ebook, according to the user's screen resolution. This flexibility enhances user experience as the images maintain their high-quality details, regardless of the screen size or resolution.

Managing SVG Images

The HTML Executable SVG Image Manager provides a straightforward interface for adding, removing, or replacing SVG images. Here's how:

Adding Images: Use the provided "Add" button to include a new SVG image in your project. Navigate to the image location on your device, select it, and confirm the addition.
Removing Images: Select the image you want to remove from the list, then click the "Delete" button. Clear will empty the entire list.
Replacing Images: To replace an image, first select it from the list, then click the "Replace" button. Navigate to the new image location on your device, select it, and confirm the replacement.

Each SVG image in the manager is identified by a unique number in the list, which is used by certain interface controls to associate the corresponding image.

Importing and Exporting SVG Images

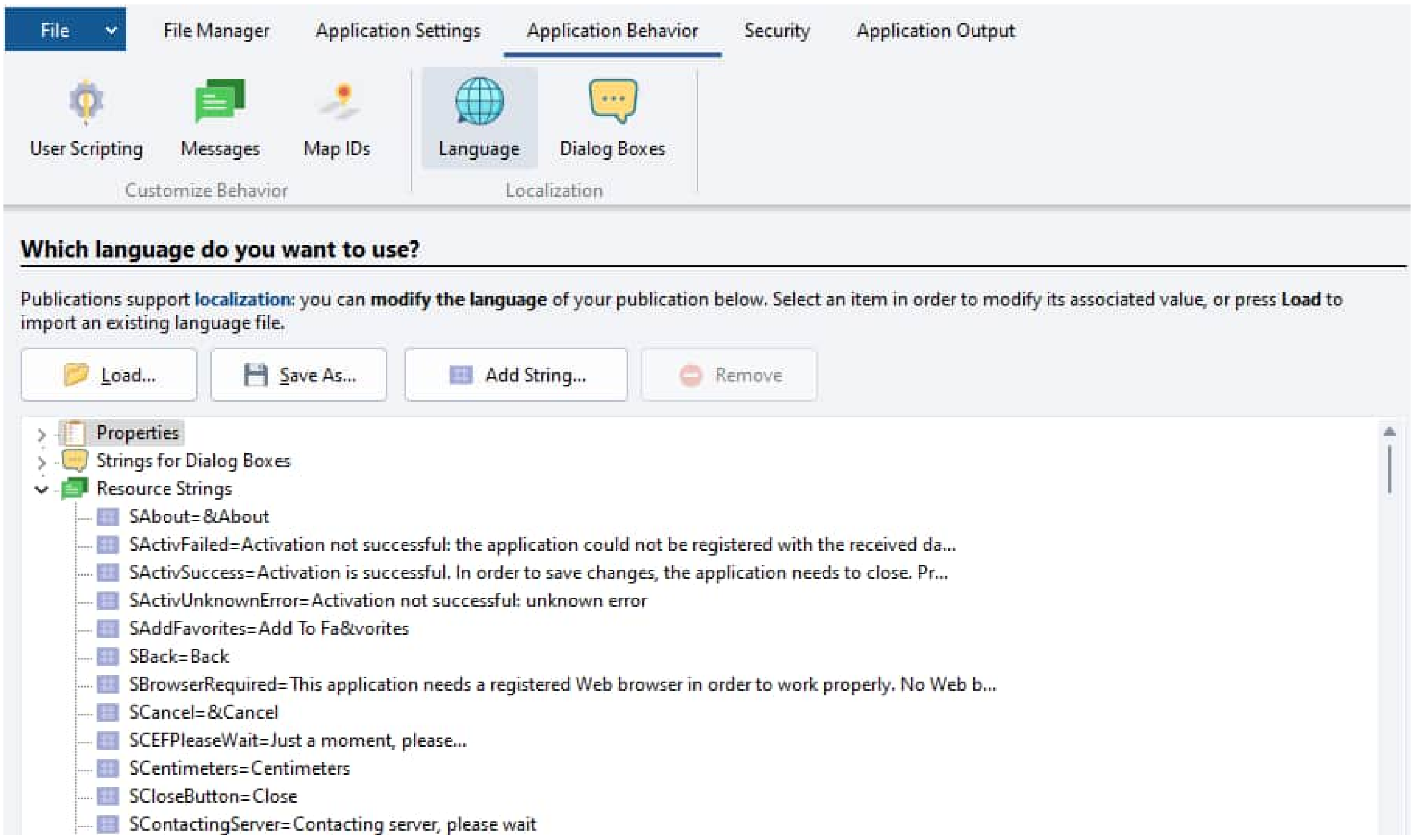
The HTML Executable SVG Image Manager allows you to import and export all project images in ZIP files. This feature facilitates the sharing and reusing of icons among different projects.

In addition, you can save the SVG image library in HTML Executable's proprietary format with the .HEIMG extension. By default, the icon library can be found in the software's installation subdirectory: `C:\Program Files (x86)\HTML Executable 2023\Resources\SVG`. Those who wish to customize the images that are loaded by default in a new project can modify this file accordingly.

3.5. Localization of Ebooks and Publications

HTML Executable provides a powerful feature that allows you to **localize your ebook, making it accessible to a wider audience**. Here's a comprehensive guide on how to do it.

All the localization tools are available on the Language page in HTML Executable:



Which language do you want to use?

Publications support **localization**: you can **modify the language** of your publication below. Select an item in order to modify its associated value, or press **Load** to import an existing language file.

- Properties
- Strings for Dialog Boxes
- Resource Strings
  - SAbout=&About
  - SActivFailed=Activation not successful: the application could not be registered with the received da...
  - SActivSuccess=Activation is successful. In order to save changes, the application needs to close. Pr...
  - SActivUnknownError=Activation not successful: unknown error
  - SAddFavorites=Add To Fa&vorites
  - SBack=Back
  - SBrowserRequired=This application needs a registered Web browser in order to work properly. No Web b...
  - SCancel=&Cancel
  - SCEFPleaseWait=Just a moment, please...
  - SCentimeters=Centimeters
  - SCloseButton=Close
  - SContactingServer=Contacting server, please wait

Understanding Resource Strings

Resource strings are the text elements used internally by HTML Executable's runtime. These include menu items, dialog box text, error messages, and more.
By modifying these resource strings, you can customize the text displayed in your ebook.

Editing Resource Strings

To edit resource strings, simply select them from the list and an editor will appear. Modify the contents of the

Value field, then press Apply to save the new value of the resource string.

To localize your ebook, simply replace the values of all resource strings with the translated text. For example, if you're localizing your ebook to French, you might replace SAbout's value with "À propos".

Importing Pre-made Language Files

HTML Executable also allows you to **import pre-made language files**. These files contain translations for all the resource strings used by HTML Executable.

📢 Currently, translations are available in English, German, French, Italian, and Spanish.

To import a pre-made language file, go to the "Application Settings" => "Language" section and click the "**Load**" button. Then, select the language file you want to import.

ℹ You'll find existing language files at
C:\Program Files (x86)\HTML Executable 2023\Languages

Compiling Your Ebook

Once you've edited the resource strings or imported a pre-made language file, you can compile your ebook. HTML Executable will use the strings from the language file, effectively localizing your ebook.

And that's it! You've successfully localized your ebook with HTML Executable. Now, readers from different regions can enjoy your ebook in their native language 🌐.

3.6. Streaming Audio and Video in HTML Executable Ebooks

HTML Executable applications and ebooks can efficiently manage the **streaming of audio and video content directly within web pages,** utilizing HTML5 `video` and `audio` tags. This offers a seamless multimedia experience for the user, with the support of various formats to ensure broad compatibility.

Let's dive in to explore how you can enrich your HTML Executable applications with multimedia features and also protect your content from unwanted copying.

Streaming Videos and Audios

Consider the following HTML code that plays a video:

```
<video id="video" loop autoplay controls>
  <source src="/assets/sunpeek.webm"></source>
</video>
```

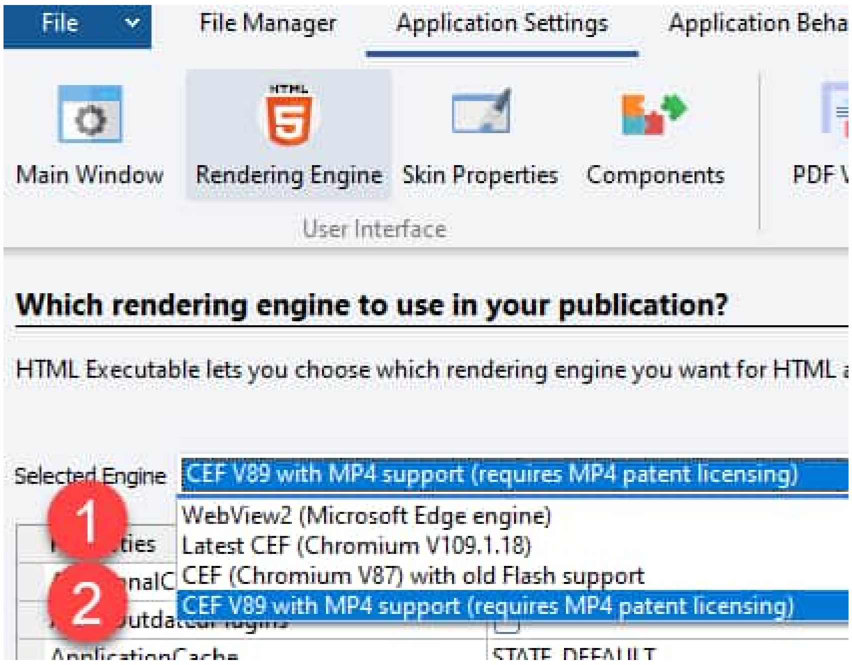
This code will automatically play a looping video with controls for the user to interact with. Autoplay is supported, providing an uninterrupted multimedia experience.

For media files that are not overly large, typically less than 5 MB, you can have HTML Executable compile them directly into the EXE file. However, for **very large video files**, we recommend you to store them externally in the same folder as the EXE. HTML Executable will automatically search for them if they aren't compiled into the executable file.

CEF restriction for open-source video and audio codecs only.

As explained in the topic dedicated to HTML renderers, if you choose CEF as your HTML rendering engine for your ebook, some restrictions apply by default: CEF does not include proprietary media codecs (MP4 videos won't be played unless you use the dedicated CEF build that offers MP4 support but requires specific licensing).

Thus, only **open-source or non-proprietary audio and video formats (and codecs)** are supported: WEBM, WEBA, OGG, and MP3. Proprietary formats and codecs like MP4 and AAC are not available in Chromium builds by default, due to licensing requirements. To support these formats, you'll need to rely on our special CEF builds with MP4 support (2), or you can **switch to the WebView2 engine (1)**.



Which rendering engine to use in your publication?

HTML Executable lets you choose which rendering engine you want for HTML a

Selected Engine: CEF V89 with MP4 support (requires MP4 patent licensing)

- WebView2 (Microsoft Edge engine)
- Latest CEF (Chromium V109.1.18)
- CEF (Chromium V87) with old Flash support
- CEF V89 with MP4 support (requires MP4 patent licensing)

We can provide our customers with specific CEF builds that handle MP4 video formats. Please contact us, expressing your intent to respect the MP4 patent licensing, to access these files.

Note: Free converter and encoding software programs are available for re-encoding your media files into open-source formats.

Handling and Protecting Large Video Files

When dealing with large video files, we recommend keeping them external to the publication .EXE file as Windows only supports EXE file sizes up to 4 GB. These are considered as external files.

Additionally, HTML Executable enables you to **encrypt your video files, limiting their access to only your publication or ebook. This security feature aids in preventing unwanted copying of your video resources.**

For more information on how to encrypt your files, please refer to the Encrypt External Files topic.

In conclusion, HTML Executable provides a comprehensive solution for integrating and securing multimedia content in your applications. With careful management and the appropriate use of features, you can enhance the user experience and maintain control over your digital resources.

3.7. Security and Trial Options For Ebooks and Apps

HTML Executable is an unmatched tool for providing **powerful security features for your ebooks and digital publications**. It offers an array of options that can be adapted to fit your specific needs, enhancing the protection of your digital content. These features can even be combined for superior control and security.

Consider viewing our video tutorial on how to convert an ebook into an evaluation version and understand how to select which pages should be accessible in this trial. For instance, if you aim to offer a trial or evaluation version of your ebook on your website for public download, you might want to lock certain crucial pages. You can freely offer the first 3 chapters to your end users, and when they purchase your ebook, send them a key to unlock the remaining chapters.

Here are the different options you have with HTML Executable to enhance your ebook's security:

Single Computer Accessibility: If you want your ebook to be accessible on only one computer, preventing sharing of your product:
- Use hardware-locked keys with the Certificates page

Remote Ebook Management: If you wish to remotely manage your ebook, lock/unlock it, or control the number of activations:
- Use online activation with the Certificates page

Prevent Content Capture: If you want to stop content capture, prevent screenshots, or blacklist screen grabbers:
- Use the ebook Content Protection features of HTML Executable.

Fully-Secured Ebooks: If you aim to create fully-secured ebooks without having to manage anything:
- Benefit from our Protect Ebook .net service

Trial Version Creation: If you want to create a trial version of your ebook:
- Use Certificates to make a Default version and one (or more) Registered version(s)

- Use security profiles to lock some pages or functionalities on these pages

Expiration: If you want your ebook to expire:
- Use Certificates to make the ebook expire after a given number of days or runs

- Use: "Set a global expiration date" in the Global Protection page if you want your ebook to expire on a specific date

Action Restrictions: If you wish to restrict the actions your customers can perform:
- Use security profiles to restrict the actions your customers can perform on specific pages, such as printing, copying, or selecting text

Use "Disable the PRINT SCREEN key" in the Global Protection page

Use "Only one instance of the publication can be run at a time" in the Global Protection page to prevent your customers from running your application multiple times simultaneously

Dongle Requirement: If you want your ebook to be accessible only when a dongle is inserted:
Link the Publication to a Dongle in the Dongle Protection page

Lock an ebook to a USB stick
See how to lock an ebook app to a USB stick (the ebook will not open if the appropriate USB stick is not plugged in).

Password Protection: If you wish to password-protect your ebook: after your customer makes a payment, you send them the password. Wildcards are accepted, and you can even generate passwords in bulk.
Use Global password in the Global Protection page

With HTML Executable, you're in full control of your ebook's security. Make the most of these features to protect your content, manage accessibility, and ensure your ebook's safety in the digital space!

The following topics explain security features in detail:

- Trial and Restricted Versions
- Certificates - Trial Options
  - Certificate Settings
- Activating a certificate
  - Making Registration Keys for Your Ebooks
  - About Hardware-locked Keys for Ebooks
- Online Activation
- Deactivate a Certificate
- Validation of User Licenses
- paypalkeygen

3.7.1. Trial and Restricted Versions

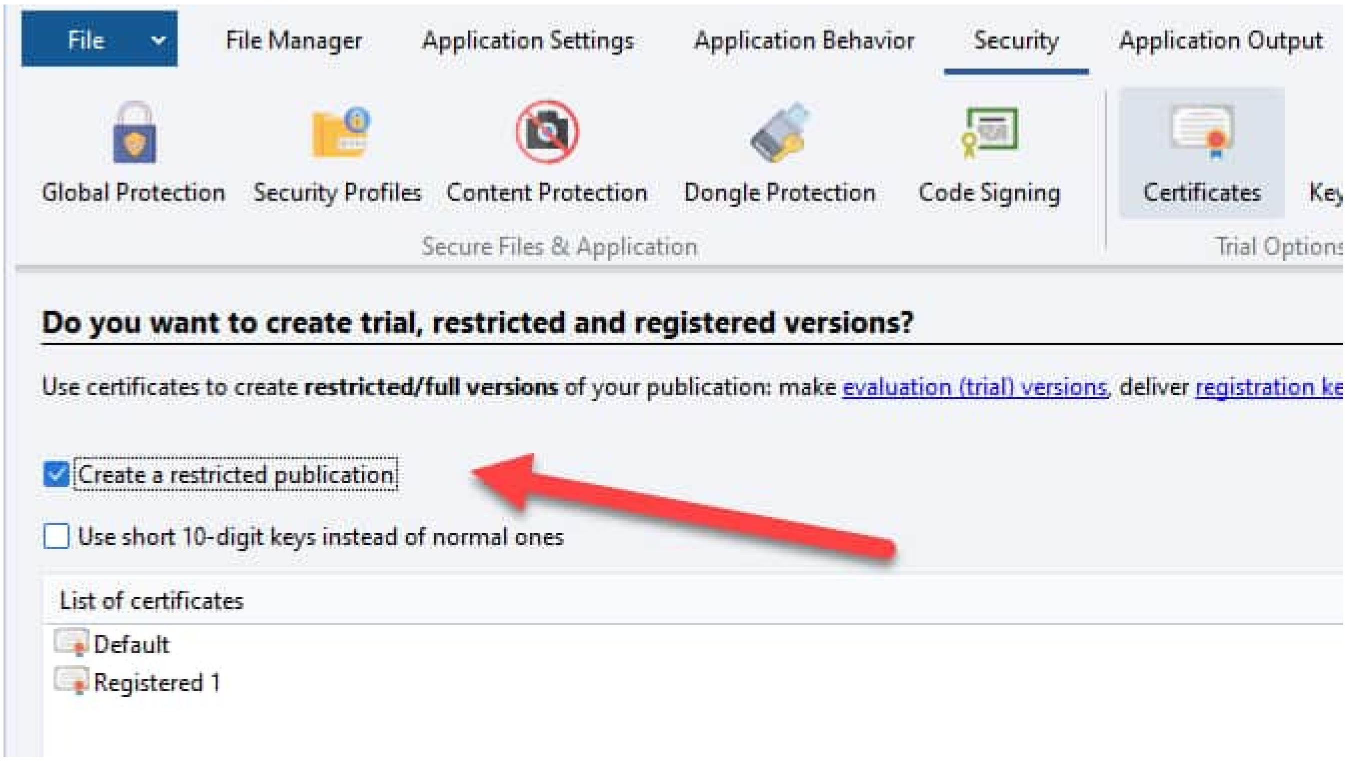
Creating and Managing Trials for Your Ebooks with HTML Executable

As an author or publisher wanting to sell your ebooks, HTML Executable enables you to turn your ebook or HTML application into a demo or trialware easily. This topic will guide you through the process of creating a time-bombed or restricted ebook or publication, which end users can register and activate to gain full access.

Note: If you're unsure about HTML Executable's licensing options, don't hesitate to post your questions or seek help in our User Forum.

Creating a Trial or Restricted Ebook

Navigate to **Security => Certificates** and toggle on **Create a restricted publication**.



Do you want to create trial, restricted and registered versions?

Use certificates to create **restricted/full versions** of your publication: make evaluation (trial) versions, deliver registration ke

☑ Create a restricted publication

☐ Use short 10-digit keys instead of normal ones

List of certificates

Default

Registered 1

Understanding Certificates

Certificates are like doors into your ebook or publication. Only those with the right key, i.e., those who pay for your work, can enter.

Each certificate has a unique Certificate ID, which HTML Executable uses to create associated keys. Remember, your unique signature should remain confidential.

The **Default certificate** is used for evaluation versions of publications. It allows users to access your publication without a key. When this certificate is active, the publication works in the Trial mode.

For registered versions, use other certificates. When a certificate other than the default one is active, the ebook works in the **Registered mode**.

Switching between Default and registered certificates requires end users to activate or deactivate it.

To verify whether an end user still has access to your ebook, use **Validation**.

Trial and Registered Modes

- The About box will display a "Trial: please register this program..." statement in Trial mode, while in Registered mode, it will show "This program is registered to...".
- You can create different registered versions of a single publication, defining which pages are unlocked for each version.
- Some registered versions can also have an expiration date. After this date, your user would need a new key.

Working with Certificates

- HTML Executable automatically adds two certificates when starting a new project. Remember to configure their properties if you plan to create a Trial publication.
- Manage the certificates of your publication or ebook with the Certificate Manager.

- Dive deeper:

  Activating a certificate

  Deactivating a certificate

  Certificate and license validation

  Certificate Properties

  Online Activation

  Hardware-locked Keys: let you prevent end users from sharing their registered ebook with others.

Using Short Keys

By default, HTML Executable generates 21-character keys, like `AFABA-FETR8-54024-20B42`. In some cases, you may prefer shorter, digit-only keys, such as for mobile payments or SMS micropayments.

> **Tip:** If the user enters non-digit characters, they are ignored. Also, certain characters like `O` and `I` are automatically replaced by `0` and `1`.

Protect Ebook .net - Hassle-free Ebook Protection

For our customers, we offer Protect Ebook .net, an affordable, stress-free ebook protection and activation service. Our team manages all technical details for you while you stay focused on your publications and customers.

Take advantage of HTML Executable features like ebook trials and activation, anti-piracy measures, and automated delivery of license keys without the need to master the HTML Executable Activation Kit.

About Protect Ebook .net

Hints for Creating Trial Publications

You may not set up an evaluation period for the Default certificate: in this case, the publication will always offer an evaluation mode unless you enable "Do not allow access to the publication without prior registration" as explained above.

Nevertheless, you can lock some pages (using security profiles) so they may be accessible only upon registration (for instance, an ebook where end users must pay to access to the remaining chapters).

If you do not want to allow an evaluation period at all, once again enable "Do not allow access to the publication without prior registration" as explained above; or alternatively you can just set the number of days to "0" for the Default certificate. In this case, a publication can only work if your user upgrades to another certificate (i.e. enters a key).

You can always lock crucial HTML pages using the Security Profiles or add more restrictions using HEScript scripts.

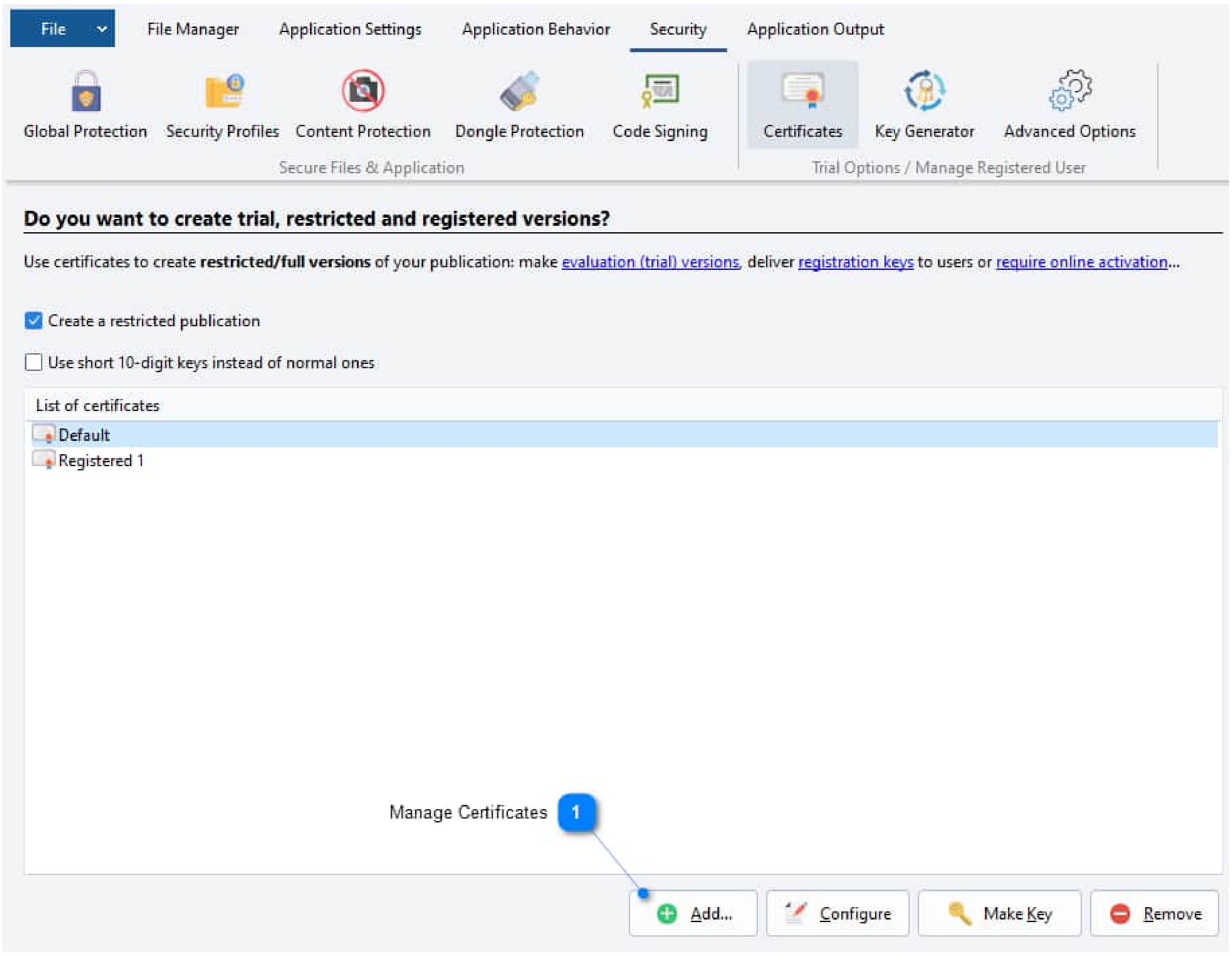
3.7.2. Certificates - Trial Options

What are Certificates

Certificates in HTML Executable are an essential component in the management and distribution of your ebooks and publications. They function as digital doors, enabling access to your work for authorized users, i.e., those who have paid for your content.
Learn more about certificates and what are Default / Registered certificates.

This page explains how to utilize HTML Executable's Certificate Manager to control and manage your certificate list.

You can manage your publication or ebook's certificates on the "Security => Certificates" page:



Do you want to create trial, restricted and registered versions?

Use certificates to create **restricted/full versions** of your publication: make evaluation (trial) versions, deliver registration keys to users or require online activation...

- [✓] Create a restricted publication
- [ ] Use short 10-digit keys instead of normal ones

List of certificates
- Default
- Registered 1

Managing Certificates

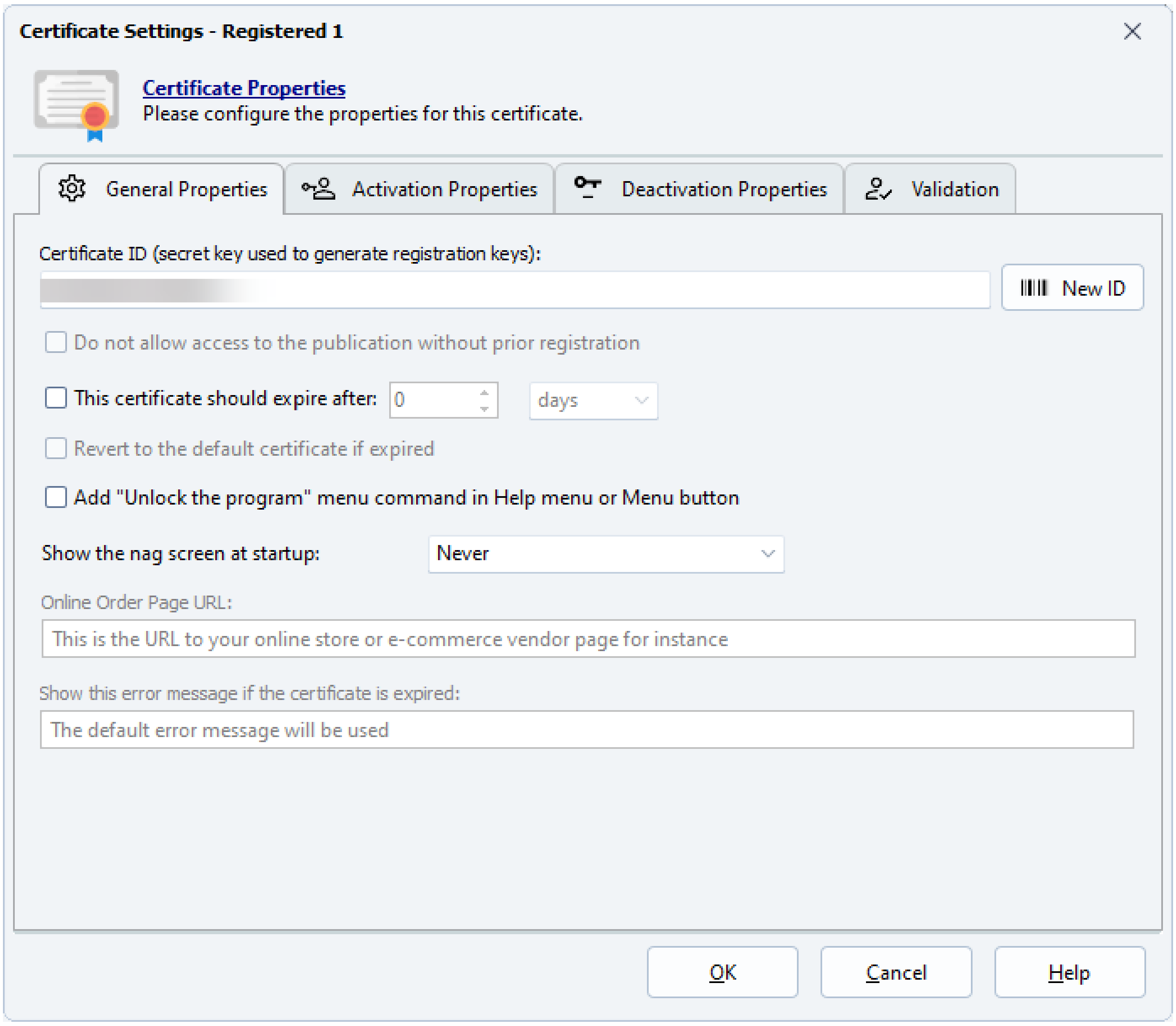
- To add a new certificate, click 'Add' and input the name of your new certificate. Only alphanumeric characters and spaces are permitted.
- To modify the properties of a certificate, either double-click on it or select it and click 'Configure'.
- To delete a certificate, select it and click 'Remove'. Remember that the Default certificate cannot be removed and there should always be at least one Registered certificate.

- To create a key for a certificate, select it and click 'Make Key'. The key generator will be displayed.
- To reset the evaluation period on your own computer, click 'Advanced Options' in the ribbon and then 'Reset'.
- To decide how the unique system ID is created, click 'Advanced Options' and select an option.
- If you wish to generate shorter registration keys, enable the 'Use short 10-digit keys instead of normal ones' option.

Also refer to the 'Certificate Properties' section for more details.

3.7.2.1. Certificate Settings

Every certificate in HTML Executable offers a range of customizable properties. To configure, select the desired certificate from the Certificate Manager and double-click it or use the 'Configure' button.



Certificate ID (secret key used to generate registration keys):

☐ Do not allow access to the publication without prior registration

☐ This certificate should expire after: 0 days

☐ Revert to the default certificate if expired

☐ Add "Unlock the program" menu command in Help menu or Menu button

Show the nag screen at startup: Never

Online Order Page URL:

This is the URL to your online store or e-commerce vendor page for instance

Show this error message if the certificate is expired:

The default error message will be used

✔ **Unique Certificate ID**: Every certificate (excluding the Default) should have a unique Certificate ID. This can be any ID of your choice or you can have HTML Executable generate one by clicking 'New Key'.

⚠ **Warning**: Altering the unique signature will render any older keys, which you've previously created and distributed, void.

✔ **Restricted Access**: Access to your publication can be denied until the user is registered. To enable this, activate "Do not allow access to the publication without prior registration". This option is available only for the Default certificate, and it always displays a nag screen called the "Unlock screen" at startup.

Expiration Features

✔ **Certificate Expiration**: You can set an expiration on certificates, defining the number of days or allowed runs for the evaluation period.

- For the default certificate (for trial purposes), the evaluation period starts with the first run of the publication by the end user.
- For other certificates, the evaluation period begins when the end user enters a key.

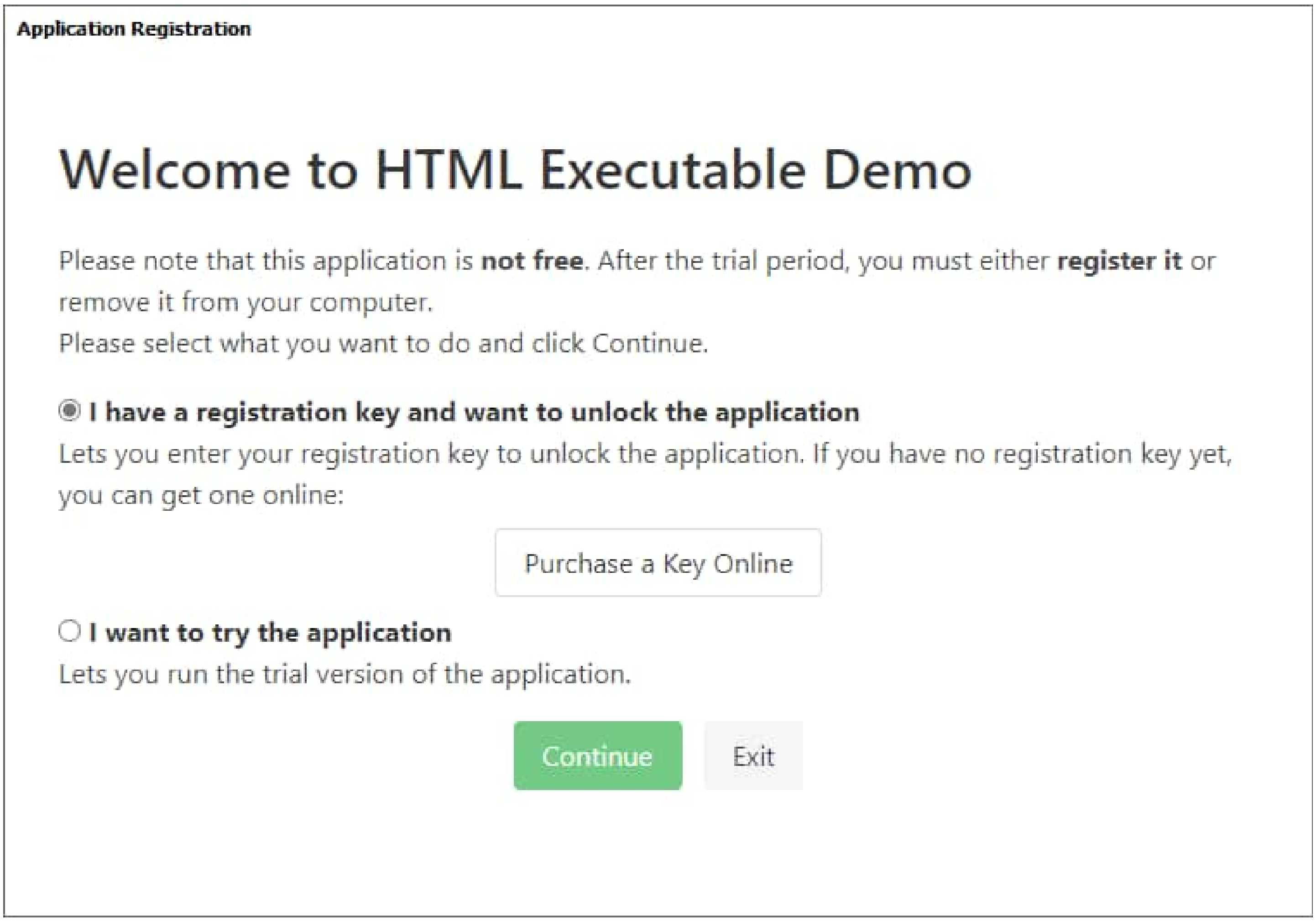
You can choose between a time-based trial period (X-day) or a usage-based one (X-use). Choose the appropriate expiration mode: "days" or "runs".

⚠ **Warning**: Altering the expiration mode may require resetting trial information on your system.

✔ **After Expiration**: Post expiration, users must enter a key to switch from the default certificate to another. For non-default certificates, enabling the "Revert to the default certificate if expired" option lets it revert to the default post-expiration. If not enabled, users won't be able to access the publication.

⚠ **Warning**: Avoid using expiration features for a portable publication. They store settings on the USB disk, allowing users to reset their trial period by deleting the settings files.

Nag Screen or Unlock Screen at Startup


Application Registration

Welcome to HTML Executable Demo

Please note that this application is **not free**. After the trial period, you must either **register it** or remove it from your computer.
Please select what you want to do and click Continue.

[✓ **I have a registration key and want to unlock the application**]
Lets you enter your registration key to unlock the application. If you have no registration key yet, you can get one online:

I want to try the application
Lets you run the trial version of the application.

✔ **Nag Screen**: The nag screen prompts your end users to activate the ebook or publication. It can be enabled or disabled with the "Show the nag screen at startup" option. You can opt for 'Never', 'Only if certificate expired', or 'Always'.

For online orders, provide a URL to your online order page. The "Purchase Online" button in the nag screen will direct users to this URL.

✔ **Unlock Menu**: If you opt not to display the nag screen at startup, you can still include a menu command in your ebook that can invoke it using the "Unlock the program" menu command option.

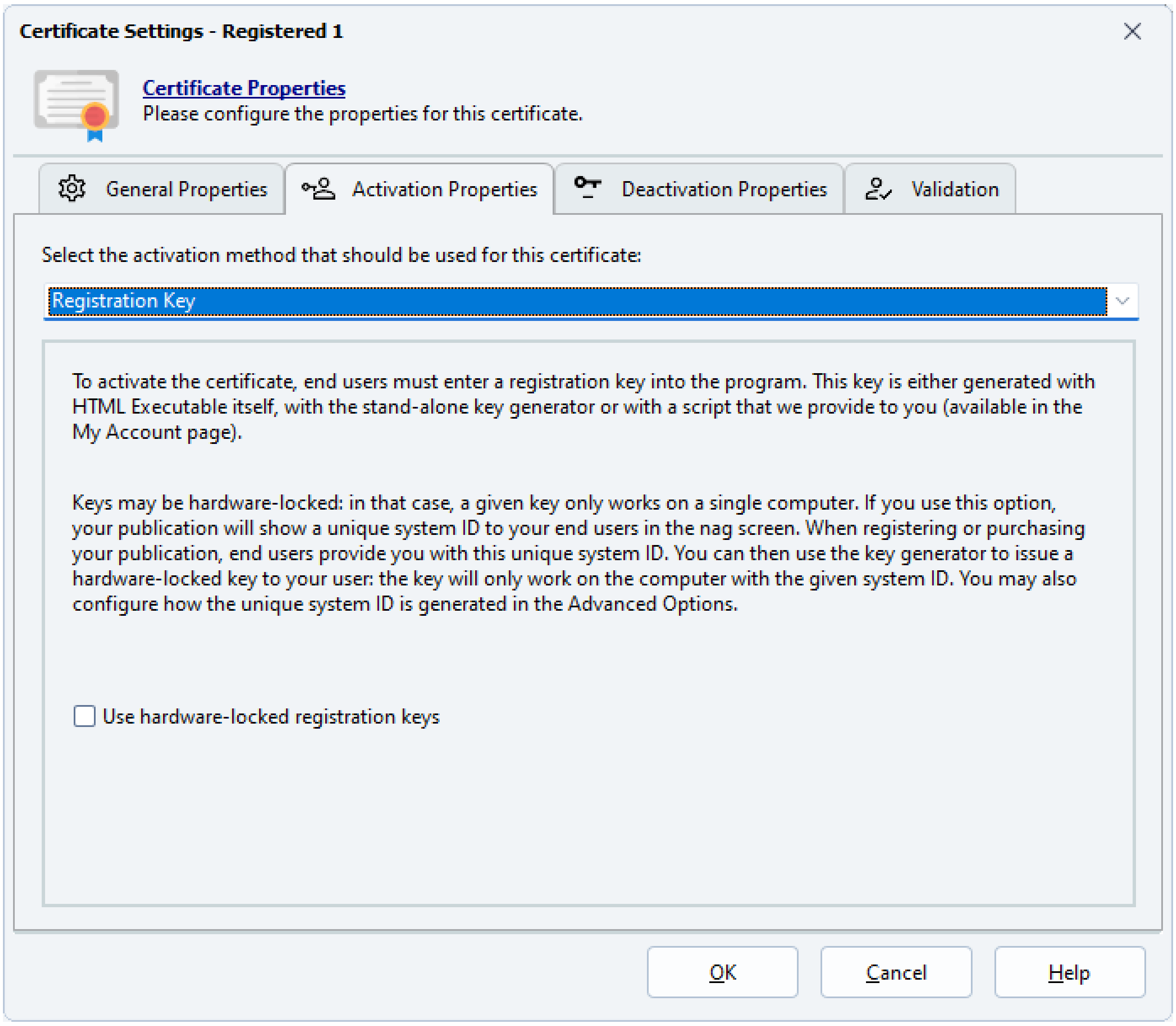
Note: The nag screen can be customized with the Dialog Boxes feature.

⚠ **Warning**: If customizing the HTML code of the nag screen, ensure you do not remove the code that defines existing fields. To hide a field in the nag screen, add a `display:none` style option to the INPUT tag.

3.7.3. Activating a certificate

To allow end users to upgrade to a certificate other than the default one, you can **either provide them with a registration key or offer online activation**. A registration key unlocks a registered certificate (and its feature that can be configured with security profiles), while online activation involves downloading the key from a remote server.

HTML Executable offers **two activation methods to activate a certificate**. To choose a method, go to the Certificate Manager, double-click on a certificate and choose the "**Activation Properties**" tab as shown below:



Select the activation method that should be used for this certificate:

Registration Key

To activate the certificate, end users must enter a registration key into the program. This key is either generated with HTML Executable itself, with the stand-alone key generator or with a script that we provide to you (available in the My Account page).

Keys may be hardware-locked: in that case, a given key only works on a single computer. If you use this option, your publication will show a unique system ID to your end users in the nag screen. When registering or purchasing your publication, end users provide you with this unique system ID. You can then use the key generator to issue a hardware-locked key to your user: the key will only work on the computer with the given system ID. You may also configure how the unique system ID is generated in the Advanced Options.

☐ Use hardware-locked registration keys

Registration Key

End users need a registration key to unlock a registered certificate. After purchase, you generate a registration key using HTML Executable (with the **Make Key** button in the Certificate Manager) or with a key generator script, and provide it to the user. The nag screen prompts the user to enter their registration key upon program startup.

About hardware-locked keys

For added security, you can create hardware-locked keys that tie the certificate to a specific system. This prevents unauthorized distribution of keys. Users must provide their system ID when placing an order, and the key generator incorporates this system ID.

Online Activation

Online activation enables control over who can run the ebook and on which computer. It requires a remote server with the free HTML Executable Activation Kit or an active account with Protect Ebook .net. Registration information is downloaded directly from the server by the publication.

You have to **provide the URL to the activation kit** installed on your server. For instance, if you installed the activation kit in a subfolder named "activation", the URL will be `http://www.yourdomain.com/activation/`

Secure connection thanks to TLS/SSL is supported by HTML Executable, you can use URLs that begin with **https://**

Manual Registration

In case some users don't have an active internet connection, you can enable manual registration. This method works similar to registration keys, allowing users without internet access to enter their key offline.

⚠ Note: Ensure that you are prepared to handle manual registration requests from users without an internet connection.

Please refer to the relevant topics for more details and instructions.

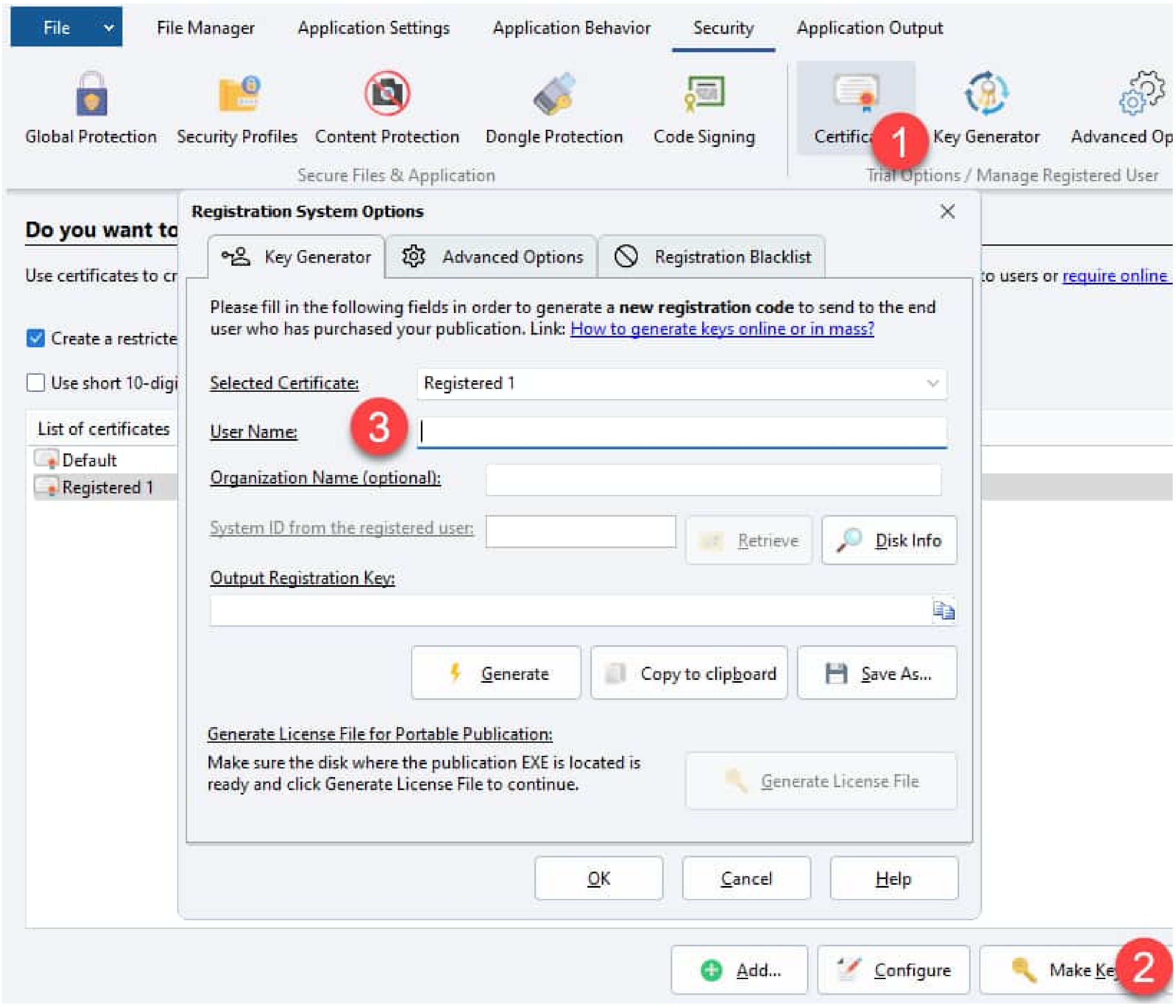
See also: How to Make Keys and How to Deliver Keys Online Instantly with PayPal, Protect Ebook.net Activation Kit, Deactivating A Certificate.

3.7.3.1. Making Registration Keys for Your Ebooks

When a user purchases your publication or ebook made with HTML Executable, providing them with a registration key is essential for accessing the registered version (activating a registered certificate). HTML Executable offers flexible options for creating and delivering these keys, allowing you to streamline the activation process and enhance security.

Manual Key Creation with HTML Executable

Generating registration keys manually is straightforward. You can create keys by clicking the "Make Key" button in the Security => Certificates page, using the Key Generator button in the Security tab, or by pressing CTRL+K.



Registration System Options

Please fill in the following fields in order to generate a **new registration code** to send to the end user who has purchased your publication. Link: How to generate keys online or in mass?

Selected Certificate: Registered 1

User Name:

Organization Name (optional):

System ID from the registered user:

Output Registration Key:

Generate License File for Portable Publication:

Make sure the disk where the publication EXE is located is ready and click Generate License File to continue.

The Key Generator prompts you to enter the user's name, optional company name, and their system ID. Upon generating the key, you can copy it to the clipboard or save it as a text file for distribution.

Note: Registration key data is saved in UTF-8 format.

Tip: Enable the "Use short 10-digit keys instead of normal ones" option in the Certificates page to generate shorter registration keys.

Licensing of Portable Publications

For portable publications, registration information is stored in a license file. HTML Executable simplifies this process by generating the license file directly in the same folder as the publication EXE file. If you are using hardware-locked keys, the system ID can be automatically determined, or you can retrieve it from the disk on which the publication EXE file is created.

Info: Learn more about locking a publication to a specific USB drive in the portable publication topic.

Additional Key Generation Options

HTML Executable provides various options for creating keys manually or in bulk:

- The Professional/Commercial editions offer a stand-alone portable key generator, available in the User Account page.
- Registered users can access scripts (PHP, C#, Pascal, VB, and VBA) for generating keys online on a web server. These scripts can be integrated into your own registration system.
- Subscribers to Protect Ebook .net can generate registration keys using the Protect Ebook .net client.
- To verify the integrity of an uninstall confirmation code (deactivation feature), you can use the dedicated function provided in the key generator scripts or the HTML Executable Activation Kit.

Delivering Keys Online After Payment

If you want to deliver registration keys to customers immediately after orders and prevent unauthorized distribution, HTML Executable offers an online simulation using the key generator PHP script. This simulation showcases the delivery process and the usage of hardware-locked registration keys.

Tip: Explore the working sample of the online key delivery process here.

Alternatively, you can utilize the online activation feature if you have the HTML Executable Activation Kit or a subscription to Protect Ebook .net.

3.7.3.2. About Hardware-locked Keys for Ebooks

Preventing unauthorized distribution and fraudulent purchases is crucial for protecting your ebooks. HTML Executable offers the capability to configure your publication or ebook with **hardware-locked keys**. These keys are unique to each user's computer system, ensuring that the registered version of your ebook can only be accessed on the authorized device.

Info: Hardware-locked keys are designed to restrict the usage of registration keys to a specific computer system.

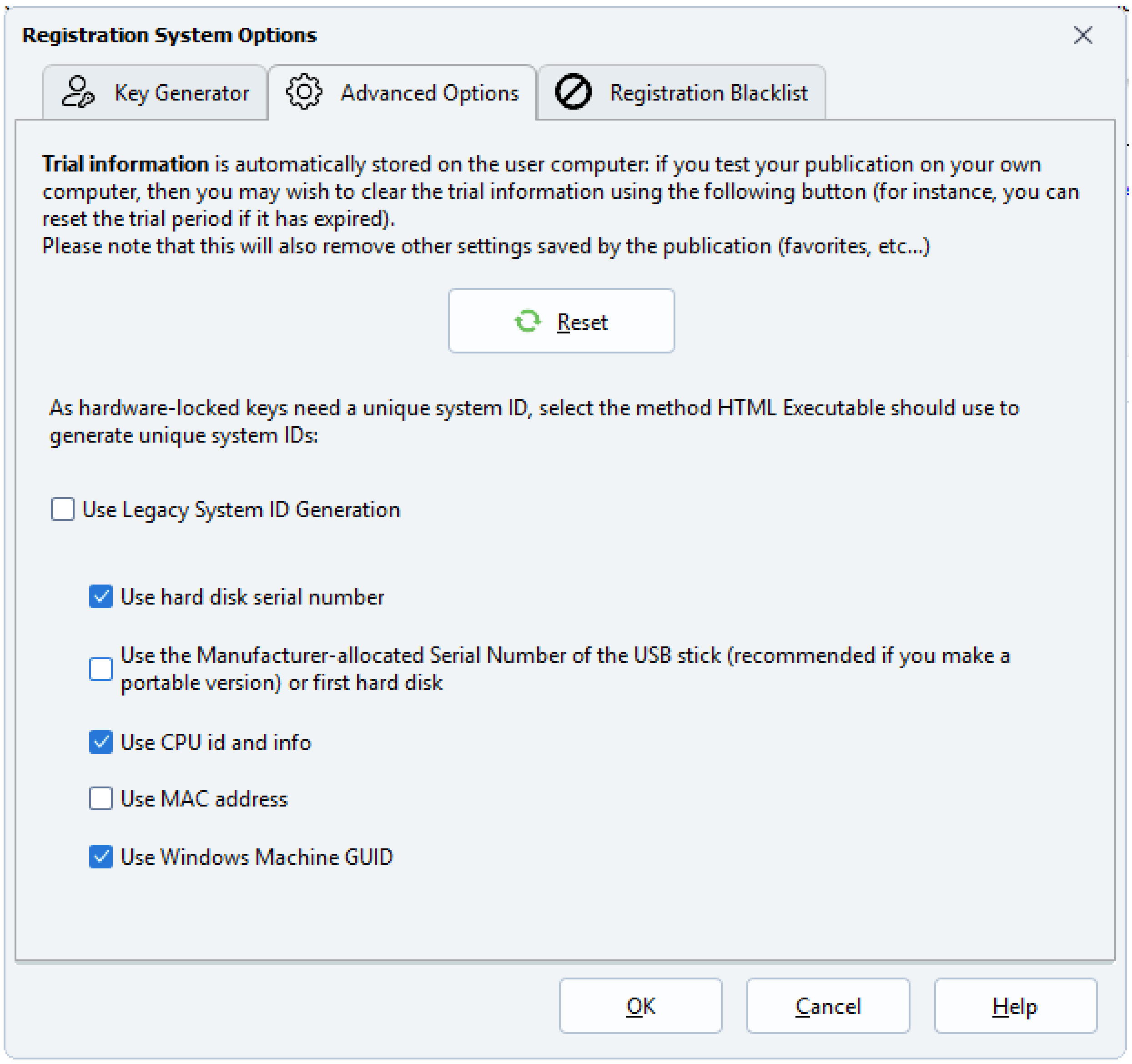
Configuring Your Ebook with Hardware-Locked Keys

You can easily enable or disable the use of hardware-locked keys through the properties of the certificate in HTML Executable. By configuring your ebook to work with hardware-locked keys, you enhance the security of your digital content.

Note: Online activation internally utilizes hardware-locked keys by default.

Customizing the Unique System ID Calculation

To compute the unique system ID for hardware-locked keys, HTML Executable takes into account various properties of the user's hardware components. You have the flexibility to customize the properties used for system ID calculation through the **Advanced Options** in the **Security** page of HTML Executable:


Registration System Options

Trial information is automatically stored on the user computer: if you test your publication on your own computer, then you may wish to clear the trial information using the following button (for instance, you can reset the trial period if it has expired).
Please note that this will also remove other settings saved by the publication (favorites, etc...)

As hardware-locked keys need a unique system ID, select the method HTML Executable should use to generate unique system IDs:

- [ ] Use Legacy System ID Generation
  - [x] Use hard disk serial number
  - [ ] Use the Manufacturer-allocated Serial Number of the USB stick (recommended if you make a portable version) or first hard disk
  - [x] Use CPU id and info
  - [ ] Use MAC address
  - [x] Use Windows Machine GUID

⚠ **Warning:** Not all disks or USB sticks have a manufacturer serial number. To verify whether a disk has a manufacturer serial number, you can use the **Disk Information** tool (in the File menu) provided with HTML Executable.

The available options for unique system ID calculation include:

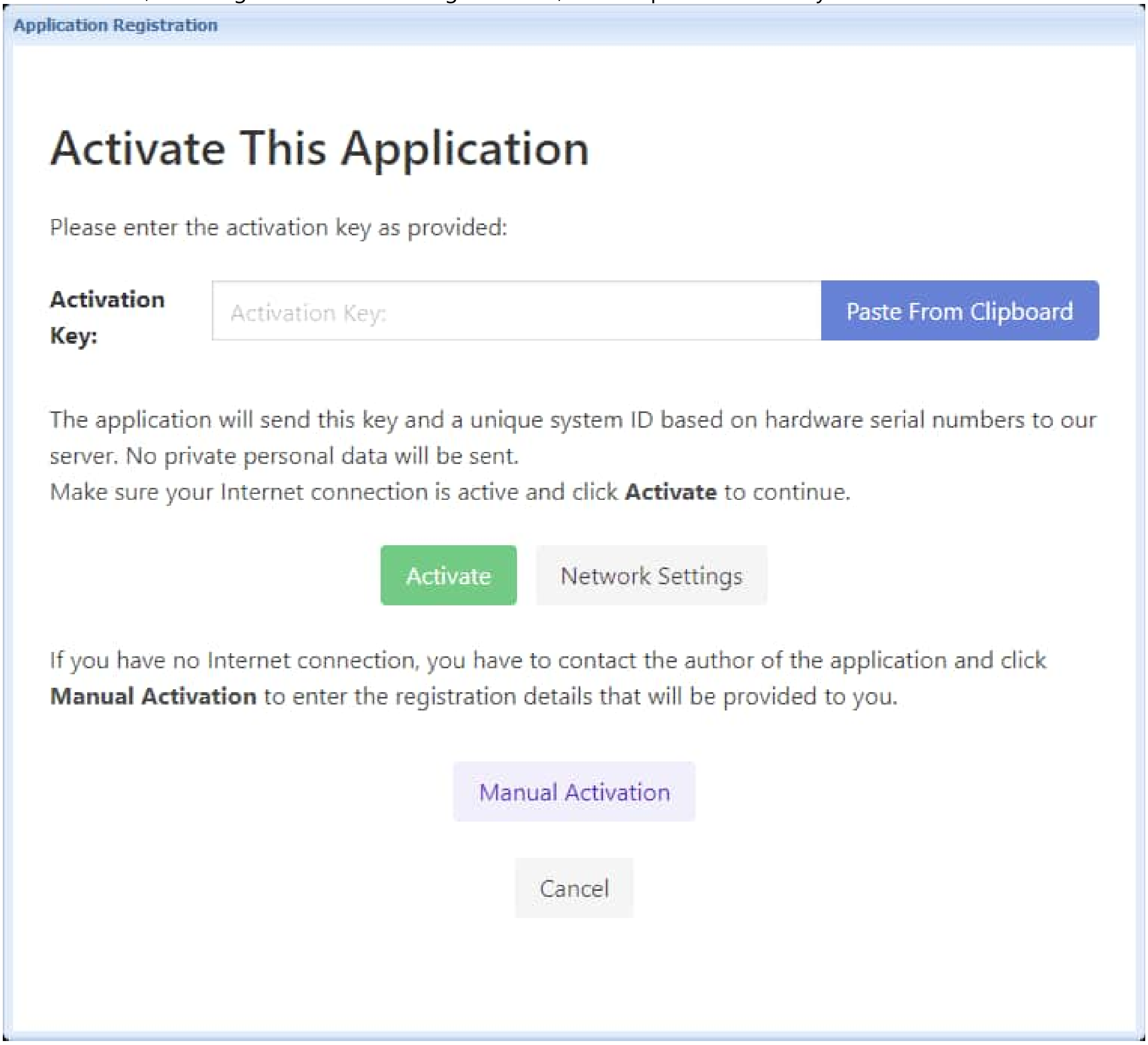
- Use the HD serial number: Based on the volume ID assigned to the main hard disk in Windows.
- Use the Manufacturer-allocated serial number: Retrieves the manufacturer-allocated serial number of the disk on which the EXE file is located (applicable for portable publication or the first available hard disk).
- Use CPU ID and info: Retrieves CPU specifications.
- Use MAC address: Utilizes the MAC address of the computer's network card (if available).
- Use Windows Machine GUID: Uses the Globally Unique Identifier (GUID) generated during the initial installation of the Windows operating system on a machine. It serves as a unique identifier for that specific Windows installation. It is primarily used for system identification, licensing validation, and other internal Windows operations. Note: the Machine GUID is tied to the Windows installation, not the hardware. If you reinstall Windows, the Machine GUID will change, even if the hardware remains the same.

You can choose a combination of different components. However, keep in mind that the more components you select, the higher the probability that one of them will change. This may lead your users to request a new key if their hardware changes.

3.7.4. Online Activation

Online activation is an effective method offered by HTML Executable to **protect and control remote access to your ebooks and publications**. Utilizing online activation provides several advantages, including remote control over activations, the ability to unlock registered editions, and the capability to limit activations to a specific number of computers.

For customers, installing their license is straightforward, as a simple activation key lets them unlock the ebook:


Application Registration

Activate This Application

Please enter the activation key as provided:

Activation Key: Activation Key:

The application will send this key and a unique system ID based on hardware serial numbers to our server. No private personal data will be sent.
Make sure your Internet connection is active and click **Activate** to continue.

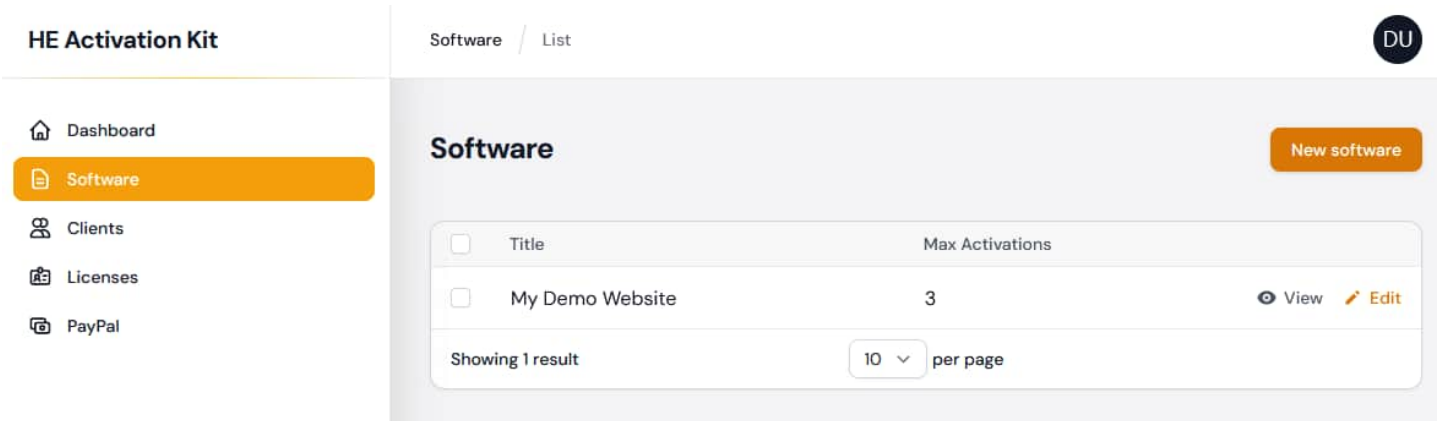
If you have no Internet connection, you have to contact the author of the application and click **Manual Activation** to enter the registration details that will be provided to you.

Note: Online activation requires a remote web server with the free HTML Executable Activation Kit installed, or an active subscription to our hassle-free ebook protection service, Protect Ebook .net.

Throughout this documentation, the remote server responsible for handling activations and managing the database of ebooks and customers will be referred to as the **'activation server'**.

By integrating online activation into your publication, you enable seamless interaction with the activation server, ensuring automation and increased productivity.

The Activation Kit or Protect Ebook .net provides you with powerful tools to manage your ebooks and customers online:



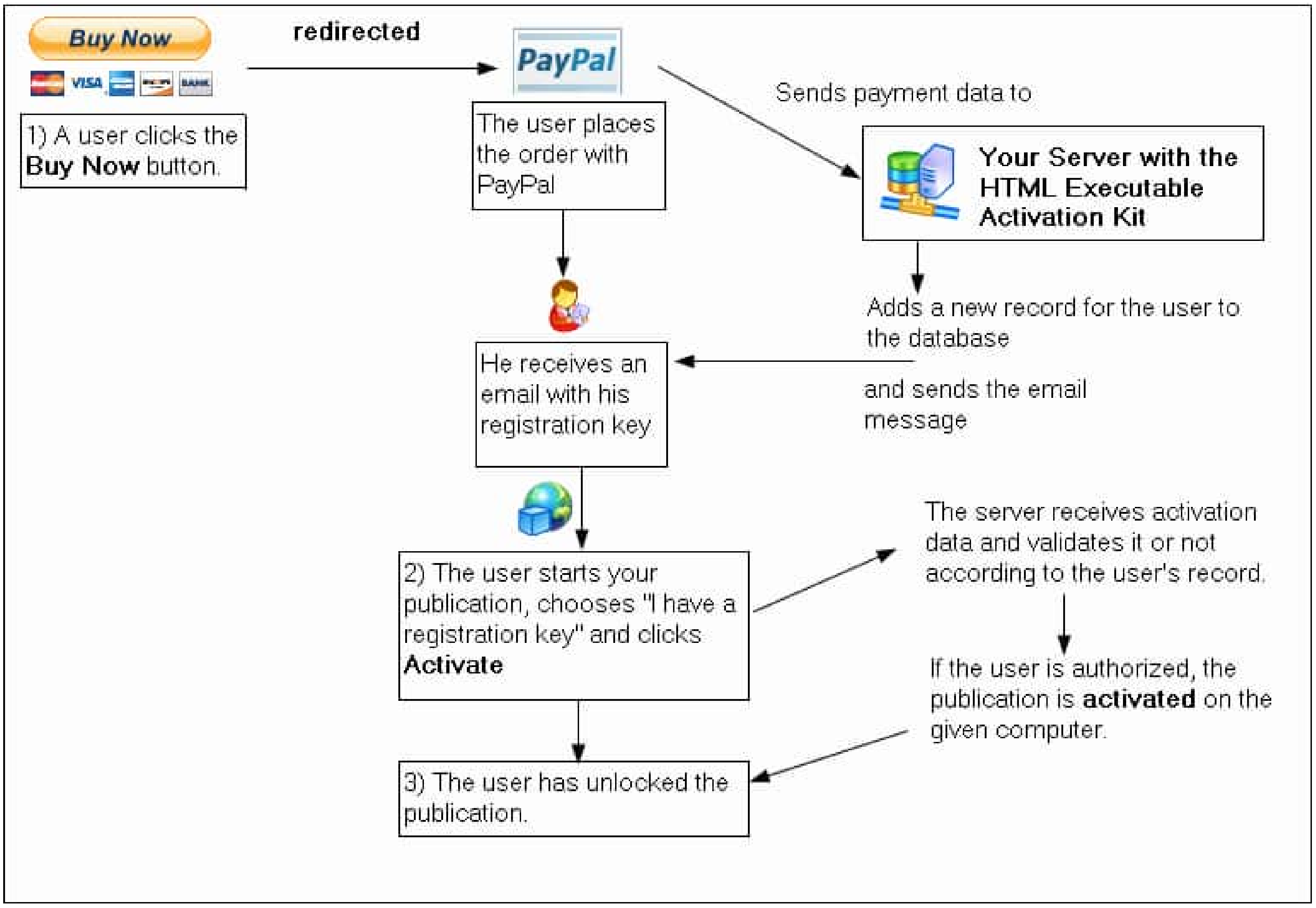
	Title	Max Activations	
☐	My Demo Website	3	View Edit

Showing 1 result 10 per page

💡 **Tip:** Experience our online shop simulation, showcasing the usage of our online activation system.

Activation Steps

Consider the following scenario: you sell an ebook through your website, with customers making purchases online, for example, through PayPal.


Buy Now
redirected
PayPal
1) A user clicks the Buy Now button.
The user places the order with PayPal
Sends payment data to
Your Server with the HTML Executable Activation Kit
Adds a new record for the user to the database
and sends the email message
He receives an email with his registration key
2) The user starts your publication, chooses "I have a registration key" and clicks Activate
The server receives activation data and validates it or not according to the user's record.
If the user is authorized, the publication is activated on the given computer.
3) The user has unlocked the publication.

1) The user places an order for your publication via your authorized reseller (e.g., PayPal). The order details, along with user information, are sent to the activation server, which is configured with the HTML Executable Activation Kit or our Protect Ebook .net activation service. An account is created in the database, and a unique activation key is generated and sent to the end user (e.g., via email).
2) The user launches the publication and enters the activation key in the nag screen. By clicking Activate, the activation process initiates.
3) The publication establishes a connection with the activation server and shares necessary information, such as the registration key and unique system ID (for hardware-locked keys). Importantly, no private personal data is transmitted.
4) The activation server retrieves the account associated with the activation key from its database. It verifies the number of allowed activations, account status (locked/unlocked), and other relevant details. If the registration is permitted, the server generates a registration key and sends it back to the publication, along with the user's name and organization (if provided).
5) The publication validates the registration data received. If the activation is authorized by the server, the corresponding certificate in the publication is successfully activated. Otherwise, an error message is displayed.

Users can configure a proxy server using the **Network Settings** if they need to connect to the internet through a proxy. The publication will retain this proxy setting for subsequent deactivation and validation processes. If the proxy server needs to be changed, it can be done via the command line.

Through the free HTML Executable Activation Kit or the utilization of Protect Ebook .net, the entire activation process can be automated, ensuring a seamless experience for both you and your end users.

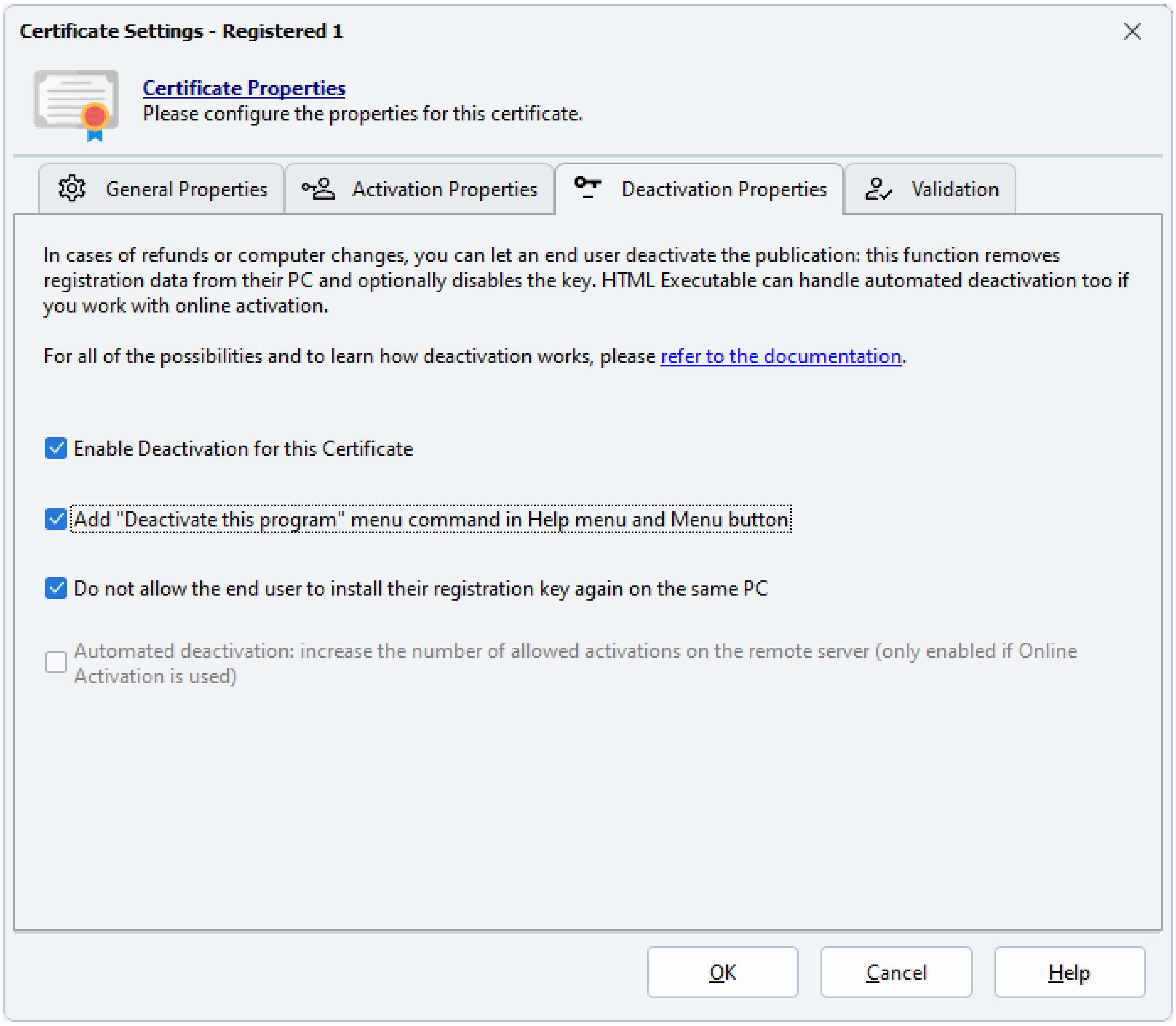
Info: Experience our online shop simulation, showcasing the usage of our online activation system.

3.7.5. Deactivate a Certificate

Your users have the ability to **deactivate their registered key and revoke it** permanently on a specific computer. Deactivation becomes necessary when users want to transfer their registered publication or ebook to another PC, ensuring the removal of the publication and activation from the current PC. Proof of deactivation is required for a refund or issuance of a new activation key.

Another scenario where deactivation is applicable is when accepting refunds for ebooks, contingent upon users providing proof of deactivation.

✔ To enable the deactivation process for your ebook or publication, navigate to the Certificate Manager, double-click on a certificate, and select the "**Deactivation Properties**" tab:



In cases of refunds or computer changes, you can let an end user deactivate the publication: this function removes registration data from their PC and optionally disables the key. HTML Executable can handle automated deactivation too if you work with online activation.

For all of the possibilities and to learn how deactivation works, please refer to the documentation.

- [✓] Enable Deactivation for this Certificate
- [✓] Add "Deactivate this program" menu command in Help menu and Menu button
- [✓] Do not allow the end user to install their registration key again on the same PC
- [ ] Automated deactivation: increase the number of allowed activations on the remote server (only enabled if Online Activation is used)

⚠ **Note:** Users can only deactivate the ebook if it was previously registered with an activated certificate. Deactivation is not supported in Trial mode.

Description of the Deactivation Process

By enabling deactivation for a certificate, end users can uninstall their registration data (key or activation) and receive a unique uninstall confirmation code in return. There are two ways for users to initiate the deactivation process:

- Launch the publication EXE file with the **deactivate** parameter, such as using the Windows Start -> Run command: `PUBFILE.EXE deactivate`
- Start the publication and choose the "Help | Deactivate this program" menu command (if enabled in the "Deactivation Properties" tab).

Confirmation of the deactivation request is required. Upon successful deactivation, the registration key is deleted or invalidated, and a message box displaying the uninstall confirmation code appears. The code is also copied to the Windows clipboard, allowing users to easily share it via email. Additionally, the publication closes and reverts back to the Default certificate, immediately expiring if the user attempts to run it again.

To verify the integrity of the uninstall confirmation code, you can utilize the provided scripts (C#, PHP, etc.)

available to registered users or leverage the uninstall code validation feature in the HTML Executable Activation Kit.

Preventing Reinstallation of Registration Key

Enabling this option permanently revokes the registration key when the user deactivates the ebook. If the user attempts to reenter the key to activate the ebook, it will be considered invalid.

> **Info:** This option is recommended unless you wish to allow users to reactivate the ebook on the same computer.

Automated Deactivation

> **Note:** This option is only available when using online activation.

✔ Online activation provides control over the number of computers on which end users can activate your ebook. Utilizing the free HTML Executable Activation Kit or Protect Ebook .net, the remote activation server tracks the activations. By enabling automated deactivation, the activation server takes into account the deactivations as well.

When automated deactivation is active, the publication sends a notification to the activation server, which increments the number of allowed activations (according to the limit set via the HTML Executable Activation Kit or Protect Ebook .net).

For instance, let's consider a scenario where John is allowed to use your ebook on two different computers. John has already activated the ebook on both computers but plans to purchase a new one to replace his old computer. Since John has reached the activation limit, he cannot install the ebook on his new computer. The solution is to deactivate the ebook on his old computer.

The deactivation process disables the ebook on John's old computer and sends a notification to the activation server. The number of allowed activations increases from 0 to 1, enabling John to activate the ebook on his new computer.

> **Info:** In case of computer crashes, where the end user is unable to perform the deactivation process, you have the option to manually increase the number of allowed activations yourself.

Automated deactivation **requires an Internet connection**. If no Internet connection is available, or if an error occurs while sending notification to the remote server, the deactivation process will still continue and the publication will display an **uninstall confirmation code** that the user can send you (as explained above). If the customer uses a proxy server, he should have already configured it when activating the publication and the latter will reuse the same proxy settings. If the proxy server has to be changed, it can be done through command line.

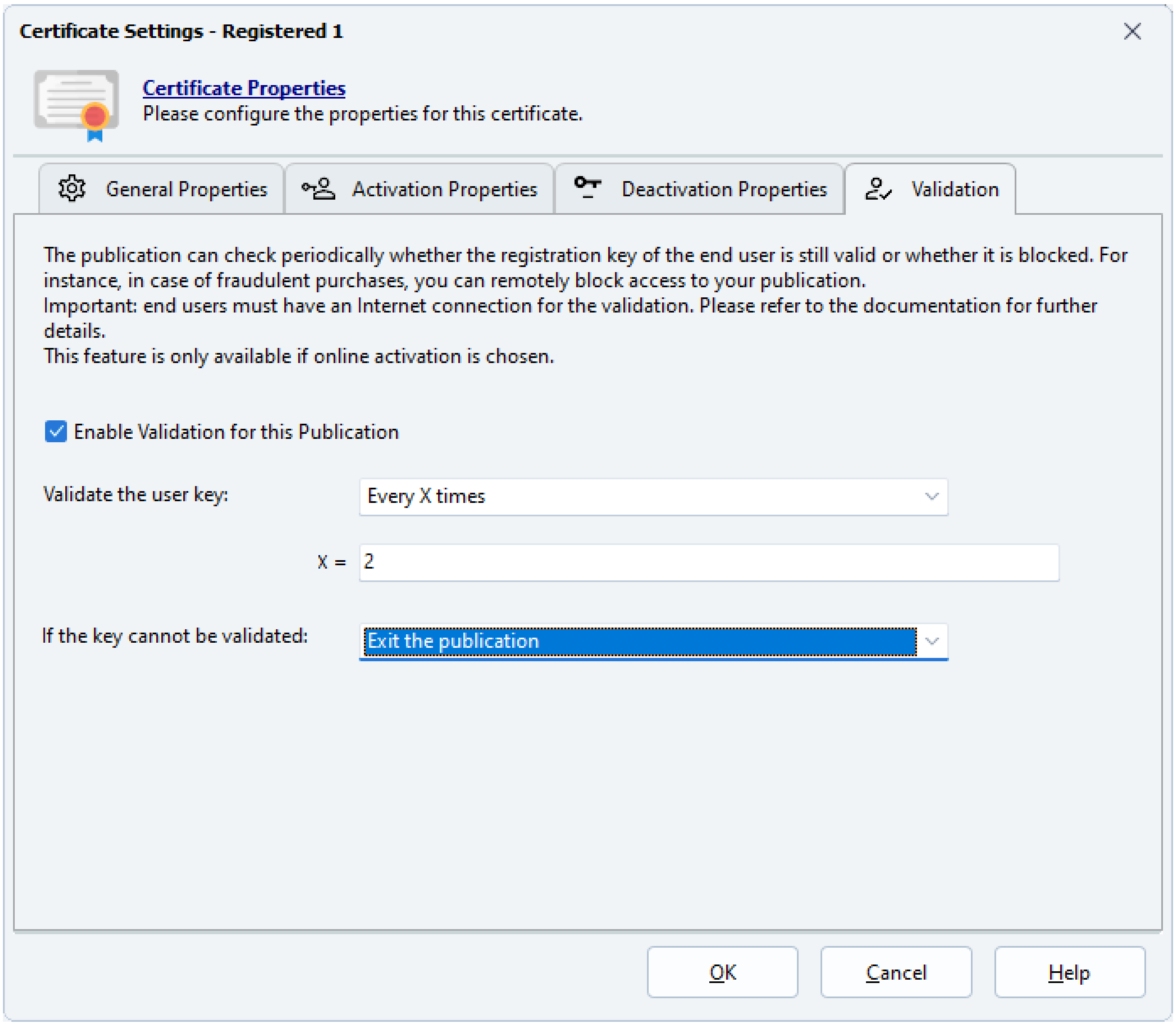
3.7.6. Validation of User Licenses

Adding License Validation to Your Ebooks or Publications

HTML Executable empowers you to enhance your ebooks or publications by incorporating license validation. This validation process ensures the verification of the end user's registration key before granting access to the ebook. By periodically performing validation, **you gain control over the usage of your ebook**, enabling you to remotely block access in cases of fraudulent purchases or refunds.

To **integrate a validation process into your ebook**, follow these steps:

1) Open the Certificate Manager.
2) Double-click on a certificate.
3) Select the "**Validation**" tab.



Certificate Settings - Registered 1

Certificate Properties
Please configure the properties for this certificate.

The publication can check periodically whether the registration key of the end user is still valid or whether it is blocked. For instance, in case of fraudulent purchases, you can remotely block access to your publication.
Important: end users must have an Internet connection for the validation. Please refer to the documentation for further details.
This feature is only available if online activation is chosen.

☑ Enable Validation for this Publication

Validate the user key: Every X times

X = 2

If the key cannot be validated: Exit the publication

Note: This option is available only for **online activation**.

⚠ **Warning:** End users must have an active internet connection to perform validation. **Validation requires a remote web server** with the HTML Executable Activation Kit installed or an **active subscription** to our ebook protection service, Protect Ebook .net.

About the Validation Process

The validation process is automated, where the publication seamlessly interacts with the activation server. During validation, the publication transmits the activation key, registration key, and unique system ID to the activation server. **No personal data is sent**.

The activation server scrutinizes the received information and queries the user's account in the databases. If the account is marked as "blocked" or an error occurs, validation is negative. Conversely, if the account is valid, the activation server issues a validation code. The publication subsequently verifies this code to determine access privileges.

> **Info:** Customers using a proxy server should configure it during activation. The publication will reuse the same proxy settings. Changing the proxy server can be done via the command line.

Validating the User Key

You have the flexibility to select the **validation frequency**. For instance, you can check the validity every X times the publication is run by specifying the value for X (an integer, e.g., 3).

Advanced users can customize the validation process in two ways:

Manually with HEScript calls: Compose your own HEScript scripts and invoke the validation process using the following HEScript code:

```
if not RunSimpleBooleanFunc("Trial.DoServerValidate", false) then
begin
    // Actions to perform when the validation is unsuccessful, such as displaying
an error message and exiting the publication
    ShowMessage("Could not validate your license. Please ensure you have a valid
subscription");
    ExitPublication;
end;
```

Using the OnKeyValidation event (in UserMain): Set the desired frequency, and your publication will trigger the UserMain.OnKeyValidation HEScript event, which you can customize on the User Scripting page.

```
function OnKeyValidation: Boolean;
begin
    Result := True; // Returns True to perform validation
end;
```

Handling Invalid Key Validation

Choose the appropriate action **if the validation is unsuccessful**:

- Exit the publication: Displays an error message and exits.
- Return to the Trial state: The publication reverts to the Default certificate and exits, erasing the registration information. However, this option is not recommended.
- Call the OnKeyNotValidated HEScript event (in UserMain): The publication invokes the UserMain.OnKeyNotValidated HEScript event, which you should customize on the User Scripting page.

```
function OnKeyNotValidated: Boolean;
begin
    MessageBox("This program will close now because it could not verify your
license", "User Error", MB_OK);
```

```
    Result := True; // Returns True to exit the program
end;
```

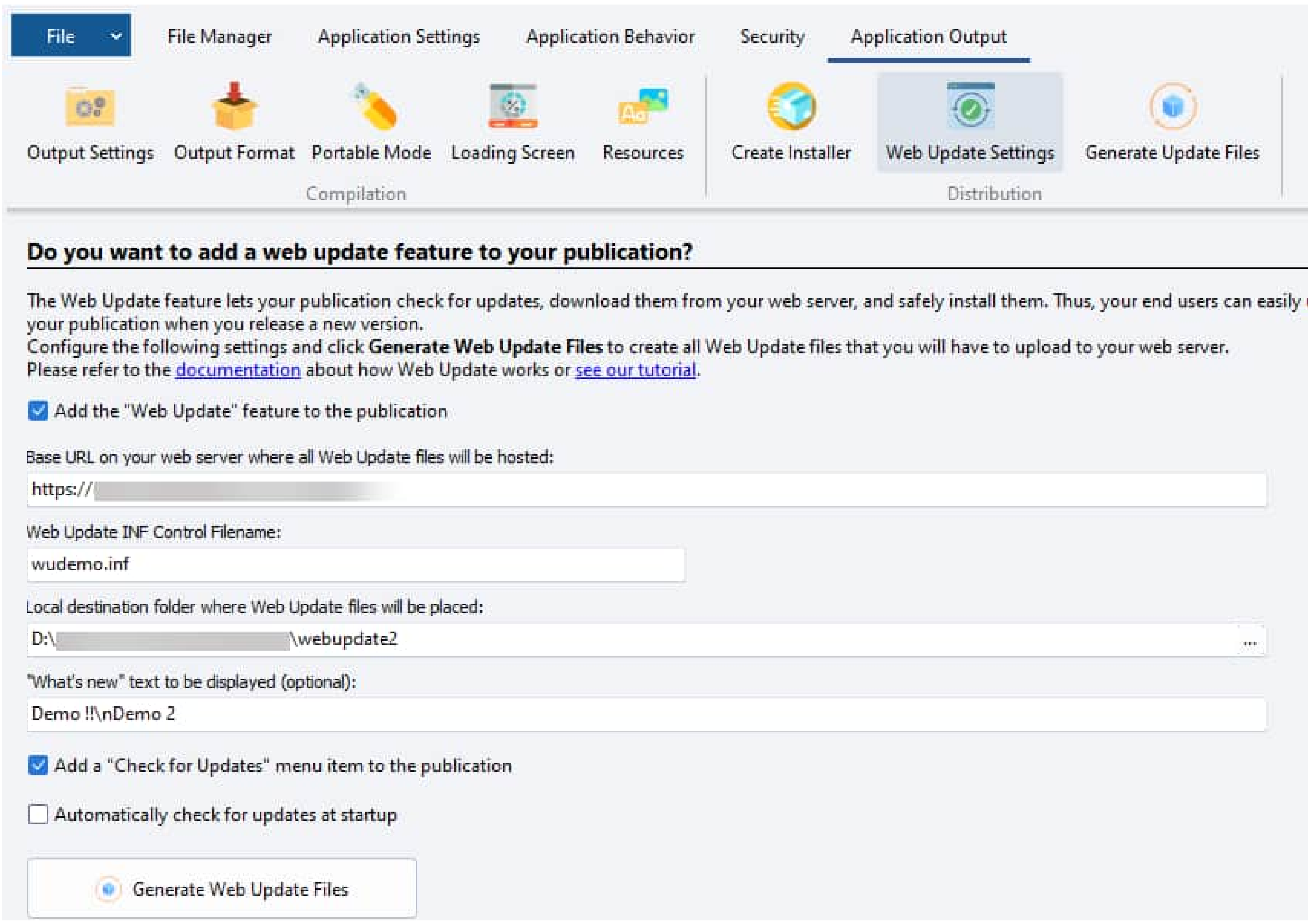
ⓘ **Info:** You can easily customize messages related to the validation process using the Language page. The publication utilizes resource strings named `SValidationXXX.`

3.8. Web Update for Publications and Ebooks

HTML Executable offers you the functionality to add a **Web Update** feature to your publications and ebooks. When a new version is released, the Web Update feature can autonomously download and safely install it from your web server. This ensures that your end users always have access to the latest version of your ebook product and can easily upgrade when a new release becomes available.

No third-party software is required for this process, although a web server is necessary to host the web update files generated by HTML Executable.

ⓘ Take a look at our Web Update tutorial video to get a clear understanding of the feature.



Do you want to add a web update feature to your publication?

The Web Update feature lets your publication check for updates, download them from your web server, and safely install them. Thus, your end users can easily your publication when you release a new version.
Configure the following settings and click **Generate Web Update Files** to create all Web Update files that you will have to upload to your web server.
Please refer to the documentation about how Web Update works or see our tutorial.

☑ Add the "Web Update" feature to the publication

Base URL on your web server where all Web Update files will be hosted:
https://

Web Update INF Control Filename:
wudemo.inf

Local destination folder where Web Update files will be placed:
D:\ \webupdate2

"What's new" text to be displayed (optional):
Demo !!\nDemo 2

☑ Add a "Check for Updates" menu item to the publication

☐ Automatically check for updates at startup

Generate Web Update Files

The Web Update Process

The update process involves a few steps:

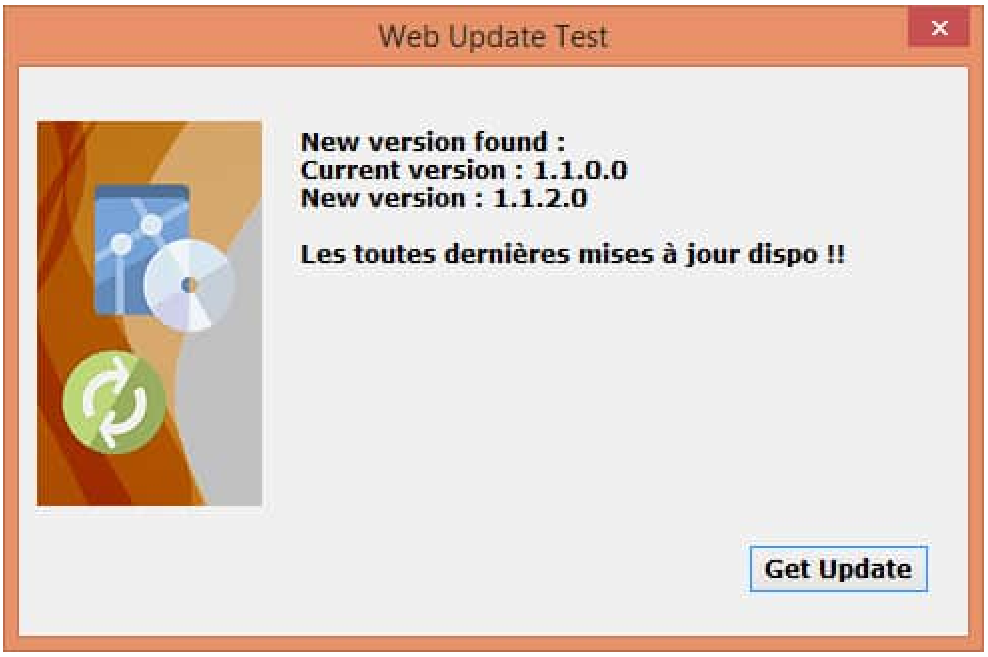
1) Fetching the control file (.INF file) from your web hosting server (HTTP/HTTPS based location).
2) Processing the .INF file - this involves verifying new files, versions, and downloading necessary files. Update files not involved in the publication's running process (.EXE, .DLL's) are immediately extracted to their destination.
3) In the final step, if necessary, the current publication is closed, new EXE and/or DLL files are extracted, and the latest version of the publication automatically starts.

Checking for Updates

You can configure your publication to either automatically check for updates at startup or add a **"Check for Updates"** option to your publication's menu.

✓ Updates can also be triggered through script calls, specifically by invoking the `Global.HEStartWebUpdate` HEScript function.

Once a new version is detected, users are informed and guided through the Web Update wizard, where they can choose whether to install the update:

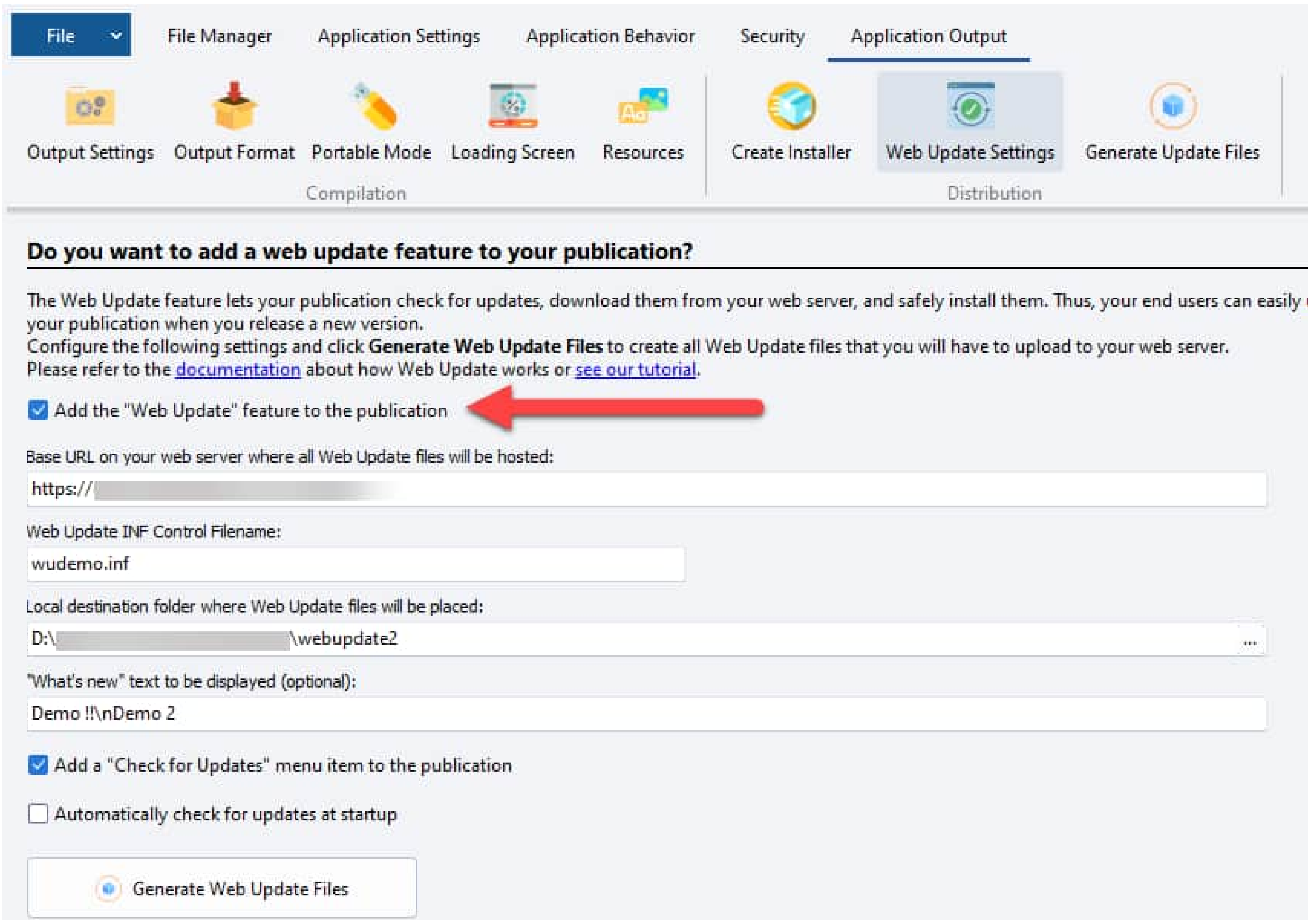


New version found :
Current version : 1.1.0.0
New version : 1.1.2.0

Les toutes dernières mises à jour dispo !!

The version number specified in the **Icon / Version** page of HTML Executable is used to detect new updates, so make sure to increase this number when you release a new version.

Configuration of Web Update Settings



Do you want to add a web update feature to your publication?

The Web Update feature lets your publication check for updates, download them from your web server, and safely install them. Thus, your end users can easily
your publication when you release a new version.
Configure the following settings and click **Generate Web Update Files** to create all Web Update files that you will have to upload to your web server.
Please refer to the documentation about how Web Update works or see our tutorial.

☑ Add the "Web Update" feature to the publication

Base URL on your web server where all Web Update files will be hosted:
https://

Web Update INF Control Filename:
wudemo.inf

Local destination folder where Web Update files will be placed:
D:\ \webupdate2

"What's new" text to be displayed (optional):
Demo !!\nDemo 2

☑ Add a "Check for Updates" menu item to the publication

☐ Automatically check for updates at startup

Generate Web Update Files

To set up the Web Update feature, you need to specify:

- **Base URL**: The location on your web server where all Web Update files will be hosted. This should be an HTTP/HTTPS URL where all update files (control and CAB files) are available for download. For example, `https://www.yourwebsite.com/myfolder`
- **Web Update INF Control Filename**: The control file with .INF format that contains necessary instructions and data about the web update.
- **Local destination folder**: The folder on your local disk where HTML Executable will generate web update files. You need to upload the entire contents of this folder to your web host.

⚠ Warning: Make sure the destination folder is empty. If not, you'll be asked to delete its contents before generating files.

- **"What's new" text**: (Optional) The text to display to your users in the Web Update Wizard when a new upgrade is available.
- **Check for Updates menu item**: This option adds a "Check for Updates" item to the Help menu of your publication.
- **Automatic update check at startup**: Enable this option to let your publication check for updates at startup without user interaction.

Generating Web Update Files

HTML Executable generates all necessary files (control and CAB files) for the web update in the specified local destination folder. You'll then upload these files to your web host. These files include:

- The .INF control file with instructions for the publication
- The .CAB compressed file containing the new publication EXE file and its associated external files

Note: The .CAB file is a standard Microsoft Cabinet file. You can open/edit it with tools like 7-Zip if you need to add more files.

Post-Update Notification

After a successful web update, you may want to display a message or a web page to your end users. HTML Executable sets the global variable **`herestartwebupt`** to `1` after a successful web update, allowing you to trigger desired actions through HEScript scripting.

Here's a script example you can use in the OnStartMainWindow event:

```
procedure OnStartMainWindow;
begin
  // When the main window is going to be displayed (just before the homepage is
shown).
  if GetGlobalVar("herestartwebupt", "0") = "1" then
  begin
    ShowMessage("Welcome to your new publication!"#13#10"What's new:");
  end;
end;
```

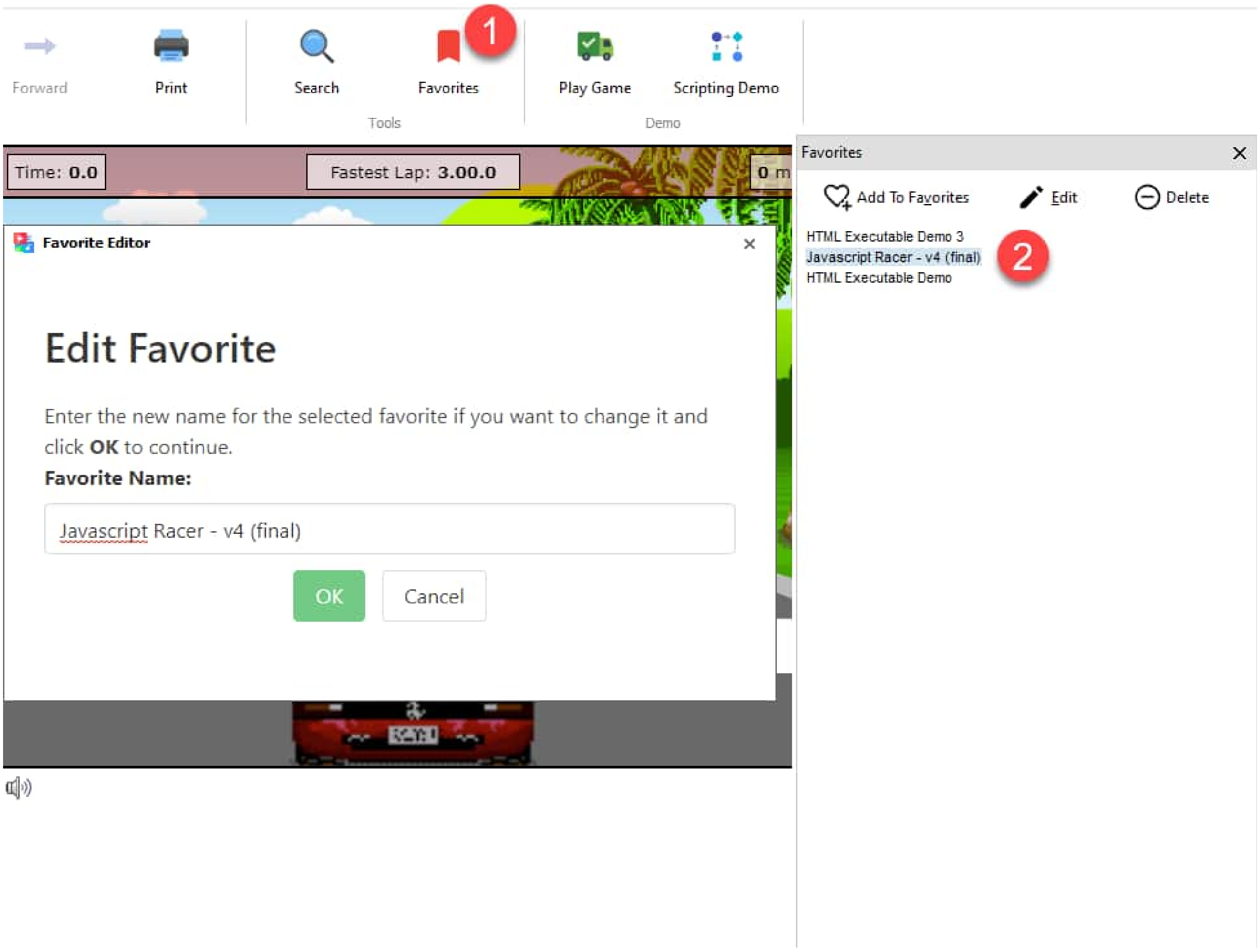
The `#13#10` characters insert a new line. You can also choose to show a custom dialog box.

Additional Notes

- If the publication EXE file is installed in a Windows system folder such as Program Files, administrator privileges are required, triggering a UAC prompt.
- If you wish to customize the billboard image of the Web Update Wizard, add a bitmap file named `hecustomuwbillboard.bmp` in the publication root folder.

3.9. User Favorite Manager for your Ebook

Ebooks or publications developed with HTML Executable support a feature that your audience is already familiar with: favorites, or bookmarks. This feature functions similarly to how bookmarks operate in popular browsers like Edge, Chrome, or Firefox, making it an intuitive tool for users navigating through your digital content.



Edit Favorite

Enter the new name for the selected favorite if you want to change it and click **OK** to continue.

Favorite Name:

Javascript Racer - v4 (final)

Disabling the Favorite Manager

While there are no direct settings to modify how the Favorite Manager functions, it's possible to remove it completely if it doesn't serve your needs. By following these steps, you can eliminate the Favorite Manager from your ebook or publication:

Navigate to `Application Settings => Visual Controls`.
Set the **`NoFavorites`** property of the `Menu Bar` to `True`.

By doing this, you ensure that the Favorite Manager doesn't appear in your ebook or publication. This can help keep your user interface streamlined and focused on the essentials if bookmarking isn't a feature you want to offer.

3.10. Cloning a project

Have you ever wanted to compile a website using the settings from an existing project? With HTML Executable, it's entirely possible, thanks to the **Clone Project feature**. This powerful feature enables you to create a replica of an existing project with a new source folder, maintaining the original project's settings. It is also handy when you need to shift your files to another directory.

Accessing the Clone Project Feature

You can access this feature in just a couple of clicks:

1. Open HTML Executable.
2. Click the File menu, select "Project" and then click **"Clone Project"**.

Step-by-Step Guide to Cloning a Project

Here's a walkthrough on how to use the Clone Project feature:

- **Open the project file**: Start by opening the project file that you intend to clone.
- **Initiate cloning**: Navigate to the File menu and select the **"Clone Project"** command. A window will pop up providing some instructions. Proceed by clicking **Continue**.
- **Select the new index page**: At this stage, you'll need to specify the HTML file that will serve as the index page for your new cloned publication. The folder containing this page will automatically become the new source folder.
- **Enter the filename for the new project**: You'll then be prompted to provide a filename for your new project.
- **Allow the cloning process to complete**: HTML Executable will initiate a new project, applying the settings from the original project, and then scan the new source folder. Once the process is complete, the project file is saved and then reloaded for memory optimization.

Please note that the Clone Project feature does not retain the original project's file list. Therefore, some links used in options like the Table of Contents may be disrupted. As a result, you might need to make some additional modifications to fix these links manually.

3.11. Locking ebook to USB sticks

Locking a Portable Publication or Ebook to a Specific USB Disk

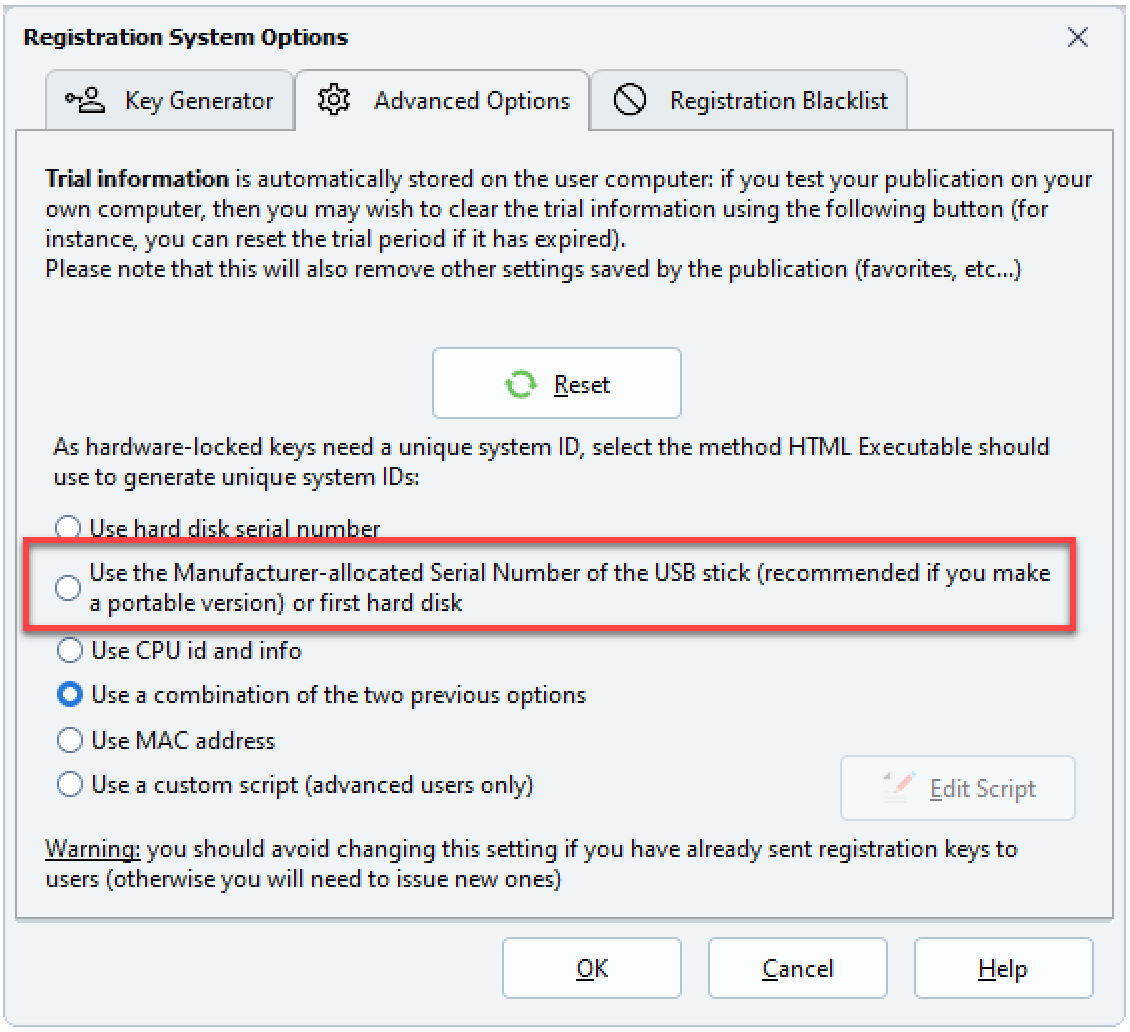
If you create a trial or restricted ebook, you can configure it so that it works only on a given USB disk/stick. For instance, clients of a publication should have the option to show the secured publication on every computer but should not have the ability to copy it to another USB-stick or computer.

⚠ **Warning**: Some disks or USB sticks **do not have a manufacturer serial number**. Only USB sticks with such serial number will work properly. In order to find out whether this is the case or not, you can use the **Disk Information** tool that comes with HTML Executable: in HTML Executable, click **File** and choose the **Disk Information Tool** menu.

Steps to Lock a Portable Publication

- Ensure you have a **portable publication** (Application Output > Portable Mode page).
- Go to the Security > Certificates page and enable "**Create a restricted publication**".
- Double-click on **Default** and turn on "**Do not allow access to the publication without prior registration**".
- Choose **OK** and double-click on **Registered**. Choose **Activation Properties**, **Registration Key** and enable "**Use hardware-locked registration keys**". Choose **OK**.

➢ ⚠ **IMPORTANT** for portable publications: click **Advanced Options** and select **Use the Manufacturer-allocated serial number of the USB disk** option.



Trial information is automatically stored on the user computer: if you test your publication on your own computer, then you may wish to clear the trial information using the following button (for instance, you can reset the trial period if it has expired).
Please note that this will also remove other settings saved by the publication (favorites, etc...)

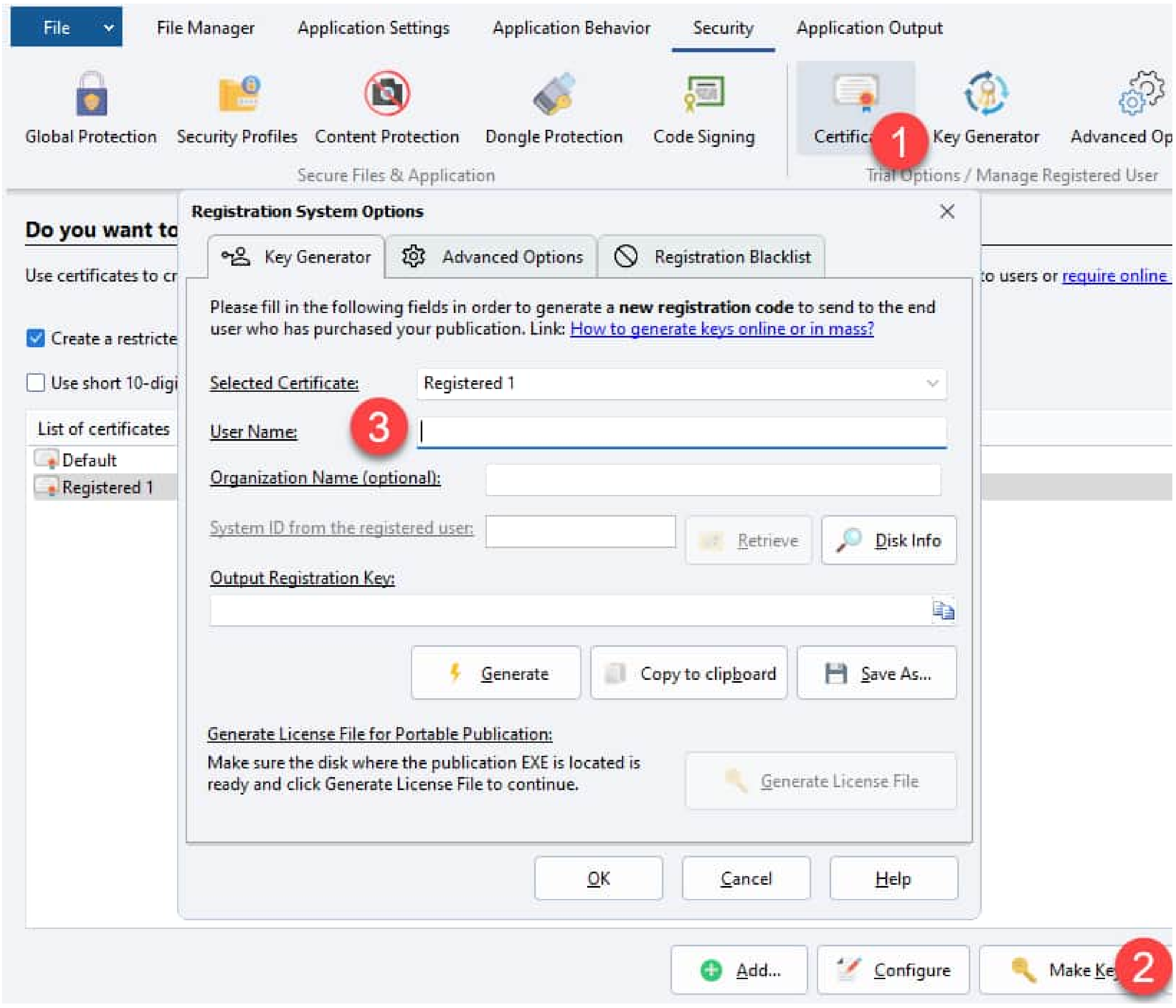
As hardware-locked keys need a unique system ID, select the method HTML Executable should use to generate unique system IDs:

- ○ Use hard disk serial number
- ○ Use the Manufacturer-allocated Serial Number of the USB stick (recommended if you make a portable version) or first hard disk
- ○ Use CPU id and info
- [✓ Use a combination of the two previous options]
- ○ Use MAC address
- ○ Use a custom script (advanced users only)

Warning: you should avoid changing this setting if you have already sent registration keys to users (otherwise you will need to issue new ones)

➢ Build your publication and copy it to the USB disk.
➢ You can **activate the registered certificate of your choice yourself**: end users who will run your publication do not need to enter anything as the publication is already pre-registered by you.

To do this, in the Security ribbon, click **Key Generator**. Enter the name of the registered user, and verify that the destination USB drive with the publication EXE file is ready.



Registration System Options

Key Generator | Advanced Options | Registration Blacklist

Please fill in the following fields in order to generate a **new registration code** to send to the end user who has purchased your publication. Link: How to generate keys online or in mass?

Selected Certificate: Registered 1

User Name:

Organization Name (optional):

System ID from the registered user:

Retrieve | Disk Info

Output Registration Key:

Generate | Copy to clipboard | Save As...

Generate License File for Portable Publication:

Make sure the disk where the publication EXE is located is ready and click Generate License File to continue.

Generate License File

OK | Cancel | Help

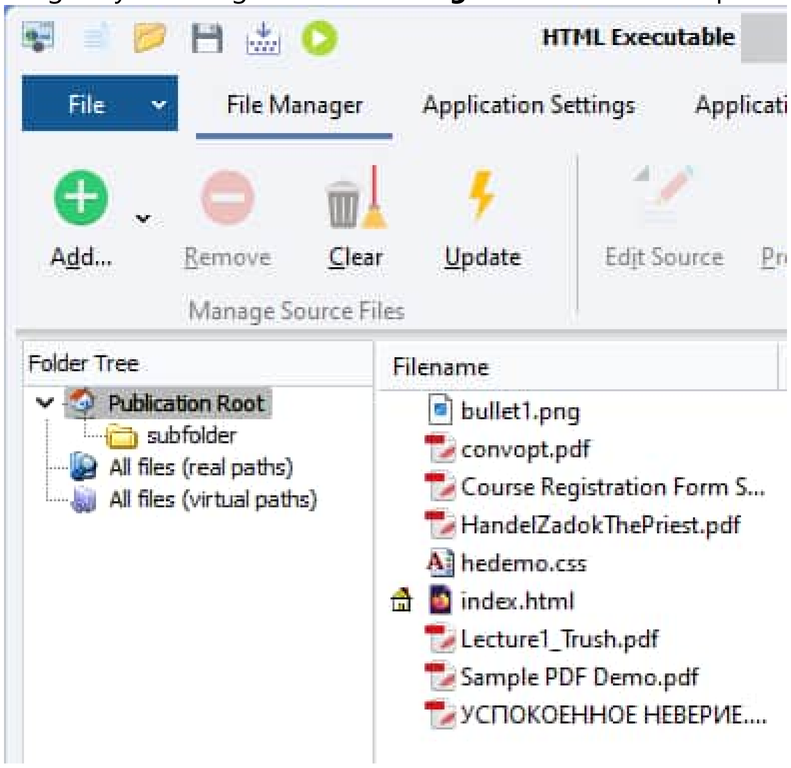
Click **Generate License File**: HTML Executable reads the system ID of the USB drive, and generates the appropriate license file directly on the USB drive. **Your USB stick is finally ready for distribution**.

Of course, you can let end users register your publication themselves, you can replace registration keys with online activation or use different certificates... HTML Executable is versatile on this point and allows a lot of possibilities thanks to certificates and security profiles.

4. Learn About All Features

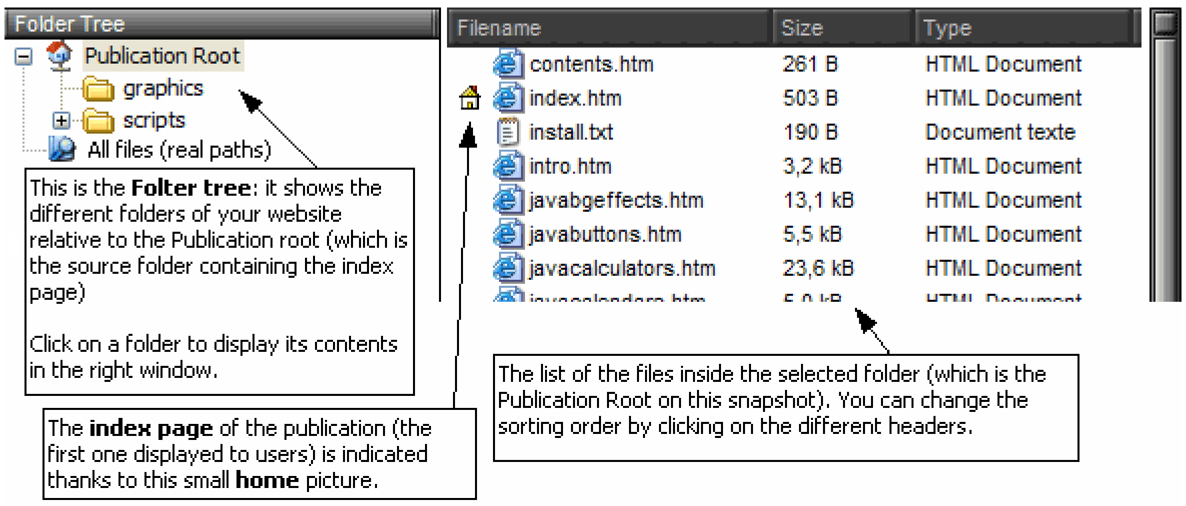
4.1. File Manager

You can access the File Manager by selecting the "**File Manager**" ribbon at the top.



It lets you manage all of the files to be **compiled** into your publication; it **behaves like Windows Explorer**.

How it works



Filename	Size	Type
contents.htm	261 B	HTML Document
index.htm	503 B	HTML Document
install.txt	190 B	Document texte
intro.htm	3,2 kB	HTML Document
javabgeffects.htm	13,1 kB	HTML Document
javabuttons.htm	5,5 kB	HTML Document
javacalculators.htm	23,6 kB	HTML Document

This is the **Folter tree**: it shows the different folders of your website relative to the Publication root (which is the source folder containing the index page)

Click on a folder to display its contents in the right window.

The **index page** of the publication (the first one displayed to users) is indicated thanks to this small **home** picture.

The list of the files inside the selected folder (which is the Publication Root on this snapshot). You can change the sorting order by clicking on the different headers.

You have a **global view** of what will be inside your publication. The Folder tree displays the different sub-folders available - as if you were using an **FTP program** to manage files on a server. The "**Publication Root**" folder represents the root of your website (this is the source folder).

To **view the contents** of a folder, just select it in the Folder tree and its contents will be listed. Note that HTML Executable can take some seconds for loading the entire file list when you display it for the first time (if your sub-folder contains a large amount of files).

The **index page** is always indicated thanks to a small house icon (as you can see on the screenshot).

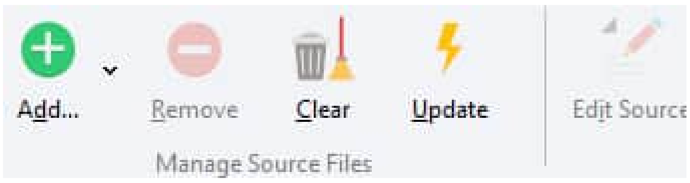
If you select the "**All Files (real paths)**" node in the Folder tree, HTML Executable will list all files to be compiled with their full path.

If you select the "**All Files (virtual paths)**" node in the Folder tree, HTML Executable will list all files to be compiled with their virtual path (i.e. the path that follows the server or domain name in a URL).

Managing files

All source files are generally added automatically when HTML Executable creates your project. But you can at any time add new files, folders or source wildcards to your list manually; or better **let HTML Executable do this task itself** thanks to the powerful Source File List Update option.

You can manage your files thanks to the button bar below or using the **mouse's context menu** (right click on the file list).



- Adding Files
- Source File List Update Operation

Removing files

To remove one or more files from the list (or an empty folder), select it/them and press the **Remove** button.

You can select multiple files by pressing Ctrl+Click or Shift+Click. You may also use the **Select All** menu command from the context menu.

Finally, the **Clear** button lets you remove the entire folder currently selected in the Folder Tree. This includes all files and subfolders which are children of that selected folder. This operation cannot be undone, so be careful!

Notes:

- you cannot remove the index page from the publication.
- you cannot clear the publication root: use the Remove operation instead.
- removing files does **not** delete files from your hard disk. This only "removes" files from the publication.

Sorting Files

HTML Executable displays information about each file: filename (or file path/virtual path), file type, size... If you need to sort your files according to a specific value (for example, file types), just click on the associated column. You may also resize the different columns.

Drag/drop operations

HTML Executable supports drag/drop operations:

- from **Windows Explorer** and other shell windows. Select all the files in the Explorer, drag them onto the manager's file list and they are automatically added to the list. You can be prompted to enter the virtual path for the dropped files as explained above.
- in the **File Manager itself**: you can move files from a virtual folder to another (thus their virtual paths can be changed the way you want). Highlight the file(s) you want to move in the list, then drag and drop them onto the destination folder in the Folder Tree. Note that real paths are not modified (source files are not physically moved).

Editing files

Select a file in the list and click the **Edit** button.
If the selected file is an HTML-compatible page, then the internal HTML editor will be displayed.
Otherwise the program registered with the file will be opened to edit it (this is exactly as if you were double-clicking the file in Windows Explorer).

More information about the internal HTML editor

Setting file and HTML page properties

You can configure some options regarding compression, security for source files. Select a file in the list and click the **Properties** button (or double-click on it). The file properties editor will then be displayed.

More information about the file properties

Using the context menu

All of the previous commands are also available from the context menu: click on the list with the mouse's **right button** to display it. It contains additional commands not directly available such as "Select All" or "File Information". The last command gives you global information about the publication files of the selected folder (total size, number of files...). Finally, the "Shell Properties" command will display the "Properties" dialog available in Windows Explorer (lets you access to the properties of the selected file).

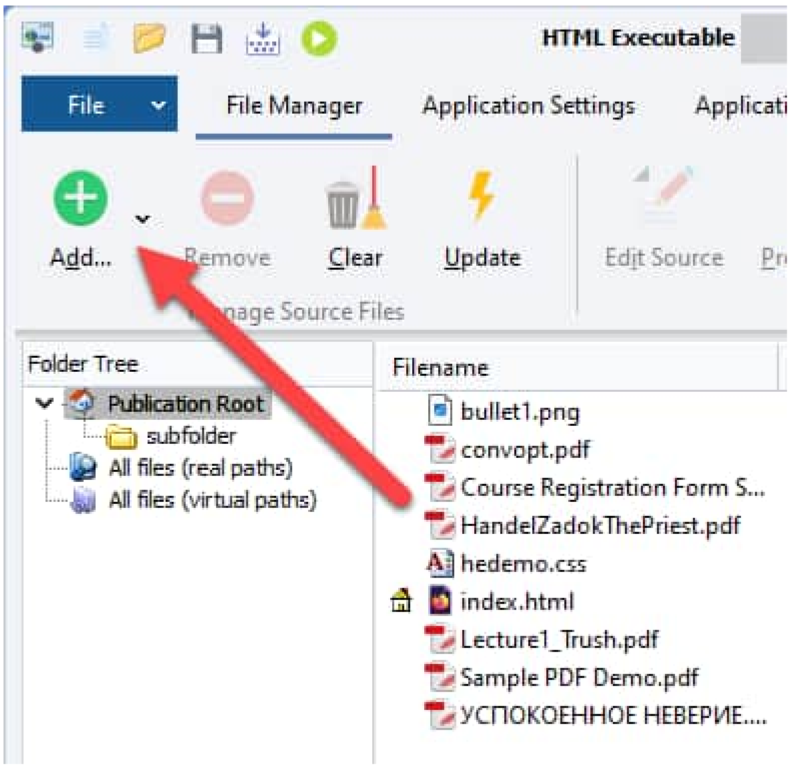
Encrypt External Files

HTML Executable lets you use external files (files that remain outside the ebook or publication .EXE file) and encrypt them: refer to the Encrypt External Files topic.

4.1.1. Adding files manually to the ebook

Remember that HTML Executable adds files from the project's source folder itself thanks to the powerful Source File List Update option. But you can still add files to your publication from other locations.

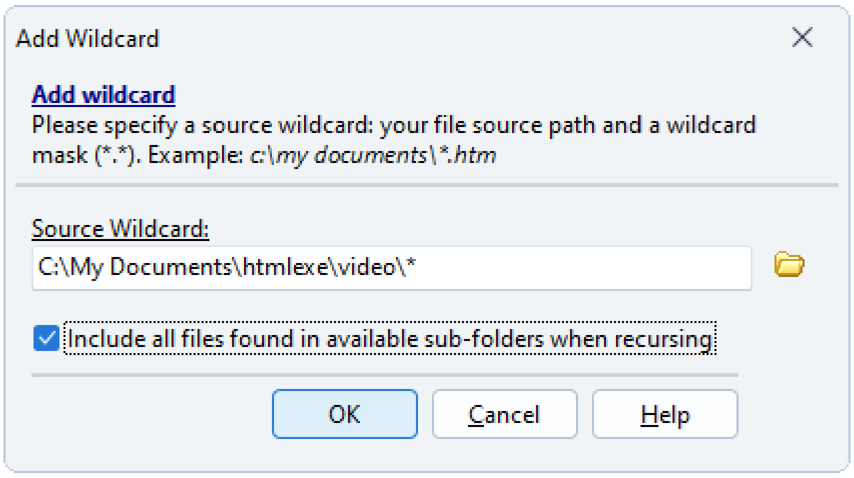
In the File Manager, click **Add**:



Then select an option: you may add single or multiple files, entire folders, virtual folders and custom wild cards. Just select the appropriate action when you have just clicked the Add button:

- Add Files
- Add Folder
- Add Source Wildcard
- Add Virtual Folder

For example, selecting "Add Wildcard" will open this window:


Add Wildcard

Add wildcard

Please specify a source wildcard: your file source path and a wildcard mask (*.*). Example: *c:\my documents*.htm*

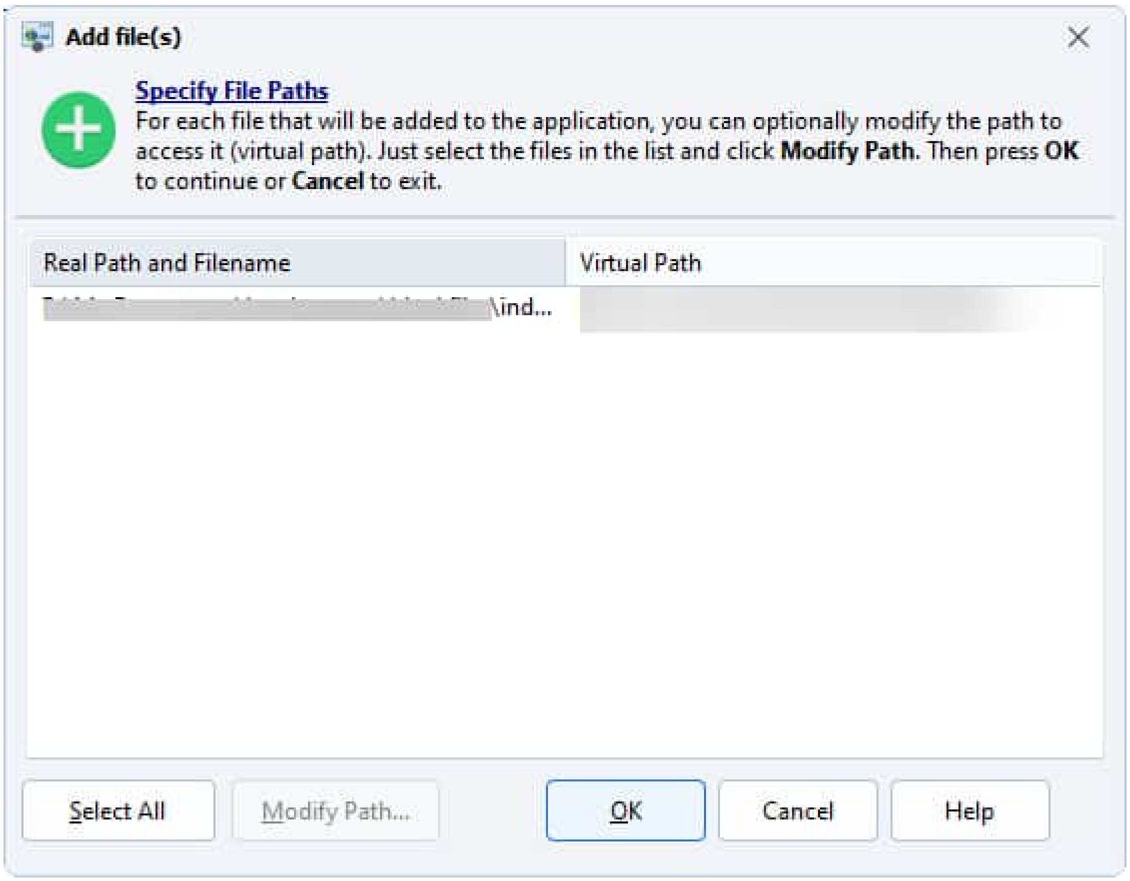
Source Wildcard: C:\My Documents\htmlexe\video\*

[✓] Include all files found in available sub-folders when recursing

You can enter a path and specify which files should be added using a mask: in the situation shown on the screenshot only .html files from c: my documents heexample will be added.

Or you may rather add entire folders (including their sub-folders or not) using the "Add Folder" command.

When added files are not in the source folder or one of its children, you need to specify the **publication path** (called virtual path) which should be used to access these files once they are compiled in the publication. HTML Executable lets you determine these "virtual" directories thanks to the following window (which is automatically shown when adding files):


Add file(s)

Specify File Paths

For each file that will be added to the application, you can optionally modify the path to access it (virtual path). Just select the files in the list and click **Modify Path.** Then press **OK** to continue or **Cancel** to exit.

Real Path and Filename	Virtual Path
[redacted]\ind...	[redacted]

The real paths appear on the left; on the right you can determine the related virtual path (a.k.a publication path).

By default, HTML Executable automatically tries to find the best virtual paths. But you can change this yourself: just select one or more files, then click "**Modify Path**". You will be prompted to enter the new virtual path and confirm your changes.

Once you have clicked OK, HTML Executable adds them to the publication's file list.

> ⓘ This option is powerful because you can actually add files from any folder and determine virtual paths yourself. The main advantage is that you do not need to copy the different source files to the source folder first in order to have them be compiled.

4.1.2. External Files

Large source files such as audio and video files may be **kept outside of the ebook or HTML application** because Windows handles EXE filesizes up to 4 GB only.

Moreover, HTML Executable allows you to **encrypt external files** so that they can only be opened/viewed by the publication.

Learn more about encrypting external files.

External files **must be deployed** with the ebook .exe file, so this option is especially useful if you plan to distribute your publication on standard media supports like CDs, DVDs...

To set a file as external, select it in the File Manager and click Properties. The "File Properties" window is displayed; turn on "**Keep the selected file(s) external**". You are then prompted to enter the **future path to the external file**.

Accessing external files from the publication

Your application can automatically load **external resource files** (not compiled into the EXE file): for instance, you can **have image and media files outside the EXE file** (in the same folder or a subfolder).

Examples:

> If an HTML file references image1.png, the application will look for the image1.png file into its compiled data; if it is not found, it will try to locate it in the same folder as the EXE file (depending on the URI) and load it.
> If you have image file in a subfolder, e.g. `<img src="myfolder/my image.png" />`, the application will expect the `my image.png` file to be in a subfolder named `myfolder` (if you leave the file external).

External files have to be deployed with the application's EXE file, in their respective folders.

Path to the folder where the file will be available

If you do not place external files in their expected folders as explained above, you can tell the ebook's application where each file is located. In this field, you can enter the **full path** to the external file. Since you can't always guess the path of the latter on the end user's computer, HTML Executable provides you with a **%PATH%** variable. This variable denotes the path to the publication .exe file. Therefore you can now tell the runtime viewer the external file's relative path.

- the external file will be in the same folder as the publication .exe file: just enter %PATH%
- the external file will be in a subfolder (say "MyFolder1"): enter %PATH%\MyFolder1

The %PATH% variable will never include the final path backslash so do not forget to insert it for subfolders: %PATH%\Subfolder is correct while %PATH%Subfolder is wrong.
Note that you can also have several subfolders: %PATH%\MyFolder1\MyFolder2\MyFolder3

Example

Take this set of files; `[root]` is the path to the root folder that contains the publication. It can have any values.

`[root]\MYPUB.EXE` is the publication .exe file built by HTML Executable. It will require the five following external files:

> [root]\docs\INFO1.PDF
> [root]\docs\INFO2.PDF
> [root]\MAP.PDF
> [root]\video\codec1\VIDEO1.AVI
> [root]\video\codec1\VIDEO2.AVI

You could imagine that `[root]` will be the root of a CD that you will distribute. You cannot guess the letter to the CD drive on the user computer, so you will need to use the %PATH% variable and enter the following values in the full path's field:

[root]\docs\INFO1.PDF -> %PATH%\docs [root]\docs\INFO2.PDF -> %PATH%\docs
[root]\MAP.PDF -> %PATH%
[root]\video\codec1\VIDEO1.AVI -> %PATH%\video\codec1
[root]\video\codec1\VIDEO2.AVI -> %PATH%\video\codec1

Notes

- Do not insert filenames in the full path field: `%PATH%\docs` is correct, `%PATH%\docs\INFO1.PDF` is wrong. HTML Executable will automatically exclude external files from compilation, but they will remain in the file list.
- Only media and very large files should be kept external: for instance you should not use this feature for HTML pages.

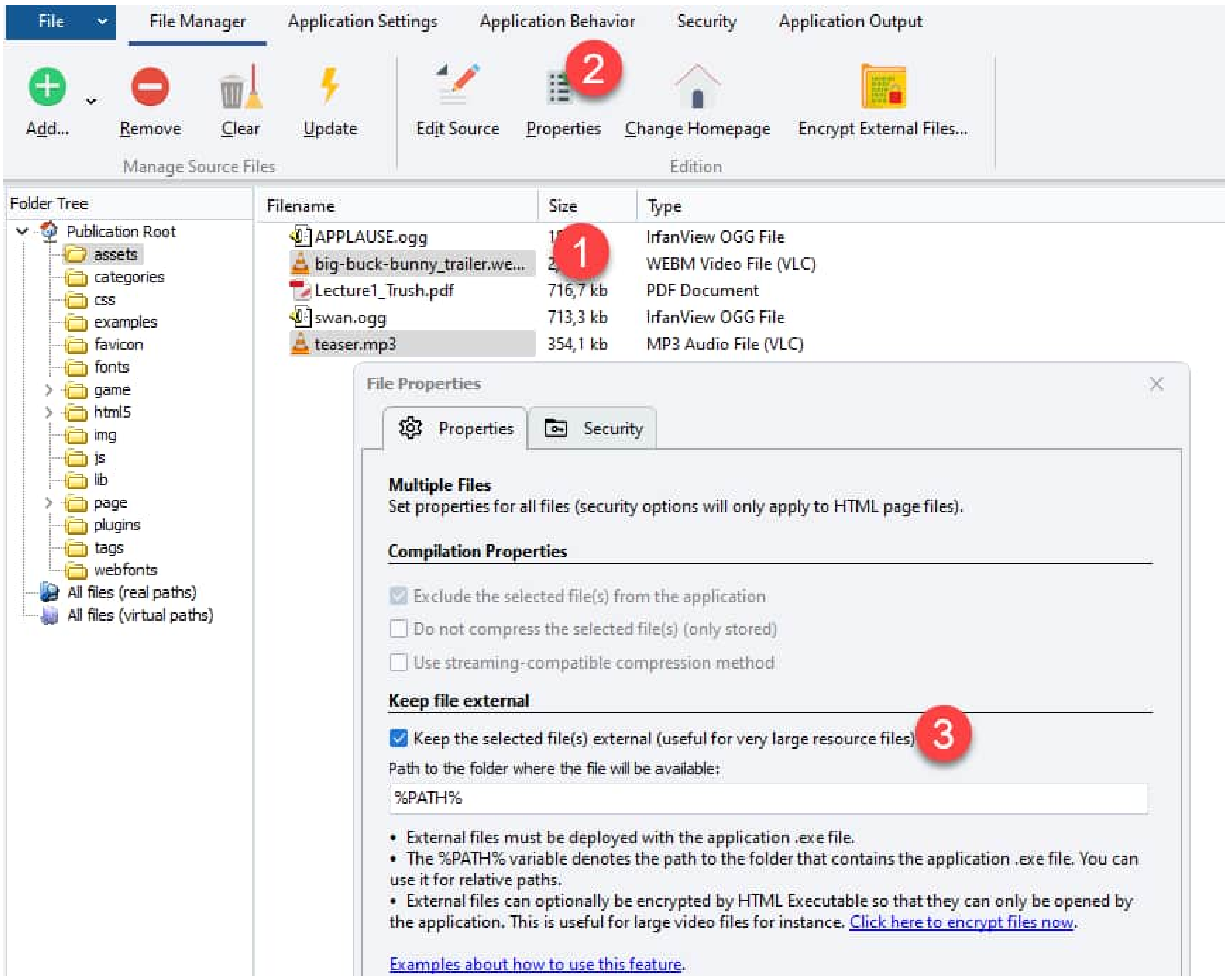
See also Encrypt External Files.

4.1.3. Encrypt External Files

HTML Executable allows you to provide enhanced security to your ebooks or publications by **encrypting external files**. This feature is particularly beneficial for large source files like audio and video, which you may want to keep outside the ebook application / publication yet protected against copying. HTML Executable ensures that your encrypted files can only be accessed by your HTML application, thereby preventing unwanted duplication of your resources.

Setting files as external files

In the `File Manager`, choose the files you want to keep outside the HTML application's EXE file. They will be considered as "External Files".



Filename	Size	Type
APPLAUSE.ogg	1	IrfanView OGG File
big-buck-bunny_trailer.we...	2	WEBM Video File (VLC)
Lecture1_Trush.pdf	716,7 kb	PDF Document
swan.ogg	713,3 kb	IrfanView OGG File
teaser.mp3	354,1 kb	MP3 Audio File (VLC)

Multiple Files
Set properties for all files (security options will only apply to HTML page files).

Compilation Properties

- [✓] Exclude the selected file(s) from the application
- [ ] Do not compress the selected file(s) (only stored)
- [ ] Use streaming-compatible compression method

Keep file external

- [✓] Keep the selected file(s) external (useful for very large resource files)

Path to the folder where the file will be available:
%PATH%

- External files must be deployed with the application .exe file.
- The %PATH% variable denotes the path to the folder that contains the application .exe file. You can use it for relative paths.
- External files can optionally be encrypted by HTML Executable so that they can only be opened by the application. This is useful for large video files for instance. Click here to encrypt files now.

Examples about how to use this feature.

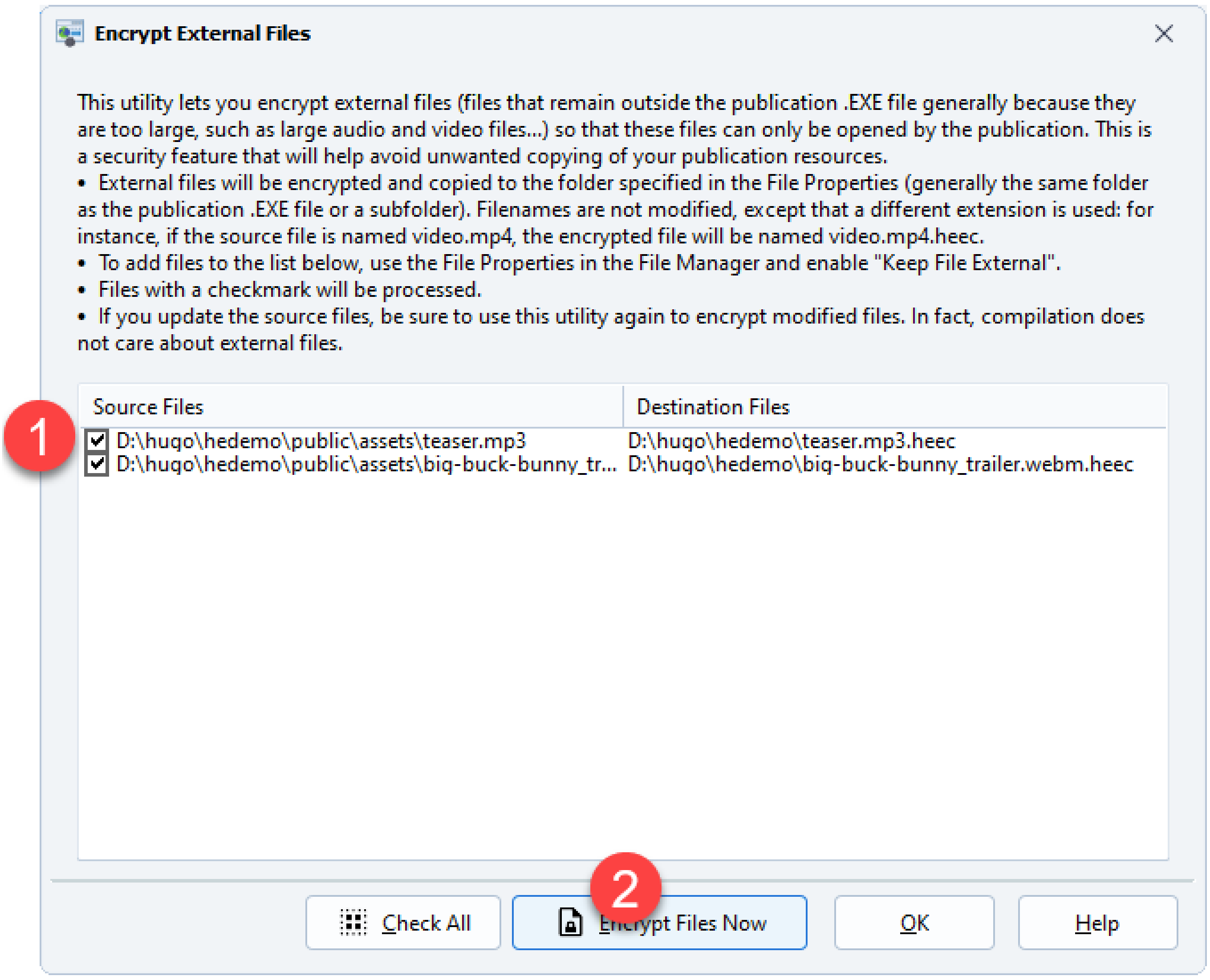
Encrypting Your External Files

To encrypt your external files, follow the steps below:

Go to the `File Manager` and click on `Encrypt External Files`. You don't need to select any specific file- HTML Executable will automatically list all external files for you.

> **Note:** External files are not automatically encrypted during the publication compilation because they are not included in the compilation process. Hence, you need to manually encrypt them as explained.

- The `Encrypt External Files` utility will display all the external files listed in the `Source Files` column.
- Encrypted external files will be copied to the folder specified in the `File Properties`, usually the same folder as the publication `.EXE` file or a subfolder. The exact destination is displayed in the `Destination Files` column.
- You might not want to encrypt all files- only those marked with a checkmark will be processed. Use the `Check All` button to select or deselect all files.
- Click on `Encrypt Files Now` to start the encryption process. A progress dialog box will be displayed during this operation, which could take some time depending on the size of your source files.


Encrypt External Files

This utility lets you encrypt external files (files that remain outside the publication .EXE file generally because they are too large, such as large audio and video files...) so that these files can only be opened by the publication. This is a security feature that will help avoid unwanted copying of your publication resources.
• External files will be encrypted and copied to the folder specified in the File Properties (generally the same folder as the publication .EXE file or a subfolder). Filenames are not modified, except that a different extension is used: for instance, if the source file is named video.mp4, the encrypted file will be named video.mp4.heec.
• To add files to the list below, use the File Properties in the File Manager and enable "Keep File External".
• Files with a checkmark will be processed.
• If you update the source files, be sure to use this utility again to encrypt modified files. In fact, compilation does not care about external files.

Source Files	Destination Files
☑ D:\hugo\hedemo\public\assets\teaser.mp3	D:\hugo\hedemo\teaser.mp3.heec
☑ D:\hugo\hedemo\public\assets\big-buck-bunny_tr...	D:\hugo\hedemo\big-buck-bunny_trailer.webm.heec

Understanding Encrypted Files

All encrypted external files adopt the `**.HEEC**` extension and are uniquely associated with your publication. For instance, if you have an original external file named `video.mp4`, the encrypted version of this file would be named `video.mp4.heec`.

Updating Your Encrypted Files

Should you modify the source files, remember to use the `Encrypt External Files` utility again to re-encrypt them. As the compilation process doesn't involve external files, you must manually re-encrypt them.

⚠ **Warning:** Encrypted files cannot be shared between different publications because they each have a unique encryption key. Moreover, always keep your original source files in a safe place, as HTML Executable does not offer a decryption feature.

Accessing Encrypted Files from the Publication

Your publication can automatically load external encrypted files - it recognizes the `.HEEC` extension - in the same manner it would load normal external files. For more information on how to deploy external files with your ebook EXE file, please refer to the relevant `External Files` topic in the documentation.

Through these steps, HTML Executable offers you a robust mechanism to protect your valuable content while ensuring seamless access within your ebook.

4.1.4. Source File List Update

Generally, when you design your ebook or publication, source files may vary: you can add new files to the source folder, update existing ones or even remove non-used ones. In other words, the contents of the source folder may vary and consequently the file list kept by HTML Executable may be outdated.

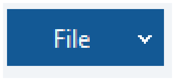
You can therefore use the **Update button:**

This action forces HTML Executable to scan the source folder (and its subfolders) and to detect all changes that happened. It then compares results and determines the appropriate actions to take:

- if new files are found (not in the file lists), they are added. You can optionally be prompted to specify the new virtual paths (see the Environment Options).
- if some source files are newer, HTML Executable updates its file lists.
- if some source files are not found (you could have deleted them), they are removed from the file lists. You can also be prompted to confirm the operation (see the Environment Options).

You consequently do not need to manage the file lists manually yourself.

There are several ways to cause a File List update:

- click the **Update** button in the File Manager.
- press **F6** or select "Edition|Automatic Source File List Update..." in the Application Menu button .
- configure automatic File List updates using the Environment Options (when a project is loaded, compiled; or enable source folder monitoring...).

Notes

- The File List Update only takes account of files in the source folder (or subfolders)! If you have files from other folders, the operation will ignore them.
- You can exclude some files from being added to the file list (based on their extension): go to the Environment Options.

4.1.5. Virtual Paths

What Are Virtual Paths?

Virtual paths are unique identifiers within a URL that follow the server or domain name. In HTML Executable, we leverage virtual paths extensively, as they mimic server-like behavior. To access a compiled file, you'll need to utilize its corresponding URL.

For example, consider the URL: `http://heserver/images/picture1.gif`

> `http://` signifies the protocol used by HTML Executable.
> `heserver` denotes the namespace of our custom server, directing you to the publication.
> `images` represents the **virtual path**.
> `picture1.gif` is the filename.

It's important to note that HTML Executable efficiently manages virtual paths on its own, and the features related to virtual paths primarily cater to advanced users.

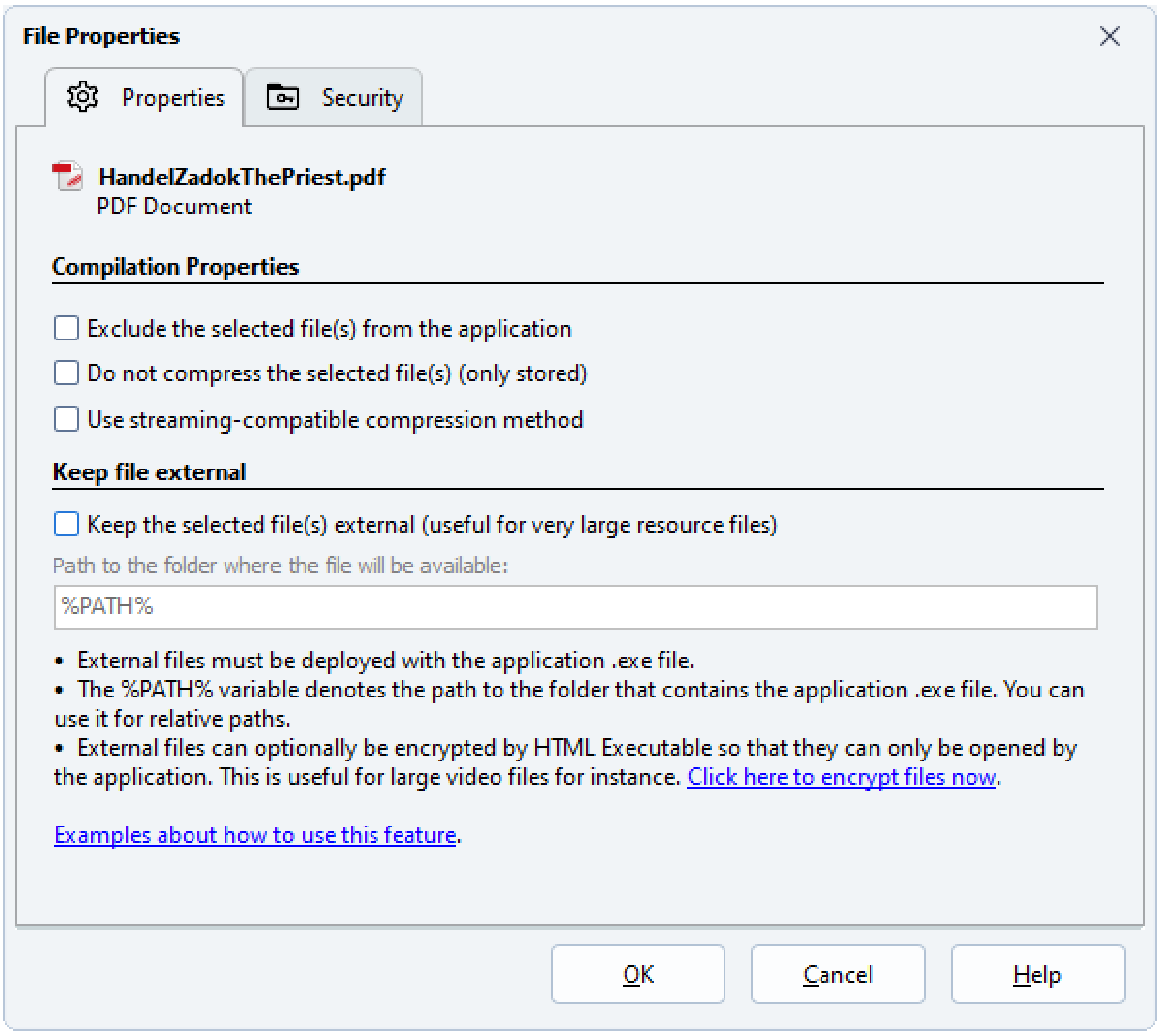
Adding Virtual Folders

Should you desire to include empty virtual folders in your publication, navigate to the File Manager, and click on **Add** followed by **Add Virtual Folder**. You will be prompted to specify its name, and it will be created as a **subfolder of the selected directory** (or as a child of the publication root if no folder is chosen). You can subsequently organize files within this folder using drag-and-drop operations.

⚠ **Important:** If you leave a blank virtual folder devoid of any files, it will be automatically removed the next time you load your project.

4.1.6. File Properties

Properties for file compression and security can be edited via the File Manager. To edit the properties of the files, select one or more files, then click **Properties** (or press Ctrl+P):


File Properties

Properties | Security

HandelZadokThePriest.pdf
PDF Document

Compilation Properties

☐ Exclude the selected file(s) from the application
☐ Do not compress the selected file(s) (only stored)
☐ Use streaming-compatible compression method

Keep file external

☐ Keep the selected file(s) external (useful for very large resource files)

Path to the folder where the file will be available:

%PATH%

- External files must be deployed with the application .exe file.
- The %PATH% variable denotes the path to the folder that contains the application .exe file. You can use it for relative paths.
- External files can optionally be encrypted by HTML Executable so that they can only be opened by the application. This is useful for large video files for instance. Click here to encrypt files now.

Examples about how to use this feature.

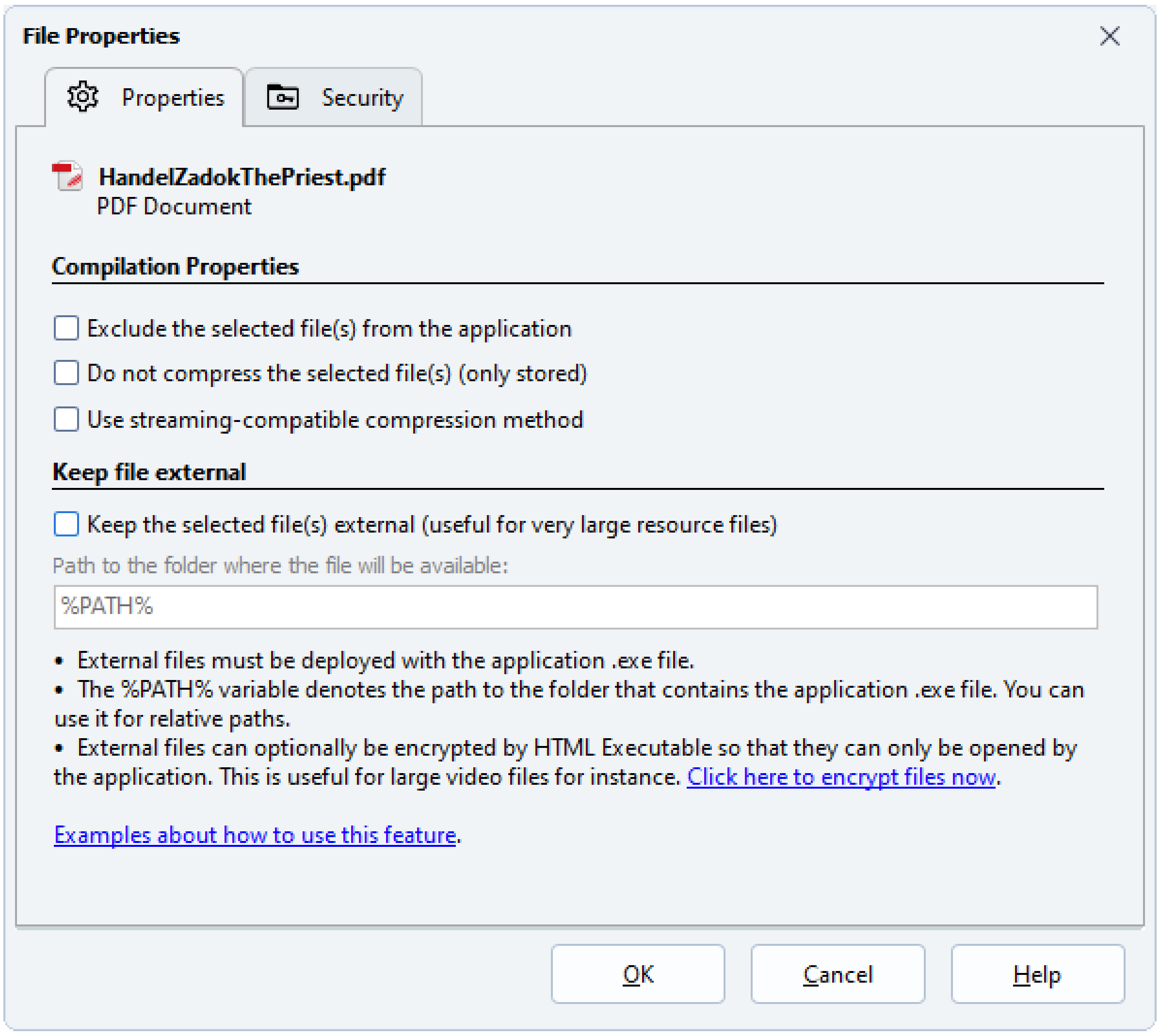
OK | Cancel | Help

If the selected file is an HTML page, you have two tabs: Properties and Security. Otherwise, you only have the Properties one.

You can see the name and the type of the selected file. If you have selected several files, HTML Executable displays "Multiple files" instead. If you leave the mouse cursor on this name, the **full path** to the file will be indicated in a small hint window.

4.1.6.1. File Properties - General

You are editing the general properties of a file that is compiled in a publication with HTML Executable.


File Properties

Properties | Security

HandelZadokThePriest.pdf
PDF Document

Compilation Properties

- [ ] Exclude the selected file(s) from the application
- [ ] Do not compress the selected file(s) (only stored)
- [ ] Use streaming-compatible compression method

Keep file external

- [ ] Keep the selected file(s) external (useful for very large resource files)

Path to the folder where the file will be available:

%PATH%

• External files must be deployed with the application .exe file.
• The %PATH% variable denotes the path to the folder that contains the application .exe file. You can use it for relative paths.
• External files can optionally be encrypted by HTML Executable so that they can only be opened by the application. This is useful for large video files for instance. Click here to encrypt files now.

Examples about how to use this feature.

OK | Cancel | Help

Compilation Properties

- **Exclude the selected file:** if you turn this option on, HTML Executable will exclude the file from the compilation; it will not be available at all.
- **Do not compress the selected file:** when enabled, HTML Executable will compile the file but it won't be compressed: it is stored - uncompressed - in the publication's data. You can use this option if you are going to include already-compressed files.
- **Use streaming-compatible compression method:** HTML Executable will use a compression method that offers on-the-fly decompression at runtime, which means that data is not unpacked to memory fully but by "chunks". This allows fast access to files and is perfect for streaming audio and video files. This method is automatically chosen when file sizes are greater than 1 MB but you can also activate it manually thanks to this option.

Keep file external

When a file was compiled in the ebook publication .exe file, it needs to be unpacked in memory in order to be viewed. This operation may require a lot of time, depending on the size of the file.

To skip the decompression step, large files like media files can be kept outside the publication. When these files need to be loaded, the runtime module will automatically look for them in the folder whose path is specified and therefore the loading time will be highly decreased.

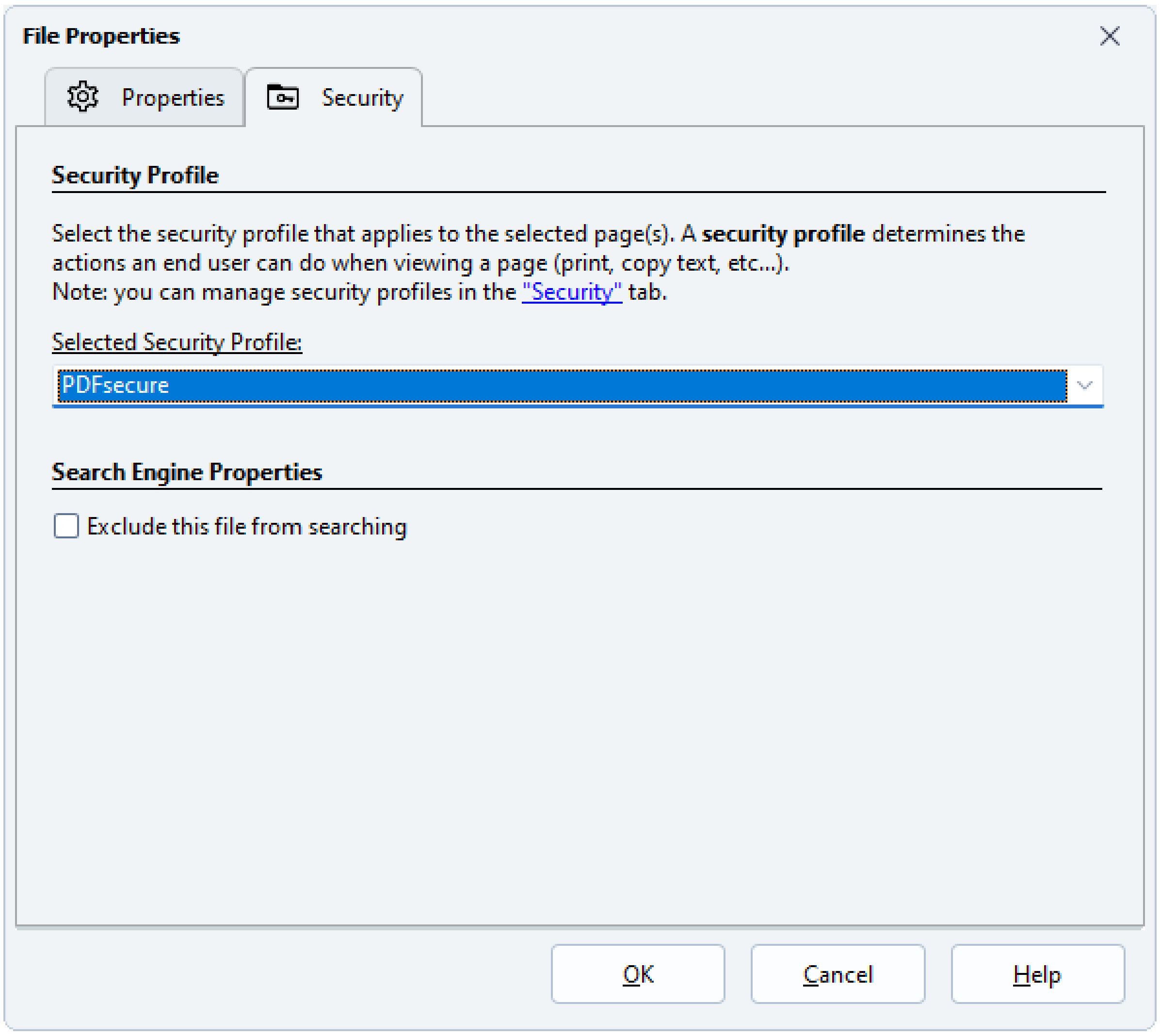
External files are not compiled and consequently **they must be deployed with your publication .exe file**. You will generally place them in the same folder as this .exe file, or a subfolder.

It is also possible to encrypt external files for improved data protection.

See how to use and link to external files

4.1.6.2. File Properties - Security

You are editing the security properties of a file that is compiled in a publication with HTML Executable.


File Properties

Properties | Security

Security Profile

Select the security profile that applies to the selected page(s). A **security profile** determines the actions an end user can do when viewing a page (print, copy text, etc...).
Note: you can manage security profiles in the "Security" tab.

Selected Security Profile: PDFsecure

Search Engine Properties

☐ Exclude this file from searching

This tab is important because it provides you with **several security features for your HTML pages, PDF and DOCX files.** You can for example restrict what end users can do: copy text, print page, etc... You may assign passwords, exclude the file from being indexed by the search engine, etc...

Security profiles may be assigned to PDF documents (only used if the built-in PDF viewer is activated) and DOCX documents (only if the built-in Word viewer is activated).

About security profiles

HTML Executable uses a system based on "security profiles" to manage the different actions allowed for HTML pages, PDF and DOCX documents.

Security profiles determine if a page can be accessed (given some conditions like password, restricted publication, script function result...) and what end users can do when viewing this page (usual actions like copying text, printing page, copying images, etc...).

See this sample about how to use a security profile to password protect pages

These security profiles are managed by you in the Security tab: you first create one or more security profiles (in addition to the default one always created when a new project is started). Then you use the File Properties to **assign your security profiles** to different HTML pages, PDF documents or DOCX documents.

Selected Security Profile

To assign a security profile to the selected HTML page(s), PDF file(s) or Flash file(s), use the list to choose the one you wish. By default, the "**Default**" security profile is always selected.

Search engine properties

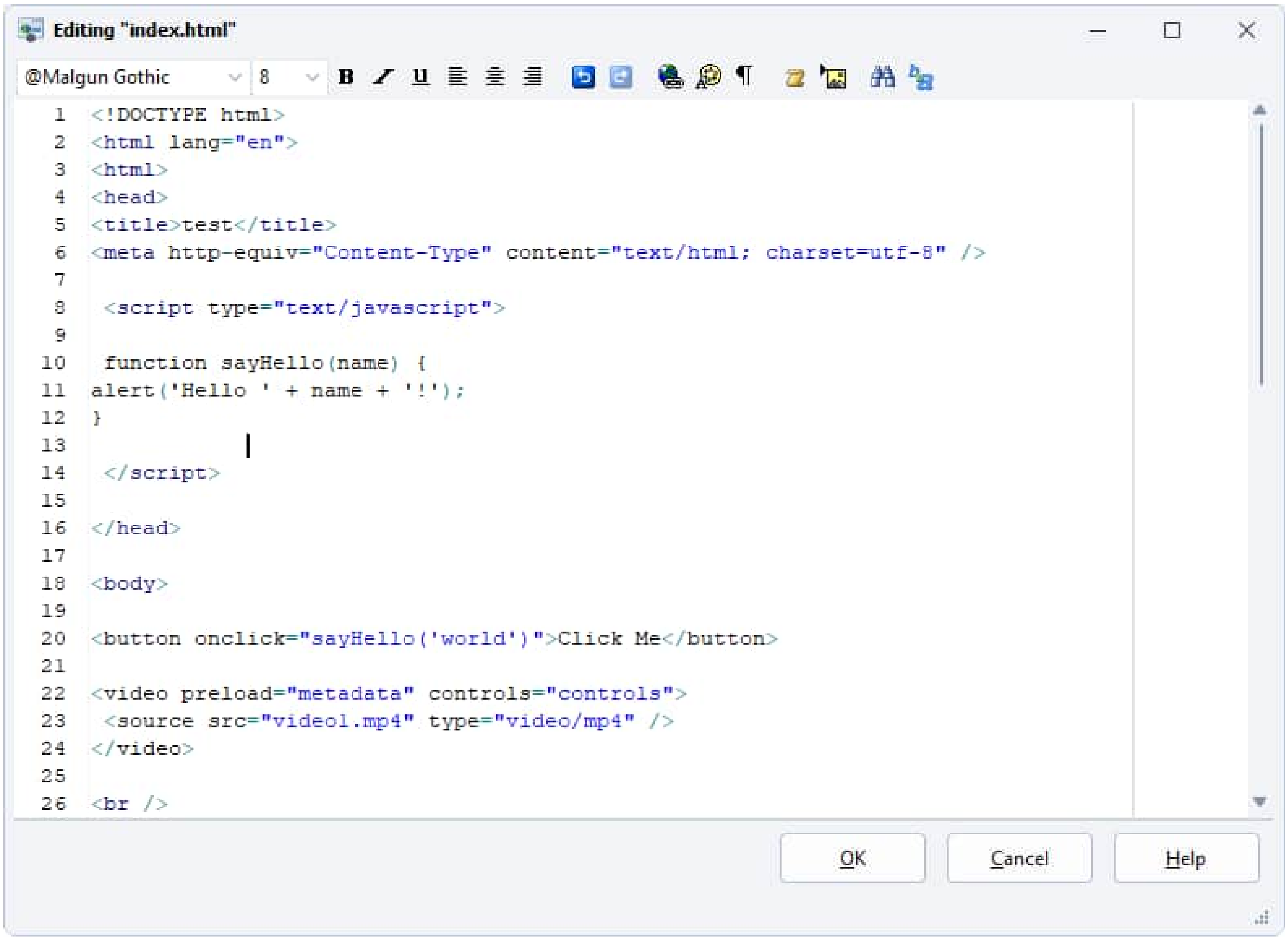
The search engine by default automatically indexes all HTML pages and PDF documents when the publication is being compiled. You can prevent the selected HTML page(s), PDF and/or DOCX file(s) from being indexed too by turning the option named "**Exclude this file from searching**" on. In this case, the document(s) will be ignored.

4.1.7. Edit HTML with Internal HTML Editor

HTML Executable features an internal HTML editor that allows you to directly edit the source code of HTML pages. This guide will walk you through the process of using this feature.

Opening the Editor

To open the HTML editor, go to the File Manager, select an HTML page, and press **Edit Source**. The editor will display the HTML code in a text box, with HTML syntax highlighted and line numbers displayed for easy navigation.



```
<!DOCTYPE html>
<html lang="en">
<html>
<head>
<title>test</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

 <script type="text/javascript">

 function sayHello(name) {
alert('Hello ' + name + '!');
}

 </script>

</head>

<body>

<button onclick="sayHello('world')">Click Me</button>

<video preload="metadata" controls="controls">
 <source src="video1.mp4" type="video/mp4" />
</video>

<br />
```

The editor also includes a toolbar at the top of the window with standard character and paragraph formatting tools, similar to those found in standard HTML editors. There are also two special buttons that provide access to special commands only available for HTML Executable publications.

Using the Editor

To modify the HTML code, simply make your changes in the text box. Be sure to press **OK** to confirm your changes; HTML Executable will save the changes back to your HTML file. A backup file with a .~HTM extension can be created for safety.

Special Commands

The HTML editor includes several special commands to enhance your editing experience:

1. **Call an HEScript function (link)**: This button is designed for advanced users working with HEScript scripts. It allows you to insert an HTML link (`<a>`) to call a procedure or a function from an HEScript script. When you click it, HTML Executable will display all of the script functions available in your project. You can then choose which one you want to insert. The result is a normal HTML hyperlink that will call the script function when end users click on it.
2. **Adding Images**: This command allows you to insert links to graphics into HTML pages. Clicking this button will display a list of available graphic compatible source files. You can select the desired graphic file and the associated HTML code will be inserted. This option can be used to add pictures to the dialog boxes if desired.

For more information about creating HEScript scripts, please go to the User Scripting tab.

4.2. Application Settings

The "Application Settings" ribbon in HTML Executable allows you to customize the GUI and navigation features of your publication.

The following topics explain these features:

- Main Window
- Rendering Engine
  - Customize the Chromium Browser options
  - Customize the WebView2 Browser options
- Skin Properties
- Components
- PDF Viewer
- Word Viewer
- Table Of Contents (TOC)
- Search Engine
- Browse Sequence

4.2.1. Main Window

A compiled ebook or publication has different windows or dialog boxes displayed to the end user. You can configure several options related to these windows.

The **main window** lets your end users navigate through your ebook, and it displays the HTML pages of your website. You can customize its appearance with the options of this tab.

Window Caption

Enter the text that should appear on the window's title bar (it only appears on the title bar: it does not affect the publication title which is displayed in the Windows taskbar or task manager). You can also display the title of the current HTML document using the **%DOCTITLE%** variable.

For example, "My first publication - %DOCTITLE%" would become "My publication - Contents Page" if the HTML page had its title set to "Contents Page".

State of the main window

Display a standard window: creates a window with the default size.
Display a maximized window: the window will be maximized, filling the screen. Some additional options for maximized windows are also available below.
Display a custom-sized window: if you wish to define the size of the window yourself, enter the default width and height in the different fields. You may also click the Auto-Sizer button: it will display a small window that you can resize as you want, then HTML Executable will automatically set the main window's

height and width according to the size you have chosen.

Allow end users to resize the window

This option determines whether end users may resize the publication's main window or not. Disabling this option is generally not recommended.

Enable Maximize Window button

This option lets you add or remove the standard Maximize button of the window. Useful if you already set the window to maximized and do not want it to be switched to the normal state.

Windows stay on top

This option indicates whether the main window remains on top of the desktop and of other windows (except ones defined with the same style in other applications).

Hide Icon

Leave this option off to display the publication's icon in the main window's title bar (allows end users access to the system menu).

Always center the window

This option centers the main window on the screen when the publication is loaded; it overrides any user-defined window position that could be saved from a previous instance.

Save user-defined window position

If turned on, the publication will save the window's size and position as it was decided by the user. The next time the publication is run, it will use the size and position it saved instead of the initial ones. This option is highly recommended. If you turned it off, the publication would always use the initial settings you set. If you wish to only save the size, then you can also turn on the "Always center the window" option.

If this option is not enabled, then the main window will appear centered in the screen.

Popups stay on top

This option makes the popup windows remain on top of the main window. If unchecked, popups appear as new windows in the taskbar and can be behind the main window.

Minimal window height/width

Lets you add resizing constraints to the main window. Leave 0 to disable these constraints.

4.2.2. Rendering Engine

This page allows you to select and configure the **HTML rendering engine** for your eBooks or applications created with HTML Executable.

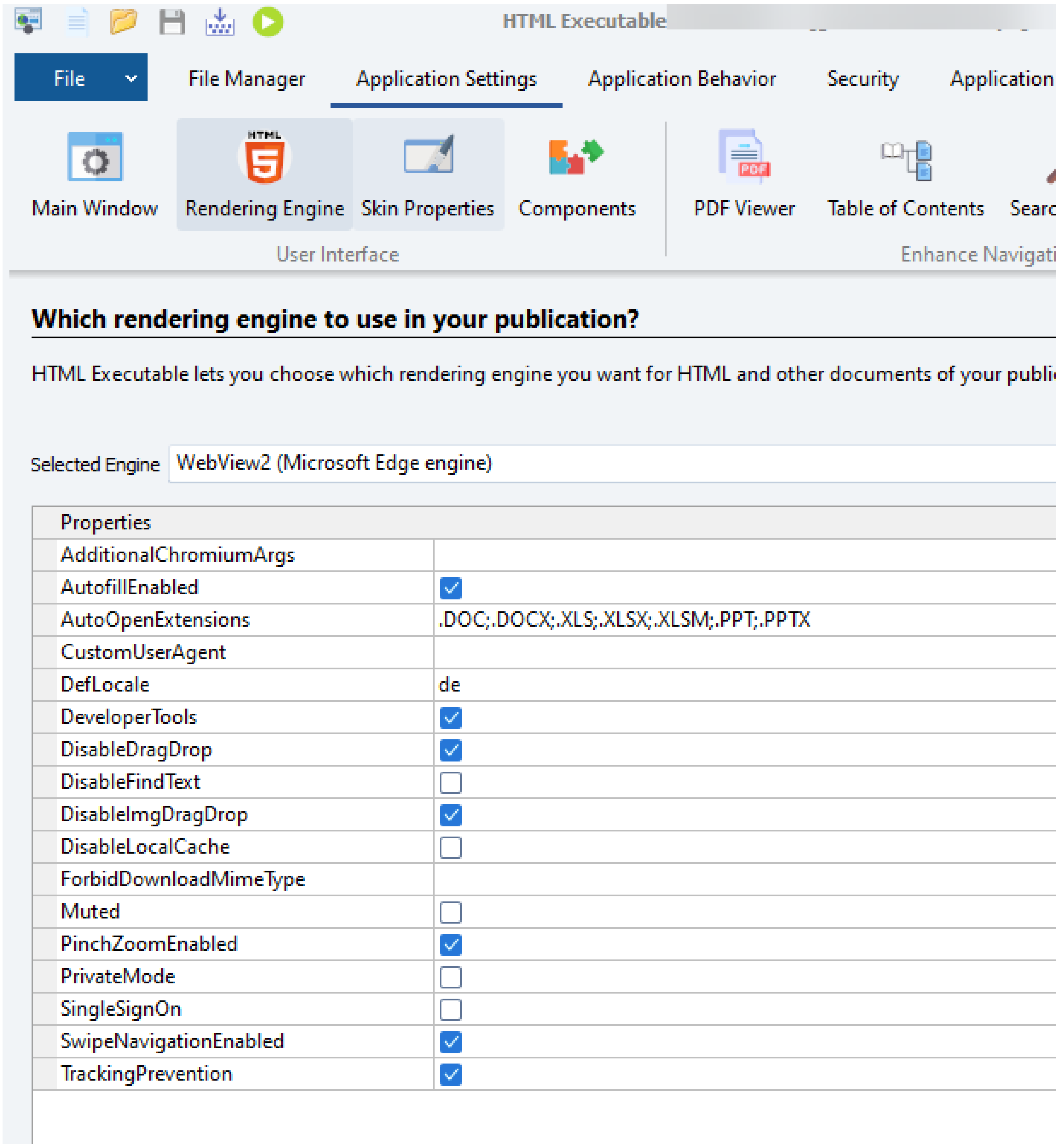
About the HTML Rendering Engines Used by Applications

HTML Executable allows end users to navigate through the pages of your eBook or application. It offers two options for the HTML rendering engine: WebView2 and Chromium Embedded Framework (CEF), both capable of rendering HTML content.

HTML Executable lets you choose between WebView2, a Microsoft Edge (Chromium) based technology, and CEF, a Chromium-based framework. The choice depends on your specific needs and the compatibility of your content.

See Choosing The Rendering Engine topic for further details about the possible choices.

Any HTML page that is compatible with Google Chrome or Microsoft Edge (Chromium) should be successfully rendered by the chosen engine, maintaining its functionality. However, unlike a web browser, the HTML and PDF source files of your eBook or application are never directly accessible by the end user. Furthermore, your eBook or application can leverage HTML5, JavaScript, CSS3, video, audio and PDF files.



Which rendering engine to use in your publication?

HTML Executable lets you choose which rendering engine you want for HTML and other documents of your publi

Selected Engine: WebView2 (Microsoft Edge engine)

Properties	
AdditionalChromiumArgs	
AutofillEnabled	✓
AutoOpenExtensions	.DOC;.DOCX;.XLS;.XLSX;.XLSM;.PPT;.PPTX
CustomUserAgent	
DefLocale	de
DeveloperTools	✓
DisableDragDrop	✓
DisableFindText	☐
DisableImgDragDrop	✓
DisableLocalCache	☐
ForbidDownloadMimeType	
Muted	☐
PinchZoomEnabled	✓
PrivateMode	☐
SingleSignOn	☐
SwipeNavigationEnabled	✓
TrackingPrevention	✓

No web server is required: HTML Executable uses its own internal protocol to display pages as if they were hosted on a remote server. Any compiled file in your eBook or application can be accessed as if it were a normal webpage.

CEF Properties

See Customize the Chromium Browser options

WebView2 Properties

See Customize the WebView2 Browser options

Selecting the CEF version to use

Unless your project requires the Flash player and you can not use the Ruffle alternative, we recommend that you **always use the latest version of the CEF renderer**. However, if you still want to use the Flash player, choose the option named "**CEF (Version 87) with old Flash support**". The software will then ask you to install the necessary files with the Web update utility if you have not already done so.

About MP4 support in CEF applications

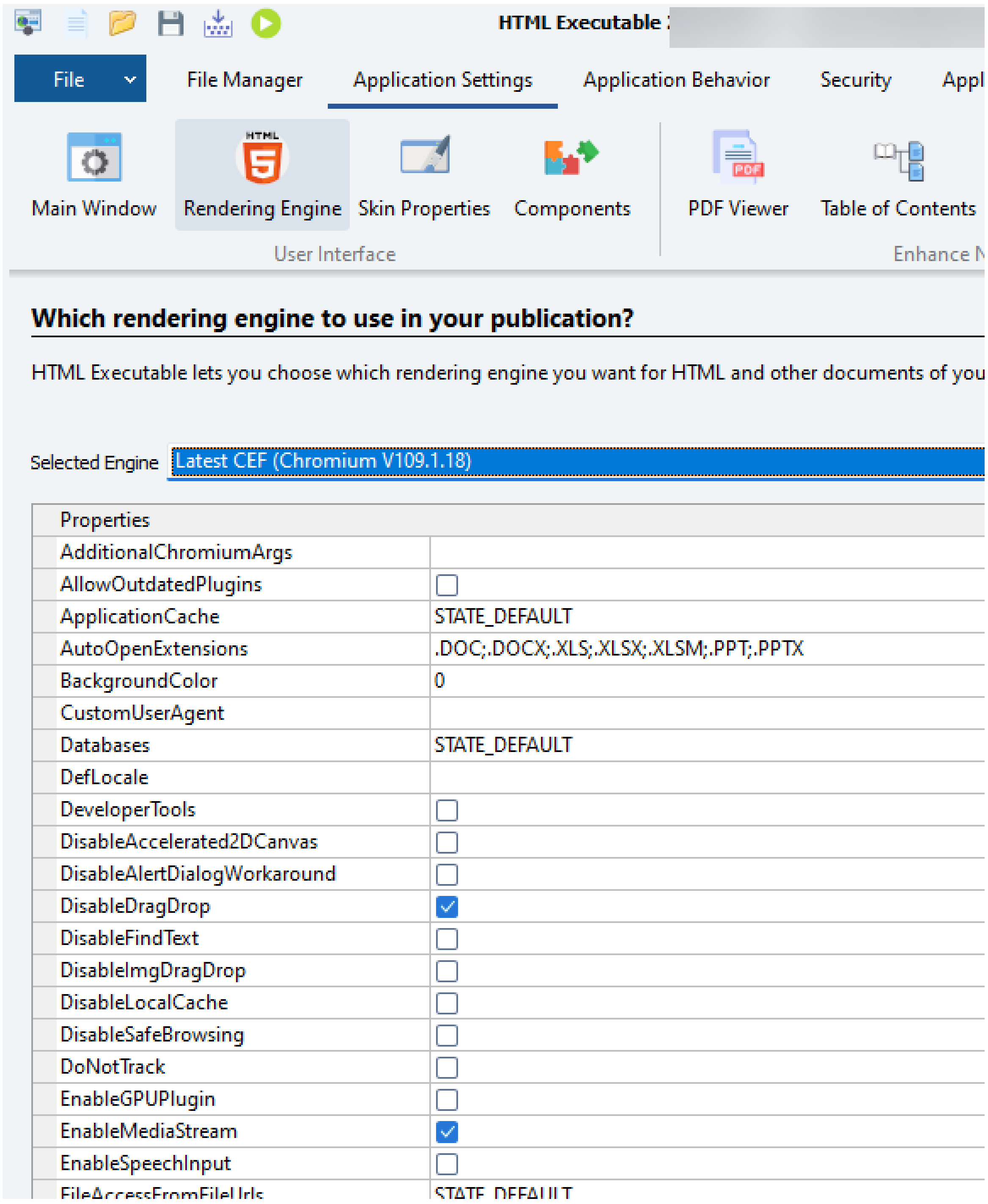
If you choose the CEF engine for your ebook or publication in HTML Executable, only open-source audio and video codecs are available: WEBM, WEBA, OGG. MP4 video format (H264 codec) is not enabled in CEF builds, because this format requires licensing.
If you want MP4 support, you can choose the WebView2 engine or use an appropriate CEF version in HTML Executable (provided that you have a valid MPEG license).

Users who would like to have the dedicated CEF build with MP4 support can contact us, but please remember that you must hold a license obtained from the Via Licensing Alliance.

4.2.2.1. Customize the Chromium Browser options

Customize the Chromium Browser options

In HTML Executable, choose Application Settings => Rendering Engine.



Which rendering engine to use in your publication?

HTML Executable lets you choose which rendering engine you want for HTML and other documents of you

Selected Engine: Latest CEF (Chromium V109.1.18)

Properties	
AdditionalChromiumArgs	
AllowOutdatedPlugins	☐
ApplicationCache	STATE_DEFAULT
AutoOpenExtensions	.DOC;.DOCX;.XLS;.XLSX;.XLSM;.PPT;.PPTX
BackgroundColor	0
CustomUserAgent	
Databases	STATE_DEFAULT
DefLocale	
DeveloperTools	☐
DisableAccelerated2DCanvas	☐
DisableAlertDialogWorkaround	☐
DisableDragDrop	✓
DisableFindText	☐
DisableImgDragDrop	☐
DisableLocalCache	☐
DisableSafeBrowsing	☐
DoNotTrack	☐
EnableGPUPlugin	☐
EnableMediaStream	✓
EnableSpeechInput	☐
FileAccessFromFileUrls	STATE_DEFAULT

A lot of Chromium options can be configured there.

AdditionalChromiumArgs

Lets you add some custom Chromium command-line arguments. For example, if you want to disable the GPU accelerated video display, add `--disable-accelerated-video`. To add multiple arguments, separate each two arguments by space.

AllowOutdatedPlugins

The Chromium engine has a security feature that does not allow users to run any outdated plugins inside the web browser. You can disable it with this property. Please note that allowing the use of outdated plugins exposes your users to security vulnerabilities of those plugins.

ApplicationCache

Controls whether the application cache can be used.

AutoOpenExtensions

Allows a compiled file within the eBook or application to be opened in its dedicated application when the associated link is visited by the end user. HTML Executable determines this behavior based on the file extension; if the file extension is included in the list for this property, then the file will be directly opened in its dedicated application.

Please note that this action temporarily extracts the file onto the user's hard drive, making it potentially vulnerable to unauthorized copying. Therefore, it is essential to consider the security aspects before using this property.

The list of extensions in the AutoOpenExtensions property should be separated by semicolons. For example, if you want to open .docx and .xlsx files in their dedicated applications, the property value should be set as .Docx;.xlsx.

The AutoOpenExtensions setting is disregarded for PDF files if the built-in PDF Viewer is enabled, and for DOCX files if the built-in Word Viewer is enabled, ensuring they open internally using the respective built-in viewers.

BackgroundColor

Opaque background color used for the browser before a document is loaded and when no document color is specified.Only the RGB components of the specified value will be used. The alpha component must greater than 0 to enable use of the background color but will be otherwise ignored.

CaretBrowsing

Controls whether the caret position will be drawn.

Custom404Error

Indicates whether the custom error page defined in Dialog Boxes should be used to display missing pages (404 file not found). If True, a 200 OK status will be returned to the web browser. If False, a 404 error will be returned to the web browser and the standard 404 error page will be shown.

CustomUserAgent

Lets you specify a custom User-Agent for the web browser.

Databases

Controls whether databases can be used.

DefLocale

Defines which locale (language) should be used by the Chromium engine (useful for localization of default buttons for instance). List of values are available in the `CEFRuntime\locales` subfolder of the HTML Executable's location.

For instance, to use the French locale, enter `fr`.

DeveloperTools

Indicates whether Chomium Developer Tools should be enabled or not. End users can access these tools with the context-menu of the internal browser.

DisableAccelerated2DCanvas

Controls whether accelerated 2D canvas can be used.

DisableAlertDialogWorkaround

Disables the status bar flicker (if the bar is hidden) related to JavaScript dialog box workaround.

DisableDragDrop

Controls whether drag/drop of external resources is allowed or not.

DisableFindText

Lets you disable the "Find Text" menu command.

DisableImgDragDrop

Lets you disable the drag/drop feature for images. For instance, if the end user drags and drops an image onto Windows Explorer, the Chromium engine will save the image as a file into the folder. It's a possible leak for your compiled image files. So, to avoid this leak, activate this property.

DisableLocalCache

Allows your application and Chromium engine to store cache and temporary files onto the end user's computer. By default, it will be a subfolder in the User Data directory as explained here.

DisableSafeBrowsing

Lets you disable the use of the Google Safe Browsing service in your application.

DoNotTrack

Lets you enable/disable the **Do Not Track** feature of the web browser.

EnableGPUPlugin

Controls whether GPUplugins can be used. Should be left off!

EnableMediaStream

Controls whether WebRTC is handled or not. Note that, for correct WebRTC support, use of the secure protocol https is recommended. Thus, do **not** enable the "Do not use secure HTTPS for internal protocol" option.

EnableSpeechInput

Controls whether speech input API can be used.

FileAccessFromFileUrls

Controls whether file URLs will have access to all URLs.

ForbidDownloadMimeType

Prevents downloads of internal resources (for instance, on default video player).

The Chromium engine allows end users to download a played video (or audio) resource using the context menu. If you cannot disable this menu from HTML, it is still possible to prevent the download of the audio or video file using this option.

For instance, if you enter `audio/mp3;audio/ogg`, HTML Executable will prevent .MP3 and .OGG files from being downloaded, but not played back by the video or audio player.

ImageLoading

Allows images or not.

ImageShrinkStandaloneToFit

How images are resized.

Javascript

Allows JS execution or not.

JavascriptAccessClipboard, JavascriptCloseWindows, JavascriptDomPaste, JavascriptOpenWindows

Customize JS permissions

LocalStorage

Allow LocalStorage or not

LogSeverity

Enable Chromium debugging log or not:

> Default logging (currently INFO logging) `LOGSEVERITY_DEFAULT`
> Verbose logging: `LOGSEVERITY_VERBOSE`
> INFO logging: `LOGSEVERITY_INFO`

WARNING logging: `LOGSEVERITY_WARNING`,
ERROR logging: `LOGSEVERITY_ERROR`,
Disables logging completely: `LOGSEVERITY_DISABLE` (default in ExeOutput for PHP)

If enabled, the Chromium log file is then stored in the same folder as the application .EXE file and with the filename: `cefdebug.log`
Ensure that the folder is writable if you want the log to be available.

MultiThreadingMode

If you're experiencing problems displaying certain resources or need to refresh a page to display all the images, for example, you can try modifying the multi-threading mode used by HTML Executable to read files in the HTML rendering engine or even disable multithreading with this property. However, this can cause your ebook application to be less responsive.

MuteAudio

If enabled, no audio will be played.

PersistSessionCookies

To allow session cookies (cookies without an expiry date or validity interval) to persist next time the application is run. Session cookies are generally intended to be transient and most Web browsers do not persist them.

Plugins

Allows Plugins or not.

SendReferrer

Should the web browser include the referrer into the HTTP headers.

TabToLinks

Controls whether the tab key can advance focus to links.

TextAreaResize

Controls whether text areas can be resized.

UniversalAccessFromFileUrls

Controls whether file URLs will have access to all URLs.

Webgl

Allows Webgl or not.

WebSecurity

Controls whether web security restrictions (same-origin policy) will be enforced. Disabling this setting is not recommend as it will allow risky/ security behavior such as cross-site scripting (XSS).

WindowlessFrameRate

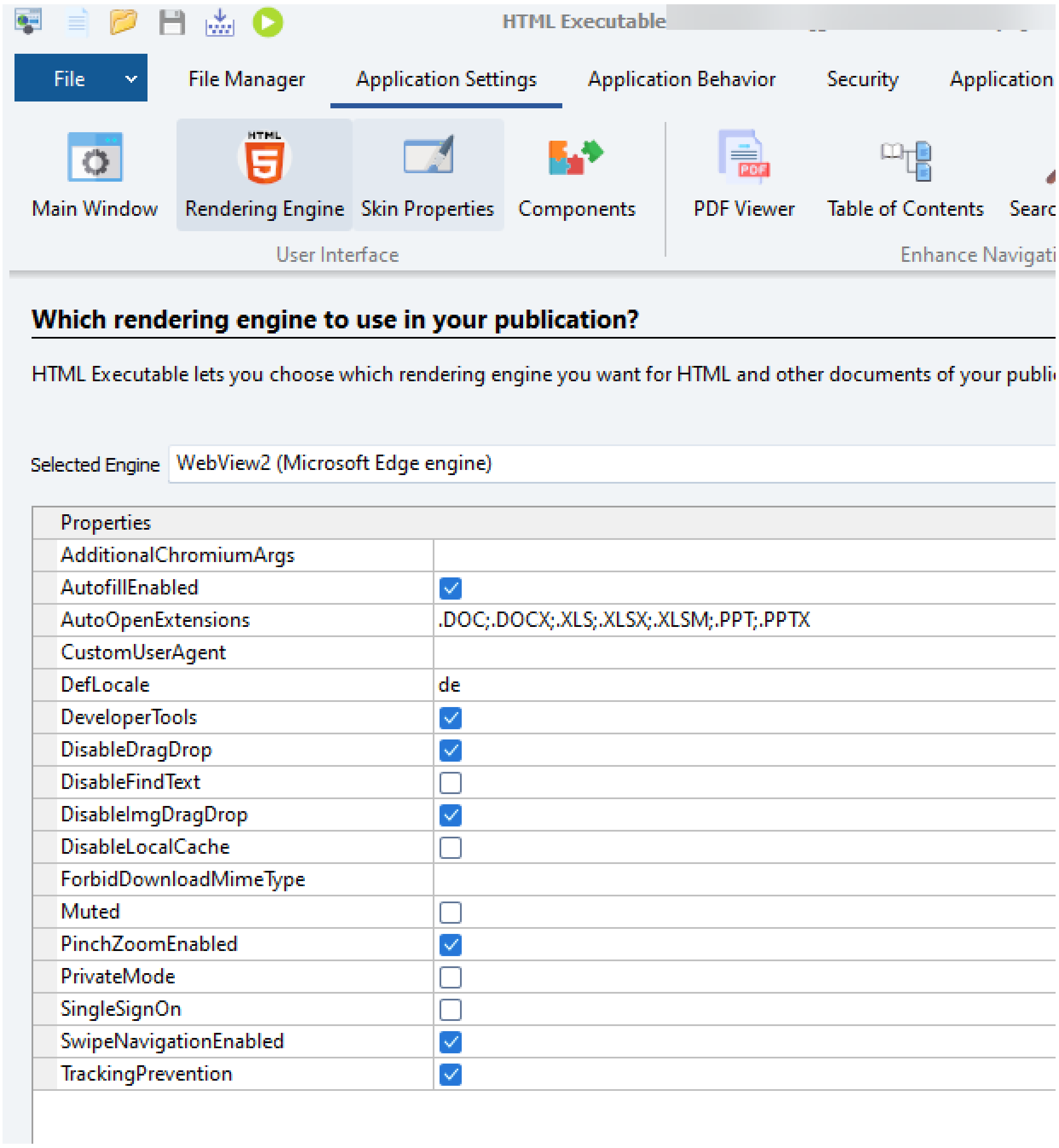
Ignored.

Please refer to the wiki of the CEF3 project for further explanation.

4.2.2.2. Customize the WebView2 Browser options

Customize the WebView2 Browser options

In HTML Executable, choose Application Settings => Rendering Engine.



Which rendering engine to use in your publication?

HTML Executable lets you choose which rendering engine you want for HTML and other documents of your publi

Selected Engine: WebView2 (Microsoft Edge engine)

Properties	
AdditionalChromiumArgs	
AutofillEnabled	✓
AutoOpenExtensions	.DOC;.DOCX;.XLS;.XLSX;.XLSM;.PPT;.PPTX
CustomUserAgent	
DefLocale	de
DeveloperTools	✓
DisableDragDrop	✓
DisableFindText	☐
DisableImgDragDrop	✓
DisableLocalCache	☐
ForbidDownloadMimeType	
Muted	☐
PinchZoomEnabled	✓
PrivateMode	☐
SingleSignOn	☐
SwipeNavigationEnabled	✓
TrackingPrevention	✓

A lot of Chromium (WebView2) options can be configured there.

AdditionalChromiumArgs

Lets you add some custom Chromium command-line arguments. For example, if you want to disable the GPU accelerated video display, add `--disable-accelerated-video`. To add multiple arguments, separate each two arguments by space.

AutofillEnabled

Determines if autofill functionality is enabled.

AutoOpenExtensions

Allows a compiled file within the eBook or application to be opened in its dedicated application when the associated link is visited by the end user. HTML Executable determines this behavior based on the file extension; if the file extension is included in the list for this property, then the file will be directly opened in its dedicated application.

Please note that this action temporarily extracts the file onto the user's hard drive, making it potentially vulnerable to unauthorized copying. Therefore, it is essential to consider the security aspects before using this property.

The list of extensions in the AutoOpenExtensions property should be separated by semicolons. For example, if you want to open .docx and .xlsx files in their dedicated applications, the property value should be set as .Docx;.xlsx.

The AutoOpenExtensions setting is disregarded for PDF files if the built-in PDF Viewer is enabled, and for DOCX files if the built-in Word Viewer is enabled, ensuring they open internally using the respective built-in viewers.

Custom404Error

Indicates whether the custom error page defined in Dialog Boxes should be used to display missing pages (404 file not found). If True, a 200 OK status will be returned to the web browser. If False, a 404 error will be returned to the web browser and the standard 404 error page will be shown.

CustomUserAgent

Lets you specify a custom User-Agent for the web browser.

DefLocale

Defines which locale (language) should be used by the Chromium engine (useful for localization of default buttons and menus for instance). List of values are available in the `CEFRuntime\locales` subfolder of the HTML Executable's location.

For instance, to use the French locale, enter `fr`.

DeveloperTools

Indicates whether Chomium Developer Tools should be enabled or not. End users can access these tools with the context-menu of the internal browser.

DisableDragDrop

Controls whether drag/drop of external resources is allowed or not.

DisableFindText

Lets you disable the "Find Text" menu command.

DisableImgDragDrop

Lets you disable the drag/drop feature for images. For instance, if the end user drags and drops an image onto Windows Explorer, the Chromium engine will save the image as a file into the folder. It's a possible leak for your compiled image files. So, to avoid this leak, activate this property.

DisableLocalCache

Allows your application and Chromium engine to store cache and temporary files onto the end user's computer. By default, it will be a subfolder in the User Data directory as explained here.

ForbidDownloadMimeType

Prevents downloads of internal resources (for instance, on default video player).

The Chromium engine allows end users to download a played video (or audio) resource using the context menu. If you cannot disable this menu from HTML, it is still possible to prevent the download of the audio or video file using this option.

For instance, if you enter `audio/mp3;audio/ogg`, HTML Executable will prevent .MP3 and .OGG files from being downloaded, but not played back by the video or audio player.

Muted

If enabled, no audio will be played.

PinchZoomEnabled

Determines if pinch to zoom functionality is enabled.

PrivateMode

Determines if private mode is enabled. Default is False.

SingleSignOn

Determines if single sign-on is enabled. Default is False.

SwipeNavigationEnabled

Determines if swipe navigation is enabled.

TrackingPrevention

Determines if tracking prevention is enabled. Default is True.

Troubleshooting CORS problems

If the console (in Developer Tools) shows navigation errors such as "Uncaught DOMException: Blocked a frame with origin "null" from accessing a cross-origin frame.", you can disable CORS checks in WebView2 by adding the **--disable-web-security** to the **AdditionalChromiumArgs** field (see above).

4.2.3. Skin Properties

About skins

One of the most interesting features of publications made with HTML Executable is their **skin support**. The entire look and feel of the application (its windows and controls) may be changed through a simple click thanks to skins.

The skin feature is similar to the Windows themes, except that HTML Executable applications use their own skin engine.

Just **select the skin you want** in the list and that's all!

A small preview of the selected skin is automatically displayed but you can get an extended preview by clicking "**Skin Full-Preview**". Note that previews do not take into account other properties.

HTML Executable has a **skin editor** (which is a separate component that you can get from your registered user account page because it is not shipped with the original distribution). You can customize existing skins or create new ones from scratch using this editor. Just click "**Open Skin Editor**" to launch it.

HTML Executable comes with several ready-to-use skins. Some skins also feature specific Aero effects (for Vista and Windows 7 only).

Note: you do not need to distribute the skin file: it is automatically compiled into your ebook's or publication's data.

Enable Dark Mode

The "Enable Dark Mode" option is a feature that allows users to switch to a darker color palette in their application's user interface. Dark mode, as it's known in Windows, changes the default bright theme to darker colors, which can be easier on the eyes in low-light environments or at night. It can also provide a different aesthetic appeal to users who prefer darker themes.

HTML Executable can simulate the dark mode by allowing you to **select a dedicated skin that will be displayed when the end user's dark mode is enabled**. Our ebook compiler comes with many ready-to-use skins that can be applied for dark mode.

Skin-related options

Create a window with no title bar

Enable this option to remove the title bar of the window. Be sure to inform your end users that they must press ALT+F4 in order to close the application.

Hide caption buttons

If you would like to remove all system buttons from the title bar (system menu, minimize, maximize, close, etc...) and keep a title bar, then turn on this option. Important: please give your users a way to properly close the application (although they can still press ALT+F4).

No skinnable window

This option disables the skin feature for the window itself; thus, you get a normal window (interesting option for Vista or Windows 7). Other controls are still rendered by the skin engine.

Do not use skins for system dialogs

Enable or not skins for standard Windows dialog boxes such as Open File or Print Options.

4.2.4. Table Of Contents (TOC)

Introduction

Adding a table of contents to your publication or ebook provides end users with a **hierarchical view of the content** (we call it a contents tree view or even TOC). Users click a topic listed in the table of contents, and are taken directly to the information they are looking for.

You can design your table of contents so that the HTML pages (or PDF documents) contained in your publication are organized by subject or by category. You can organize your subject topics in the table of contents using icons that identify main topics and subtopics. For example, if you use the default icons, HTML Executable provides a "**chapter**" (or header) icon for main topics and a "**page**" icon for subtopics. You can of course change the default icons: special icons indicating new or updated pages are available to help end users quickly scan the table of contents for new information.

When an end user opens the table of contents and clicks a topic title (or press Return), the HTML page associated with that title will open.

Updating the TOC at runtime

Configuring TOC properties

You can **modify the properties for the contents tree** thanks to the Application Settings => Components => Contents Bar object.

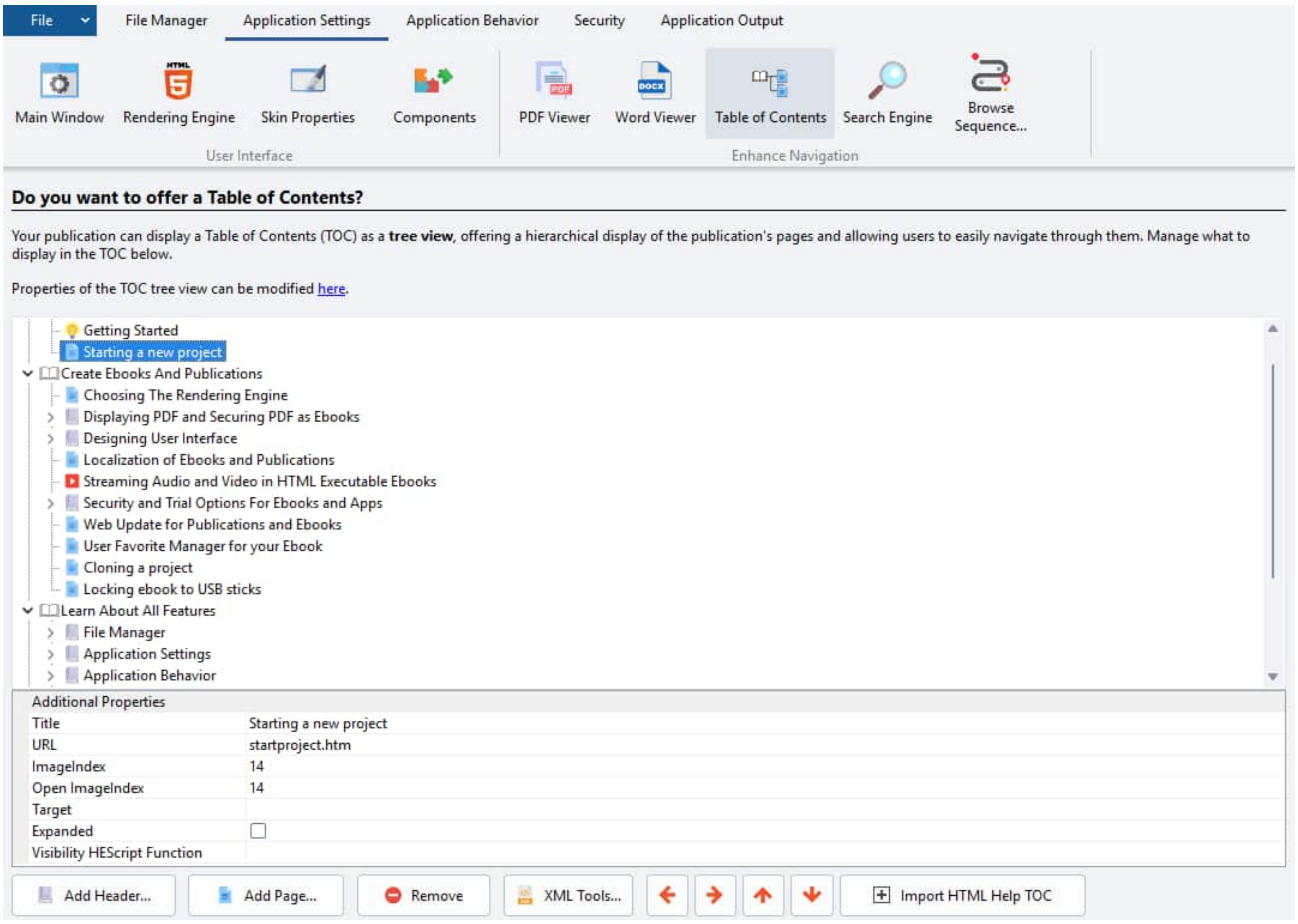
IMPORTANT: be sure that the Visible property in the user interface tab is set to **TRUE** if you want your contents tree to appear in the publication!

You may want to show all topics or show only those topics that beginning users are likely to need. When designing a table of contents, keep the following in mind:

Book icons represent headings. Double-clicking a book icon enables users to see the subentries under that heading. Subentries can be other book icons or page icons.
Page icons represent topics. Double-clicking a page icon enables users to display a topic or run a macro.

Managing TOC entries

You can manage your TOC visually using the Tree editor as shown on this snapshot:



Do you want to offer a Table of Contents?

Your publication can display a Table of Contents (TOC) as a **tree view**, offering a hierarchical display of the publication's pages and allowing users to easily navigate through them. Manage what to display in the TOC below.

Properties of the TOC tree view can be modified here.

- Getting Started
- Starting a new project
- Create Ebooks And Publications
  - Choosing The Rendering Engine
  - Displaying PDF and Securing PDF as Ebooks
  - Designing User Interface
  - Localization of Ebooks and Publications
  - Streaming Audio and Video in HTML Executable Ebooks
  - Security and Trial Options For Ebooks and Apps
  - Web Update for Publications and Ebooks
  - User Favorite Manager for your Ebook
  - Cloning a project
  - Locking ebook to USB sticks
- Learn About All Features
  - File Manager
  - Application Settings
  - Application Behavior

Additional Properties	
Title	Starting a new project
URL	startproject.htm
ImageIndex	14
Open ImageIndex	14
Target	
Expanded	☐
Visibility HEScript Function	

Add Header: This option allows you to add a new "node chapter" to the contents tree. Chapters are generally not associated with any HTML page, but you can assign HTML pages or PDFs to chapters if desired. You will be prompted to enter the name of the chapter.
Add Page: This will prompt you to select one or more HTML page(s) or PDF(s) from the publication's file list, and then the associated page node(s) will be created. The title of the node is directly read from the HTML code (`<TITLE>` tag).

Remove: Use this button to delete an existing node.
XML Tools: Click this, then select *Export* to save your Table of Contents (TOC) to an external XML file. Enter the filename when prompted. You can edit this XML file with your preferred XML editor or even Notepad, and it can be shared with different projects. To import an XML file back into the project, click *XML Tools*, select *Import*, and choose the file to import.
Up and Down buttons: Use these to move nodes within the TOC.
Parent/Child buttons ← → : If you want a node to become a child of another node, use these buttons. The left button will make the selected child node have the same parent as its own parent. The right button will make the selected node become the child of the previous one. Use these four buttons (Up, Down, Parent, Child) to arrange and reorder the nodes as desired.
Import HTML Help TOC: lets you importing an existing Table of Contents from an HTML Help Workshop Contents File (.HHC). See below.
Drag/Drop operations: You can move and exchange node positions using drag and drop. Two modes are available: right-click on the tree editor, and you can choose between "Exchange dropped and target nodes" (the node being dragged will swap places with the destination node), and "Move dropped node to new location" (the node is moved to just before the destination node).
Delete Entire TOC: To delete all nodes of the Table of Contents at once, right-click and choose the context menu command "Delete Entire TOC".

HTML and multiline support

The Table of Contents (TOC) feature in HTML Executable supports the use of HTML tags within the title elements. This means you can use tags such as `<b>` for bold text, `<i>` for italicized text, and so on, to further customize your TOC entries. Moreover, it also supports multiline title entries. However, please note that multiline support is only available if the user initially sets a greater height than the default. This can be done by going into the TOC properties and adjusting the `itemheight` property. The default value of `itemheight` is 18, so if you intend to use multiline titles, you would need to set it to at least 36 or higher, depending on your needs.

Editing TOC entries

To edit a TOC entry, just select the node you wish and its properties will be listed in the node editor (below the tree editor). There are four properties:

Title indicates the title of the entry that will appear in the contents tree.
URL defines the HTML page to be displayed when end users click on the entry in the contents tree. You can click the button on the right with '...' to select an existing page in the publication's file list. You are even allowed to use HEScript calls, external Internet links like https://www.gdgsoft.com/. URL is optional: in case you leave the field blank, nothing will happen when end users click on it (example: chapter nodes).
ImageIndex defines the icon to be displayed in the contents tree. Click on the '...' button to select an icon visually from the default icon list.
Open ImageIndex defines the icon to be displayed when the entry is selected by the user in the TOC, or when the entry has children that are visible. Generally, ImageIndex and Open ImageIndex have the same value, except for chapter nodes.
Target: optional, indicates the frame target for the TOC entry. It can be a special HTML Executable target.
Expanded: shows subentries under the heading at startup.
Visibility HEScript function: for advanced users only. Reference to a Boolean HEScript function that defines whether the TOC entry is visible or not. See dynamically updating the TOC.

Importing an Existing Table of Contents from an HHC File

HTML Executable has the ability to import an existing Table of Contents (TOC) directly from an HTML Help Workshop Contents file (.hhc). This feature is particularly useful if you are converting an existing CHM help file or migrating a project from another help authoring tool that exports to the .hhc format.

By importing an .hhc file, you can quickly populate the TOC for your publication, saving significant time compared to manually creating each entry. The hierarchical structure, titles, and page links defined in the .hhc file will be translated into HTML Executable's TOC format.

This further solidifies its role as a modern replacement for CHM.

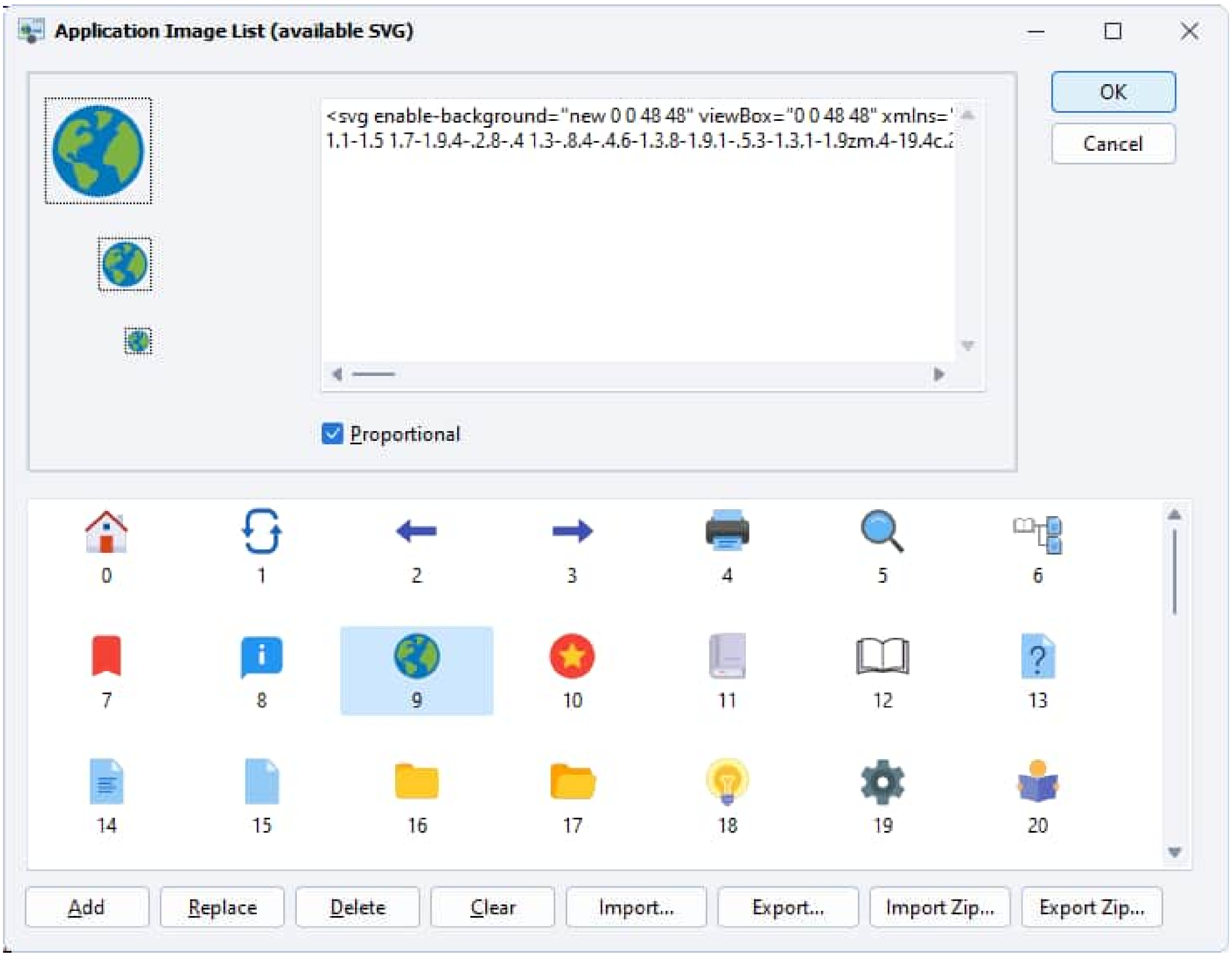
How to Import an .HHC File:

In the Table of Contents editor, click the **XML Tools** button.
From the dropdown menu, select **Import from HHC file...**.
An open file dialog will appear. Navigate to and select your .hhc file, then click **Open**.
HTML Executable will parse the .hhc file and populate the TOC editor with its contents. Existing TOC entries in your project will be replaced by the imported content.
Review the imported TOC. You may need to adjust paths or properties if the linked HTML files in your HTML Executable project have different virtual paths than those specified in the .hhc file. You can also customize icons and other properties as needed using the TOC editor.

This import feature streamlines the process of creating a functional TOC, especially for large documentation sets or when migrating existing help projects.

How to add/export custom images to the TOC

You can import custom images and associate them to the TOC entries. In the node editor, select the **ImageIndex** property and click the '...' button to open the SVG Image Manager:



```
<svg enable-background="new 0 0 48 48" viewBox="0 0 48 48" xmlns="
1.1-1.5 1.7-1.9.4-.2.8-.4 1.3-.8.4-.4.6-1.3.8-1.9.1-.5.3-1.3.1-1.9zm.4-19.4c.
```

Click "**Add**" and select the image file you want to import. It must be in SVG format.

Using it for a browse sequence

The Table of Contents can be used to define a browse sequence.

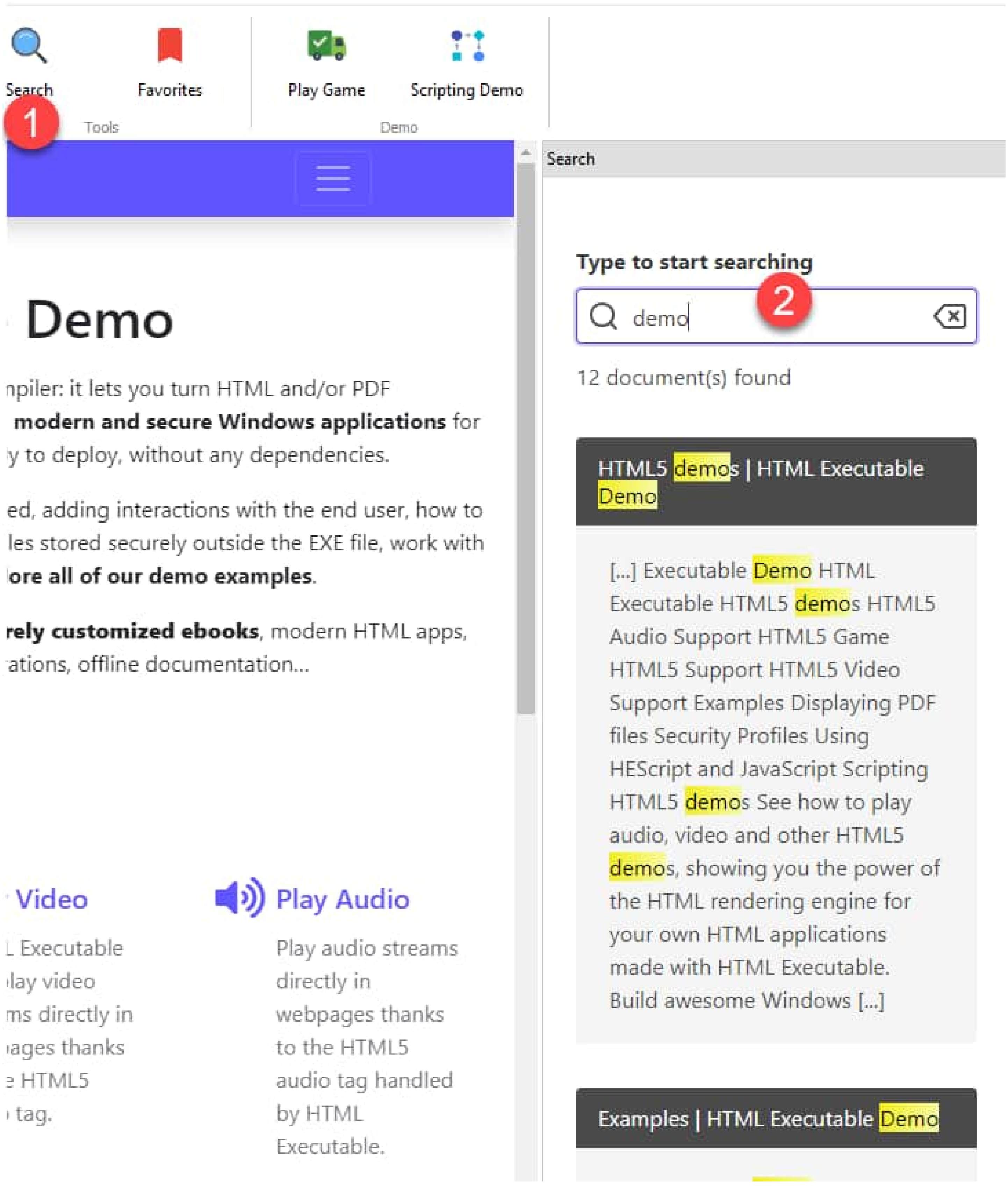
Updating the TOC at runtime

You can modify the TOC at runtime thanks to HEScript: see the Dynamically updating the TOC page.

4.2.5. Search Engine

About the built-in search engine

Publications or ebooks created with HTML Executable come with a built-in search engine that allows end users to **search for specific words or expressions** through the entire publication in seconds. When compiling your publication or ebook, HTML Executable parses all HTML pages, PDF documents and DOCX documents, collecting keywords from them. These keywords are then indexed and the result is stored in the publication's data. Since keywords are indexed, it only takes seconds for a search query to be completed.



12 document(s) found

HTML5 demos | HTML Executable Demo

[...] Executable Demo HTML Executable HTML5 demos HTML5 Audio Support HTML5 Game HTML5 Support HTML5 Video Support Examples Displaying PDF files Security Profiles Using HEScript and JavaScript Scripting HTML5 demos See how to play audio, video and other HTML5 demos, showing you the power of the HTML rendering engine for your own HTML applications made with HTML Executable. Build awesome Windows [...]

Examples | HTML Executable Demo

HTML Executable indicates the number of pages and unique words that were found while indexing pages in the compilation log.

End users can access the search panel by clicking the "**Search**" button or by selecting "Navigate|Show Search" in the application.

Configuring the search engine

Enabling the search engine can result in a **larger EXE output file** (it depends on the number of HTML pages, PDF and DOCX documents you compile). If you, therefore, do not want to include a search engine in your publication, then turn the following option on: "**Disable the search engine**".

PDF documents can be indexed too, even if the built-in PDF viewer is deactivated.

When a search is complete, the application lists the results. Each result displays the page's title and an extract with the found keyword(s) or expression(s).

Search results are **automatically sorted by relevance**. To do this, HTML Executable counts the number of occurrences of the search terms in each page, then assigns a percentage of relevance.

Some keywords may be automatically **excluded** from the index so they won't give any result if end users search for them. In addition to some common words, you may add your own sensitive keywords to the exclusion list. Just press **Add** and specify the keyword to add. On the contrary, you can remove keywords from the exclusion list by selecting them and clicking **Remove**. Keyword exclusion lists may be imported/exported from/to XML files using the **XML Tools** button, so you can edit them manually using any XML editor.

Language Support

The search engine supports multiple languages, enhancing the search experience for various kinds of content. You are required to specify the language of your HTML and PDF files to optimize the search engine for that particular language. Especially, certain commonly used keywords in the language, known as "stop words", are excluded by default from the search. These words are typically short, functional words such as "and", "the", and "of" in English, which are often filtered out as they occur so frequently that including them could skew the relevance of the search results. These stop words are defined in a supplementary JavaScript file to lunr.js and are automatically selected based on the user's chosen language. This approach ensures more accurate and language-specific search functionality in the compiled eBooks or applications.

For advanced users, the JavaScript language files are available here if you wish to view their contents:

C:\Program Files (x86)\HTML Executable 2023\Resources\JavaScript

Indexing content available only inside the following HTML tag

HTML Executable's search engine functionality allows you to index the content of HTML pages specifically within certain tags. By default, it indexes the content within the 'body' tag. However, if you're using a template with various frames where the same words appear on all pages of your website, this can skew the search results. To address this, you are encouraged to specify the name of the tag that contains unique content on each page. For instance, suppose your website contains content within a 'div' HTML tag with the ID 'content'. In this case, you would input `<div id="content"` into HTML Executable. This way, HTML Executable will only index the content enclosed between the `<div id="content">` tag and its corresponding closing `</div>` tag. This feature provides a more accurate and focused search functionality by indexing only the relevant content on each page.

Support for Unicode

The search engine is **Unicode-enabled**. When parsing HTML pages, HTML Executable takes account of the encoding format and the charset defined in HTML documents. All keywords are natively converted and stored in

UTF-8 format.

If no charset is defined in an HTML file, you can specify the **default HTML charset** that should be used (by default, UTF-8).

Indexing DOCX Document Content

HTML Executable can index the textual content of your DOCX files, making them searchable via the publication's search engine. To enable this, go to **Application Settings => Word Viewer** and check the option **Index content from Word files with the search engine**. Specific DOCX files can be excluded from the search using File Properties - Security.

About searches

HTML Executable uses lunr.js as its search engine. According to its documentation:

At the most basic level, search queries can consist of a single term, like 'hello'. However, they can also include multiple terms, which are joined with an OR operator. For example, the query 'hello world' will retrieve documents containing either 'hello' or 'world', but documents containing both will rank higher.

You can add wildcards to terms to represent one or more unspecified characters. These wildcards can be positioned anywhere within the term and a term can contain more than one wildcard. While this broadens the range of documents found, it can negatively affect query performance, especially when a wildcard is placed at the beginning of a term.

By default, when the end user types a query, HTML Executable immediately starts searching for it (a wildcard is always added to the end of the query entered).

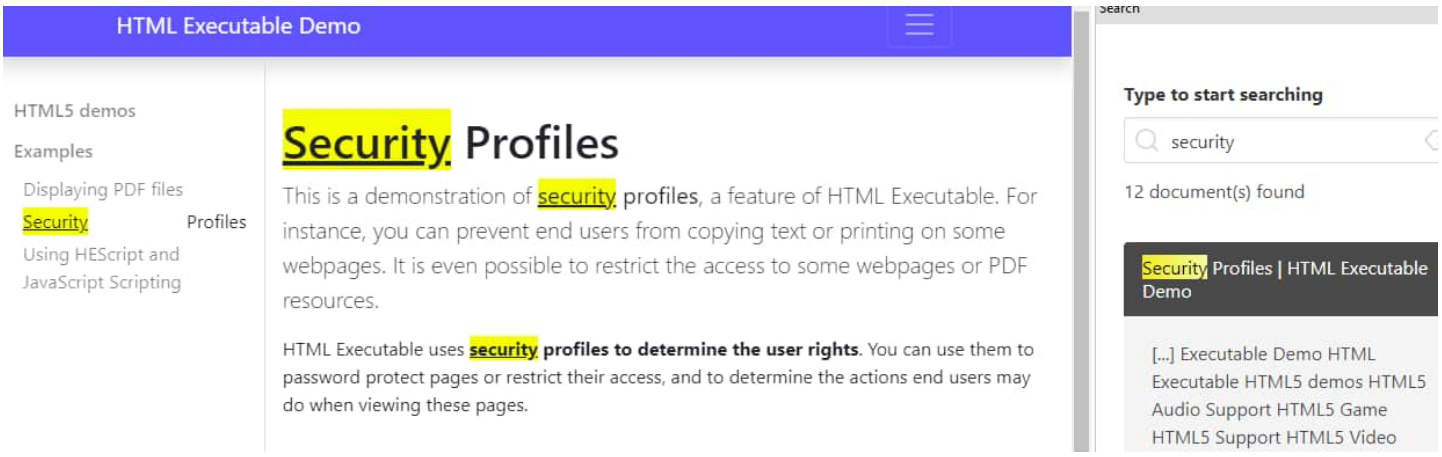
You can limit terms to specific fields. For instance, with 'title:hello', only documents with 'hello' in the title field will match. Using a field that isn't in the index will result in an error.

HTML Executable's search facility supports modifiers for terms, including edit distance and boost. Boosting a term (e.g., 'foo^5') increases the ranking of documents matching that term. Edit distance enables fuzzy matching- for example, 'hello~2' will match documents containing 'hello' within an edit distance of 2. To improve query performance, it's best to avoid large values for edit distance.

Terms can have a presence modifier. By default, the presence of a term in a document is optional, but you can make it required or prohibited. Prefix the term with '+' to require its presence (e.g., '+foo bar' searches for documents that must contain 'foo' and may contain 'bar'). Prefix with '-' to prohibit its presence (e.g., '-foo bar' searches for documents that cannot contain 'foo' but may contain 'bar').

To escape special characters, use the backslash ('\'). This allows you to include characters in searches that would typically be viewed as modifiers. For instance, 'foo\~2' will search for the term "foo~2" instead of trying to apply a boost of 2 to the search term "foo".

When a page from a search result is opened, keywords that were searched for may be **highlighted**. For PDF documents, keywords are highlighted too.


HTML Executable Demo

HTML5 demos

Examples

Displaying PDF files

Security Profiles

Using HEScript and JavaScript Scripting

Security Profiles

This is a demonstration of security profiles, a feature of HTML Executable. For instance, you can prevent end users from copying text or printing on some webpages. It is even possible to restrict the access to some webpages or PDF resources.

HTML Executable uses **security profiles to determine the user rights**. You can use them to password protect pages or restrict their access, and to determine the actions end users may do when viewing these pages.

Search

Type to start searching

security

12 document(s) found

Security Profiles | HTML Executable Demo

[...] Executable Demo HTML Executable HTML5 demos HTML5 Audio Support HTML5 Game HTML5 Support HTML5 Video

Customizing the display of search results

For advanced programmers, it's perfectly feasible to modify the HTML code and JavaScript scripts that enable the HTML Executable's search engine to function. Indeed, all these HTML and JavaScript resources can be found in the zip file named **chromium.zip**, which is available in the HTML Executable installation directory, usually located at:
C:\Program Files (x86)\HTML Executable 2023\Resources\Chromium\

The search engine's functionality can also be customized by modifying the JavaScript code found in **assets\js\search.js** within the Chromium.zip file.

However, it's important to note that only experienced users should attempt these modifications. It's essential to make backups before making any changes to prevent loss of original files or data.

Large search index

If you get an "**out of memory**" error while compiling your publication, try to enable the **Keep the search index data outside the EXE file** option available in Output Format. The error means that you have reached the free memory limit available for 32-bit programs (2 GB). In that case, HTML Executable cannot store your search index in memory, and must store it as a file on the hard disk.

4.2.6. Browse Sequence

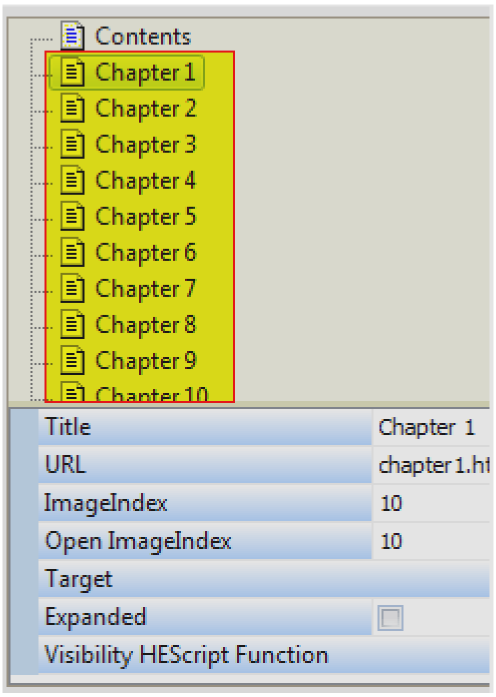
If you're creating ebooks or presentations, you might want your readers to **navigate pages in a specific order**.

HTML Executable allows you to control the behavior of the **Back and Forward buttons/menus** using two modes: **History** and **Browse Sequence**.

> **History Mode** is the standard mode where the Back and Forward buttons operate similarly to web browsers, enabling users to return to previously visited pages.
> **Browse Sequence Mode** associates the Back and Forward buttons with a specific browsing sequence. HTML pages are displayed in a predetermined order, eliminating the need for users to click on hyperlinks for navigation.

To use Browse Sequence Mode, you must have created a Table of Contents. However, displaying the Table of Contents to the users is not mandatory. You can set its 'Visible' property to 'False', preventing it from being displayed to users.

The sequence of pages is **automatically defined by their order of appearance in the Table of Contents**:



Title	Chapter 1
URL	chapter1.ht
ImageIndex	10
Open ImageIndex	10
Target	
Expanded	☐
Visibility HEScript Function	

On this screenshot, the "Contents" HTML page will be displayed first. If the user clicks Next, the publication will show "Chapter 1" then "Chapter 2", etc.

Notes

- You can of course have a visible Table of Contents and a Browse Sequence together.
- It is highly recommended that the index page of your publication or ebook belongs to the sequence (not necessarily in the first position).
- The Back and Forward buttons (and menus) are automatically disabled when the first and last pages in the sequence are displayed.

4.3. Application Behavior

The ribbon "Application Behavior" in HTML Executable lets you customize the behavior and functionality of your publication, and to localize it to other languages.

The following topics explain these features:

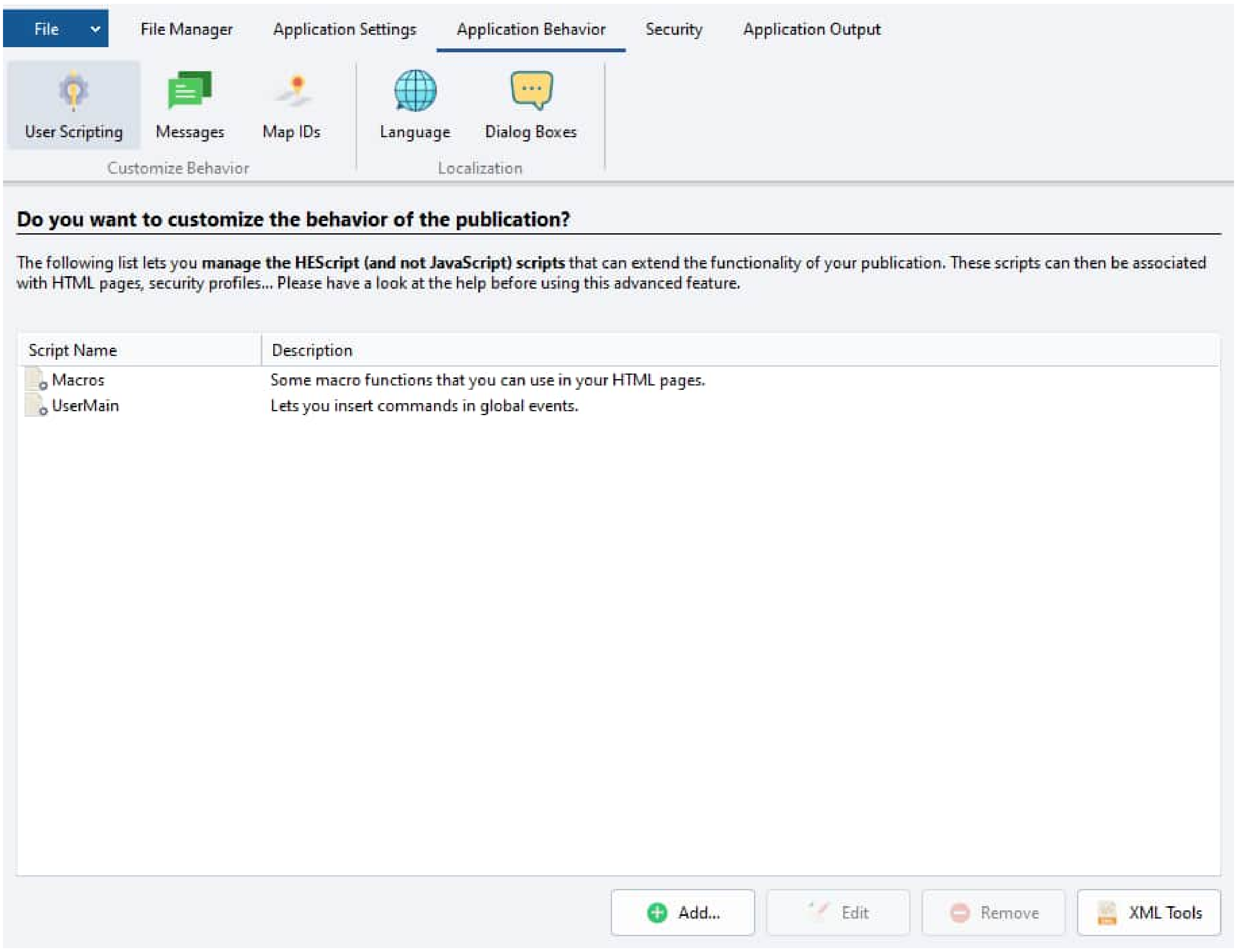
- User Scripting
- Prompt Messages
- Map IDs
- Language And Localization
- Dialog Boxes

4.3.1. User Scripting

HEScript is the scripting language of HTML Executable used to customize the behavior of your publications and ebooks. Scripts are compiled during the compilation into bytecode for faster execution at runtime (contrary to JavaScript). That's why scripts are separated from HTML pages and stored in a script manager. You can **manage scripts using this User Scripting page**.

JavaScript and HEScript can be easily combined.

Introduction to scripting with HEScript



Do you want to customize the behavior of the publication?

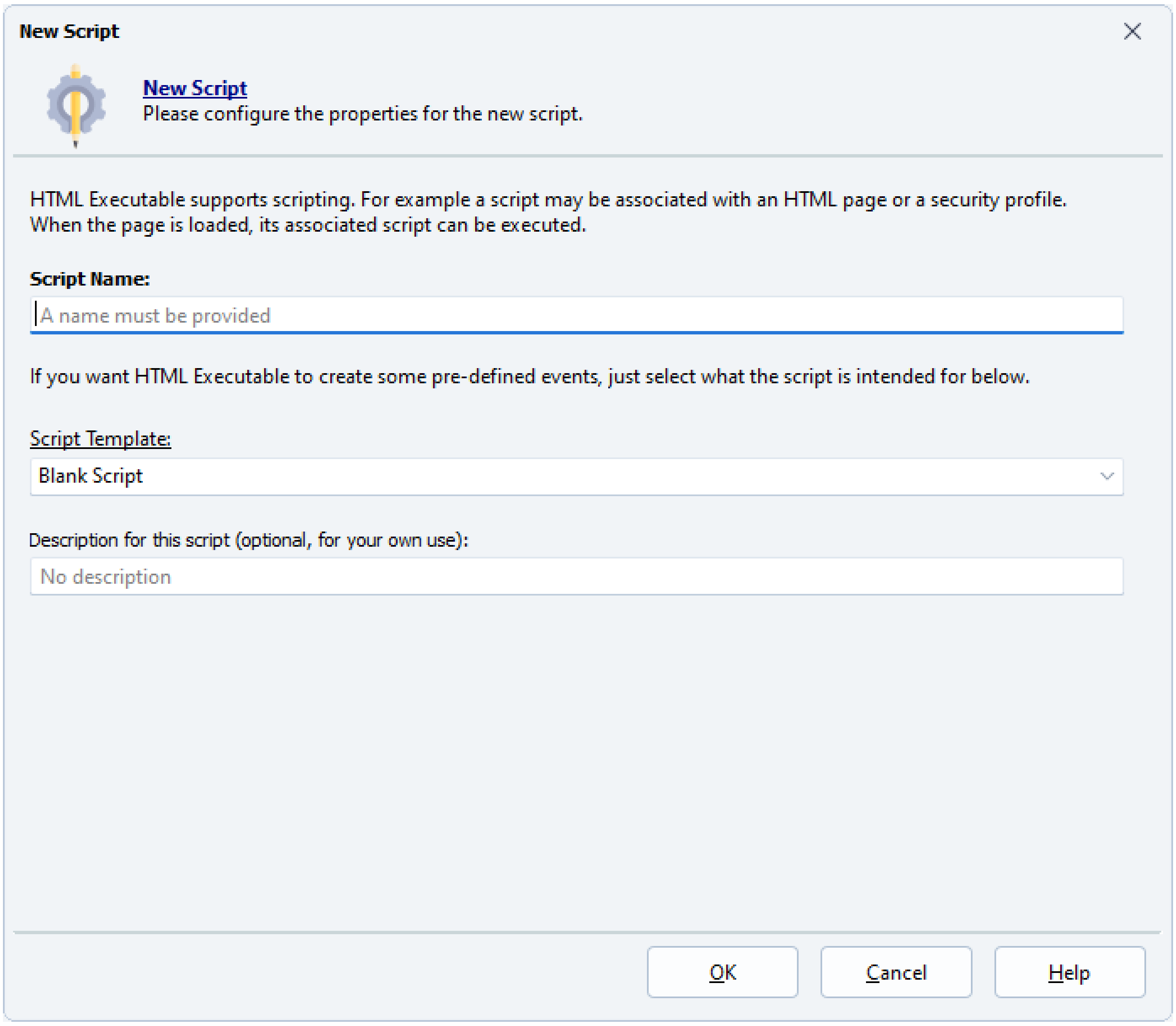
The following list lets you **manage the HEScript (and not JavaScript) scripts** that can extend the functionality of your publication. These scripts can then be associated with HTML pages, security profiles... Please have a look at the help before using this advanced feature.

Script Name	Description
Macros	Some macro functions that you can use in your HTML pages.
UserMain	Lets you insert commands in global events.

Each script that will be compiled into the script collection of the publication is **listed**. You have the **name** of the script and an optional **description**. Each script must have a **unique** name (similar to a namespace).

Adding and managing scripts

To add a new script, press **Add** and the following window will appear:


New Script

New Script
Please configure the properties for the new script.

HTML Executable supports scripting. For example a script may be associated with an HTML page or a security profile. When the page is loaded, its associated script can be executed.

Script Name:
A name must be provided

If you want HTML Executable to create some pre-defined events, just select what the script is intended for below.

Script Template:
Blank Script

Description for this script (optional, for your own use):
No description

You first give a name to the script. Space and special characters are **not** allowed (only alphanumeric ones). The maximum length for a script name is 255 characters.

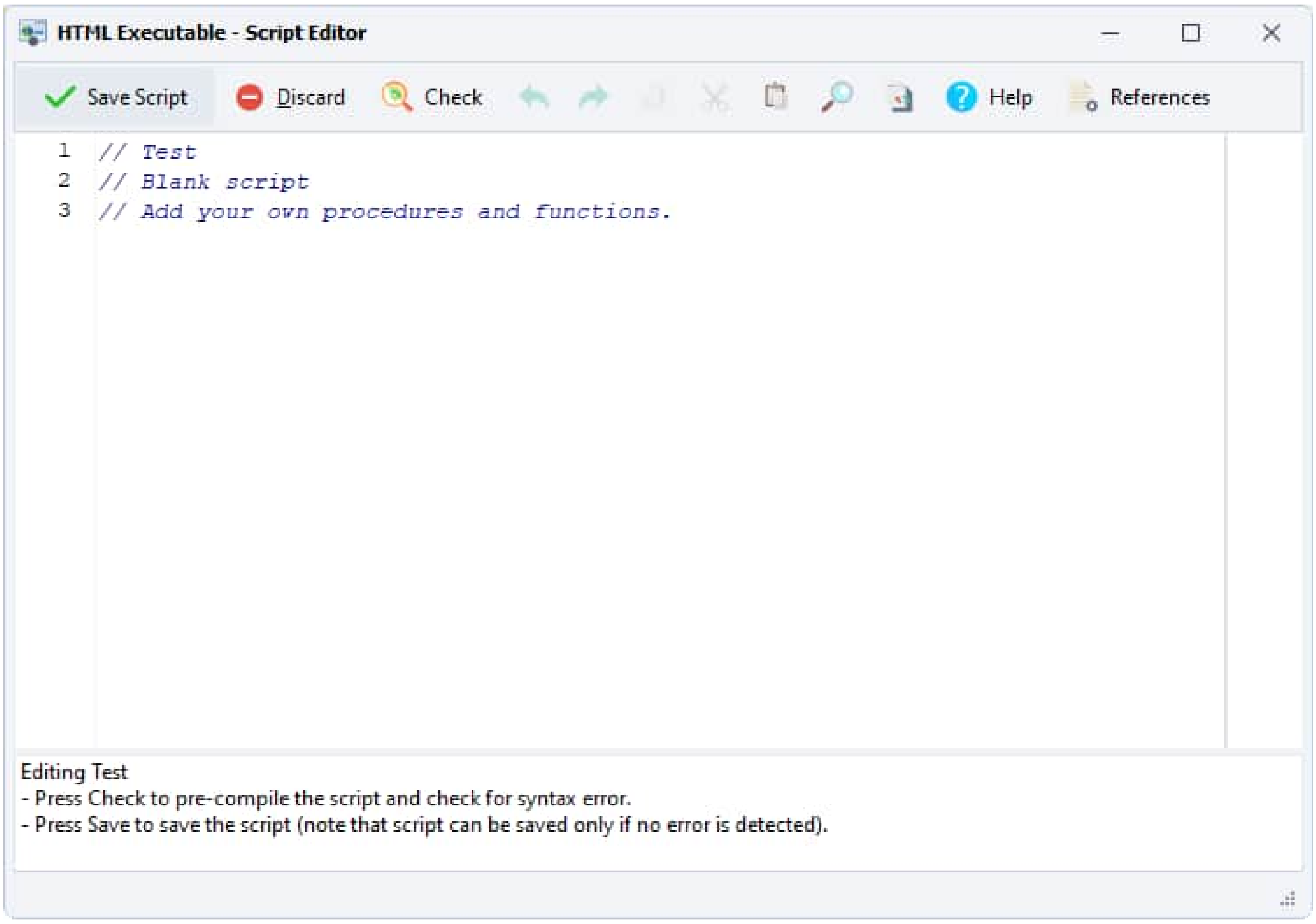
Then you can choose a template for your script. A script template is actually a blank script with some pre-defined procedures or functions called **events**. When a script is, for example, associated with an HTML page, the publication will call a pre-defined function when its related event occurs (when an HTML object is clicked for instance).

More information about pre-defined scripts, templates, UserMain and Macros

script for a security profile: this script is designed to be associated with a security profile. For instance, it may be used to create some conditions (result of a Boolean function) for the security profile.

Finally, you can give an optional description to your script. However, it is not used by HTML Executable at all.

Press **OK** to start your new script; the script editor will then be displayed to edit your new script:



```
// Test
// Blank script
// Add your own procedures and functions.
```

Editing Test
- Press Check to pre-compile the script and check for syntax error.
- Press Save to save the script (note that script can be saved only if no error is detected).

You are not forced to edit your script right now. Just click **Save** to finish the script editing: you can see that the new script then appears in the script list.

More information about the script editor

To edit a given script, select it in the list and press **Edit**. You can also double-click on it. The script editor will be displayed.

To remove a script, select it in the list and press **Remove**. Note that you can't remove the UserMain script.

To import/export a script, select it in the list and press **XML Tools**. Then select **Import/Export** and you will be prompted to enter the filename for the XML file. Scripts are stored in the XML format so you can edit them with any XML editor (or even Notepad if you wish). When you import a script back, the script editor is displayed: press Save to import the script. Note that an imported script is automatically syntax-checked (this prevents possible script errors during the compilation of the publication).

Notes

- By default HTML Executable automatically adds a "UserMain" script when a project is created. This script is designed for your own use and contains some pre-defined global events related to the ebook.
- Scripts may be associated with conditions of security profiles, etc... You can also call HEScript functions from your HTML code (links) and even from JavaScript. More information is available here.
- Scripts are stored inside the project file. No external file is required.

4.3.2. Prompt Messages

This page allows you to **display message boxes**, also known as **Prompt Messages**, to inform end users about their actions. HTML Executable enables you to display two prompt messages at different stages of the publication/ebook's execution: at the beginning and at the end.

Start Prompt Message

This message is displayed immediately after the publication or ebook is launched. It prompts end users to confirm whether they wish to proceed with your publication (Yes and No buttons are provided). An example of such a message could be: "Welcome to my publication. This publication is designed for optimal viewing on a large screen resolution. Do you wish to continue?".

End Prompt Message

This message is displayed as soon as users close the publication. It is an opportunity to thank your users or provide additional information, for example: "Thank you for reading my publication! Find more information at: http://www.mywebsite.com".

Notes

- If the fields for these messages are left blank, these message boxes will not be displayed.
- To insert a carriage-return-line-feed between "Line 1" and "Line 2", use `"#13#10"`, like `Line1"#13#10"Line2` (including the quotes).
- These prompt messages are not rendered with the skin engine, because at the time they are displayed, the skin engine is either not yet loaded or has already been unloaded.

4.3.3. Map IDs

As a programmer, you can leverage HTML Executable to **compile help files or documentation in HTML format** for your applications. The HTML Executable APIs allow you to open help publications, specify the help topic to be displayed, and close them when your application is closed, effectively integrating the publications with your own applications.

Rather than specifying the HTML page's filename for the help topic you wish to display, you have the option to pass a numeric ID to the API or the command line. HTML Executable conveniently associates a unique ID (page ID) with each HTML page it compiles into a publication.

These IDs collectively form what is known as a map. HTML Executable can be configured to export this map to a file (termed as MAP file) that can be seamlessly integrated into your application's project.

At present, HTML Executable can generate MAP files in Object Pascal (compatible with Embarcadero Delphi and FreePascal), C/C++ (compatible with Borland C++ Builder or Microsoft Visual C++) and in simple text files. Just select the programming language that suits your need.

By default, the map file is created in the same folder as the output file (provided the option is enabled).

If desired, you can add your own custom page IDs (similar to HTML anchors, HEScript commands) to work with map IDs or the HTML Executable API SDK. These custom page IDs will be automatically exported to the map file. To add a custom page ID, click 'Add' and input its target. To remove a custom page ID, select it and click 'Remove'.

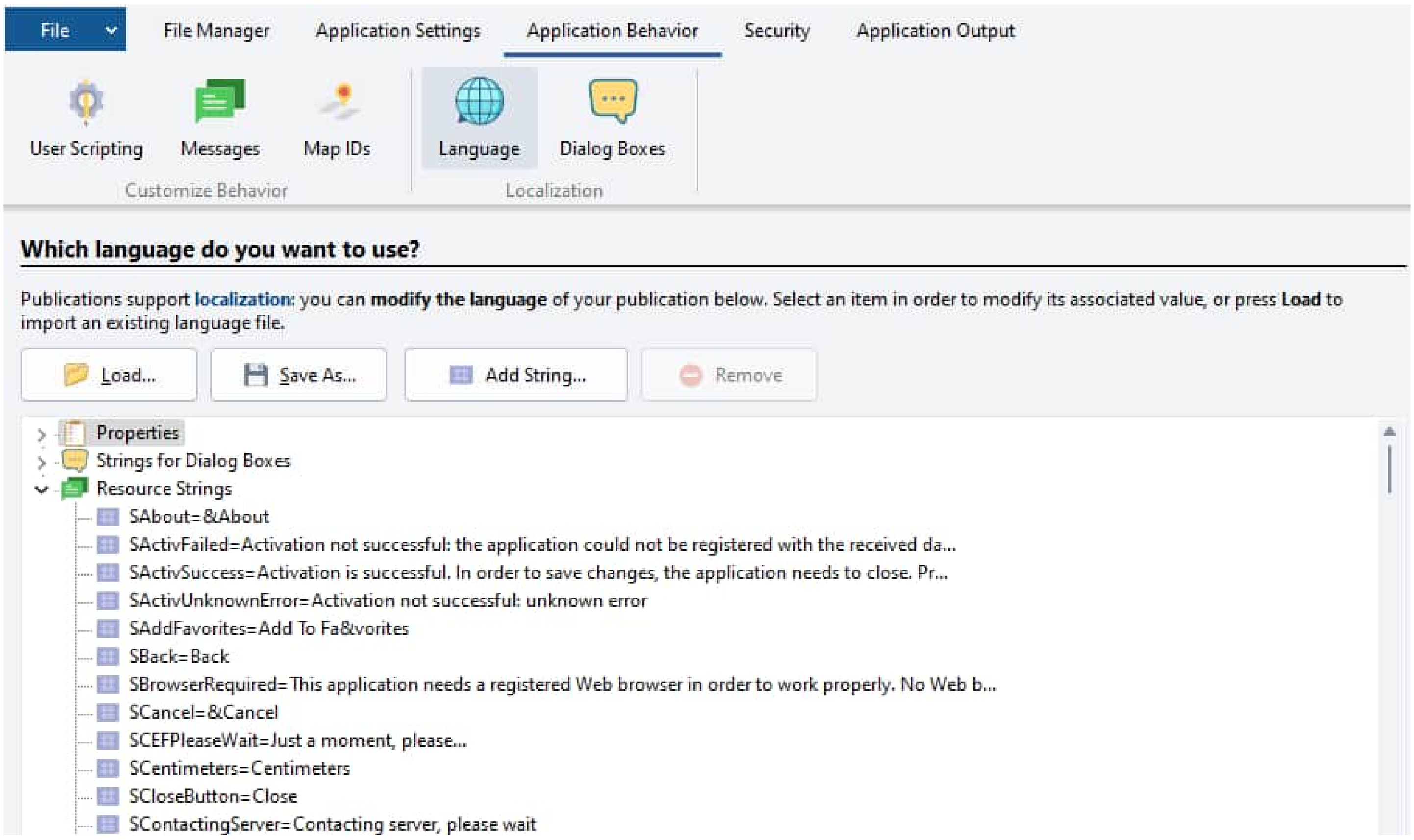
Map IDs can also be passed via the command line, giving you control over your publication from other applications.

The HTML Executable API SDK is available to registered users upon request.

4.3.4. Language And Localization

HTML Executable offers **localization support** for your publications and ebooks. It can display explanatory field text and dialog captions in your chosen language when your publication or ebook is run. This page guides you through managing the localization of your publication and **loading/saving language files**.

To alter the locale of certain internal menus or texts utilized by the HTML rendering engine, you can specify the desired locale thanks to the DefLocale property (e.g., 'en-us' for English, 'fr' for French, 'de' for German, and so forth).



Which language do you want to use?

Publications support **localization**: you can **modify the language** of your publication below. Select an item in order to modify its associated value, or press **Load** to import an existing language file.

- Properties
- Strings for Dialog Boxes
- Resource Strings
  - SAbout=&About
  - SActivFailed=Activation not successful: the application could not be registered with the received da...
  - SActivSuccess=Activation is successful. In order to save changes, the application needs to close. Pr...
  - SActivUnknownError=Activation not successful: unknown error
  - SAddFavorites=Add To Fa&vorites
  - SBack=Back
  - SBrowserRequired=This application needs a registered Web browser in order to work properly. No Web b...
  - SCancel=&Cancel
  - SCEFPleaseWait=Just a moment, please...
  - SCentimeters=Centimeters
  - SCloseButton=Close
  - SContactingServer=Contacting server, please wait

Resource Strings

Resource strings are specific string constants used to translate different messages and captions of user interface items such as menus, buttons, and window titles.

For instance, if you aim to build a publication in both French and English, you do not need to translate all texts in the different fields, find all options, actions, etc. Instead, you just need to modify a list of strings, which is the primary function of the resource string editor.

Resource strings can be invoked with the **percent symbol** `%` placed before and after a constant, like so: `%SAbout%`. These "string identifiers" replace string variables or literals. At runtime, publications substitute each string identifier with its corresponding string value.

You can also embed resource strings directly into your HTML pages (explained below).

To **edit a resource string**, simply select it from the list and an editor will appear:

Modify the contents of the Value field, then press Apply to save the new value of the resource string.

To **add a resource string**, click "Add String". You will be prompted to enter a name for the new resource string. This name should begin with an `S`, contain no spaces or special characters (only alphanumeric ones), and must be unique. You can then modify the value of the new resource string and press Apply to finalize.

To **remove a resource string**, select it and click Remove. Only resource strings that you added (user resource strings) can be removed.

How to Embed a Resource String Directly Into Your HTML Page

Use JavaScript code to insert resource strings:

```
<script>
document.write(window.external.GetString('[name of the resource string]'));
</script>
```

If we follow the same example as above, this will lead to:

```
<script>
document.write(window.external.GetString('SPubCopyright'));
</script>
```

Strings for Dialog Boxes

These specific strings can only be used in Dialog Boxes to translate their titles and messages.

They can be invoked with the following syntax `[#ID]` where ID is the "string identifier". Such IDs always begin with `Y`; example: `[#YAbout]` will be replaced by the value of "YAbout" which is "About".

They are **not available at runtime** contrary to resource strings: all references to these strings are directly replaced when HTML Executable compiles the publication.

Managing these strings is exactly the same as for resource strings: see above.

About language files

You can **import/export** all strings from/to **language files**. These files are given the .hel extension. Use the **Load/Save** buttons to import or export language files.

Thus language files may be used in different publications. You can even share them with others.

Please update the **properties** of a language file if you plan to distribute it. Properties are only for informative purposes at this time, but this may change in the future. The **locale** number is the numeric "language identifier" of the language. Visit [http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx](https://msdn.microsoft.com/en-us/goglobal/bb964664.aspx) for the list of available language identifiers.

Current strings are automatically saved in the project file, so you do not need to maintain a language file for your project.

You can specify a default language file in the Environment Options so that it is always loaded when a new project is created.

Additional resource strings

Some resource strings are automatically created by HTML Executable when compiling your publication:

- SPubTitle contains the title of your publication.
- SPubHomepage contains the default homepage URL.
- SPubAuthor contains the author name.
- SPubCopyright contains the copyright of your publication.
- SPubEMail contains the author E-Mail.
- SPubVersion contains the version number of your publication, as defined in the Icon / Version tab.
- SPubProductVer contains the product version number of your publication, as defined in the Icon / Version tab.

You can use these resource strings in your HTML pages and scripts.

4.3.5. Dialog Boxes

Introducing dialog boxes

All dialog boxes displayed by publications and ebooks are actually HTML pages acting as custom HTML dialogs. Among dialog boxes you can find the Search engine panel, the About dialog box, error pages, favorite manager, etc...

Because you may want to customize the contents of these dialog boxes, you can edit their HTML code.

However, please note that you **should never modify the scripts or the forms inside these dialog boxes**. In fact, these pages are also managed by internal scripts that control how the publication works.

HTML code for dialog boxes is encoded in UTF-8 format, for Unicode support.

Some dialog boxes contain items added at runtime by the publication (like Favorite Manager, Add Favorites, Search Results...). You can customize these items with the dedicated resource strings: for instance, see the resource strings that lets you customize search results.

Dialog boxes use resource strings called "**Strings for dialog boxes**" for easier localization. All references to these strings are directly replaced when HTML Executable compiles the ebook. To **insert a reference** to a string for a dialog box, use this syntax: `[#ID]` where ID is the id of the string (IDs always begin with Y); example: `[#YAbout]` will be replaced by the value of "YAbout" which is "About".

Modifying dialog boxes

To **edit a dialog box**, select it in the list and click **Edit** (or double-click). The internal HTML editor will appear. Press OK to save your changes. It is your responsibility to check whether the pages work fine once modified.

To **import/export a dialog box**, select it in the list and press **XML Tools**. Then select **Import/Export** and you will be prompted to enter the filename for the XML file. Pages are stored as CDATA items in the XML format so you can edit them with any XML editor (or even Notepad if you wish). When you import a dialog box, the existing one is overwritten without any prompt.

If you have made several modifications to dialog boxes and you would like to use them in other HTML Executable projects, you can export all dialog boxes to a **Template file**. Template files are files with .HESP extension. You can import and export dialog boxes with the **Templates** button.

Templates are designed for exporting and importing all dialog boxes. If you want to export one dialog box only, use the **XML Tools**.

When you start a new project, HTML Executable loads initial dialog boxes from its internal resources in the zip file named **chromium.zip**, which is available in the HTML Executable installation directory, usually located at: C:\Program Files (x86)\HTML Executable 2023\Resources\Chromium\

By default, dialog boxes can be **resized by end users**. If you want to avoid this, please add the following meta tag to the HEAD section of the system HTML page's code:

```
<meta name="WindowNoResize" content="1">
```

4.4. Security

The ribbon "Security" in HTML Executable lets you control the security options of your ebook or HTML publication, and to create restricted versions with tools dedicated to selling ebooks.

The following topics explain these features:

- Global Protection
- Security Profiles
  - Security Profiles - Conditions
  - Security Profiles - Actions and Restrictions

- Certificates - Trial and Registered
- Content Protection
- Code Signing (Digital Signatures)
- Dongle Protection

4.4.1. Global Protection

Securing your digital content is a primary concern in today's digital age, especially when it comes to your websites. Compiling your websites is an effective approach to **protect your HTML pages from unauthorized access, copying, or theft**. End-users must execute the compiled website's .exe file to view your content, ensuring the safety of your original files. Moreover, it is impossible to unpack a compiled publication or ebook using a file archiver, such as 7-Zip, without actually running it. Furthermore, non Self-Extracting publications are resistant to decompilation. Please note that it's important to maintain a backup of your source files because once a publication is compiled, the extraction of source files is not feasible.

HTML Executable offers **an array of security options** for your publications and ebooks, which you can tailor according to your requirements:

Global Password

If you wish to restrict the access to your publication, you can **password protect** it: end users will be prompted for the password before the ebook starts. If the password entered is incorrect, then an error message is displayed (given by SInvalidPassword resource string) and the ebook closes immediately.

You can also customize the behavior of the publication when the password is incorrect thanks to the HEScript `UserMain.OnInvalidPasswordAtStartup` Boolean event. The runtime module invokes this event if the password provided by the user is incorrect. If you set the result of this event to True, the publication does not exit.

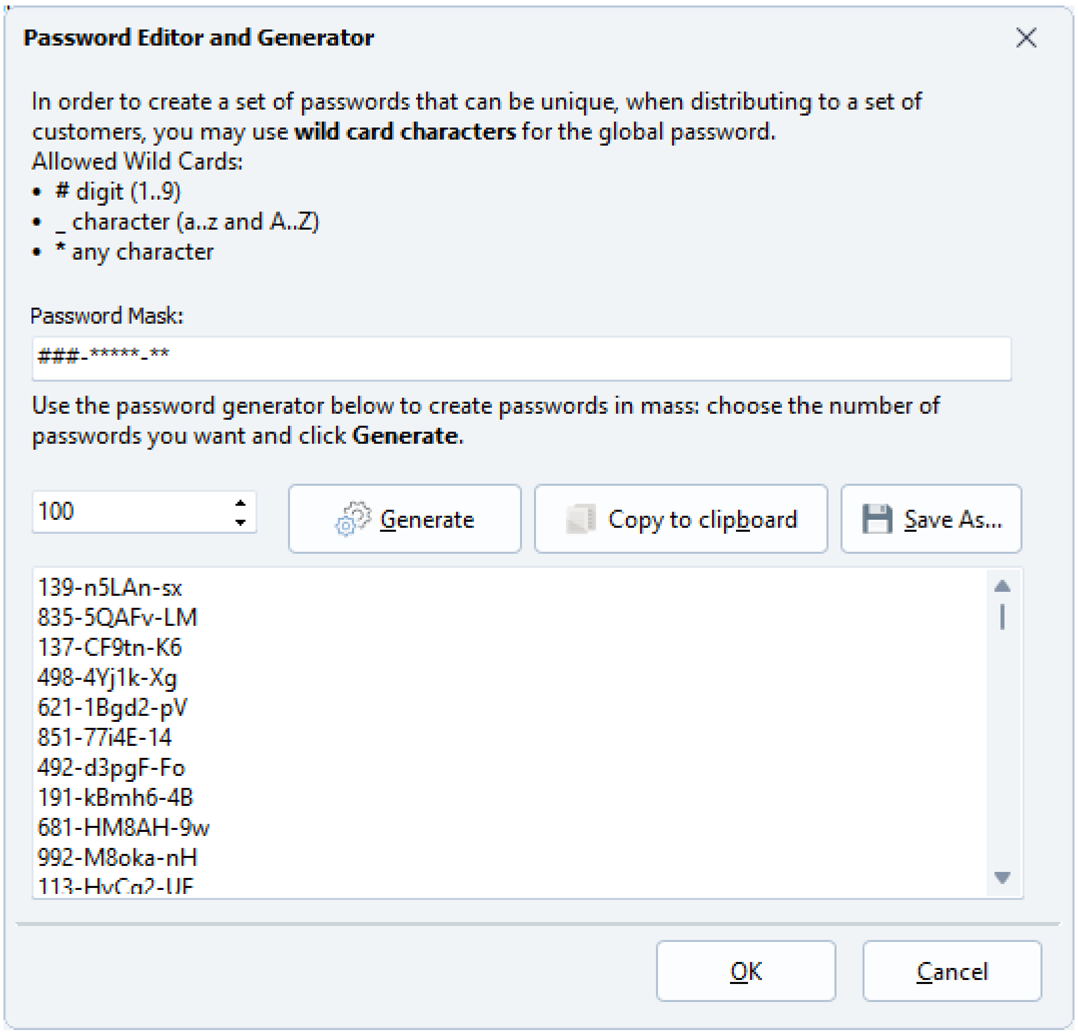
```
function OnInvalidPasswordAtStartup: Boolean;
begin
 Result := True;
end;
```

In order to create a set of acceptable passwords that can be unique, when distributing to a set of customers, **wild card characters are accepted**. Thus, you can set up the following global password: `123-4-**A**` (and so any character can be input for the wildcard place holder).

Allowed wild card characters:

`#` digit (1..9)
`_` single character (a..z) and (A..Z)
`*` any character.

HTML Executable comes with a **password generator** that lets you create lists of random passwords based on the provided mask.


Password Editor and Generator

In order to create a set of passwords that can be unique, when distributing to a set of customers, you may use **wild card characters** for the global password.
Allowed Wild Cards:
- # digit (1..9)
- _ character (a..z and A..Z)
- * any character

Password Mask:

```
###-*****-**
```

Use the password generator below to create passwords in mass: choose the number of passwords you want and click **Generate**.

100

```
139-n5LAn-sx
835-5QAFv-LM
137-CF9tn-K6
498-4Yj1k-Xg
621-1Bgd2-pV
851-77i4E-14
492-d3pgF-Fo
191-kBmh6-4B
681-HM8AH-9w
992-M8oka-nH
113-HvCq2-UF
```

Choose the number of passwords to create and click **Generate**.

Ask for the password only once and store it

Once you have set up a global password, your end users will be asked for it each time they start your ebook. If you wish to avoid that, enable this option. Once the password has been validated, it will be stored and your users can access your ebook **without being prompted for the password each time**.

The password is either stored on the local computer OR on the storage device (USB stick for instance) if you are making a portable publication.

Set a global expiration date

If you would like your publication to expire after a specific date (it does not run anymore), then just select the expiration date you want. After that date, an error message is fired (given by `SPublicationExpired` resource string) and the publication closes immediately. Changing the system clock will not modify the expiration state.

{{% notice info %}} This option, however, is not very safe: if you want to create Trial publications that can really expire "forever", then make a restricted publication. {{% /notice %}}

When testing your publication on your own computer, you can remove the expiration state by clicking "**Clear expiration info**". This function will only work if the publication has expired. It is possible to remove the expiration state on other computers using the Cleanup utility.

By **associating the expiration date to a version number**, you have a way to reset the global expiration once you release a new version of your ebook. For instance, you have a global rule to expire the software on `5/1/2016`, and once it's expired, you will be releasing a new version requiring everyone to upgrade.

Finally, you can also customize the behavior of the publication when the expiration date is reached thanks to the HEScript `UserMain.OnExpiredPublication` Boolean event. The runtime module invokes this event and if you set its result to True, the publication does not exit.

```
function OnExpiredPublication: Boolean;
begin
 Result := True;
end;
```

Check publication size at startup

When a publication is downloaded from the Internet, if the download was not successful, this may result in a truncated file. In this case, running non-complete publications may not be safe. To prevent truncated downloads or size-modified publication files, you may enable this option. When turned on, it forces the publication to check its size. If the size is not the same as it was when the publication was built, an error message is fired.

This option is superseded by digital signatures. If you have the necessary Authenticode files (a code signing certificate), then it is even better to sign your publication.

Disable the PRINT SCREEN key

The PRINT SCREEN key allows Windows users to capture the whole screen to a bitmap, called a screenshot. This screenshot is then saved to the Clipboard and users can paste the result in any word processing tool or image editor. If you would like to disable this function, just turn on "**Disable Print Screen**": pressing the key won't take screenshots anymore when the publication is running.

This function, however, does **NOT** stop screen capture tools. Please take a look at Content Protection instead.

Notes:

Some programs (such as these capture tools) may also try to override the Print Screen hotkey. In this case, there may be conflicts.
Self-Extracting publications are not able to disable all Print Screen keystroke combinations contrary to other publication types.

Only one instance of the publication can be run at a time

Enable this option to make sure that only **one** instance of your publication executes. If the user tries to run a second instance, it immediately exits and the previous instance gets enabled and visible.

Moreover, command-line arguments are directly passed to the running instance: this feature is useful for help files as it lets you change the current topic for instance without having to close and start another instance of the publication.

Do not allow virtual printers

This is only available for PDF files displayed with the built-in PDF viewer.

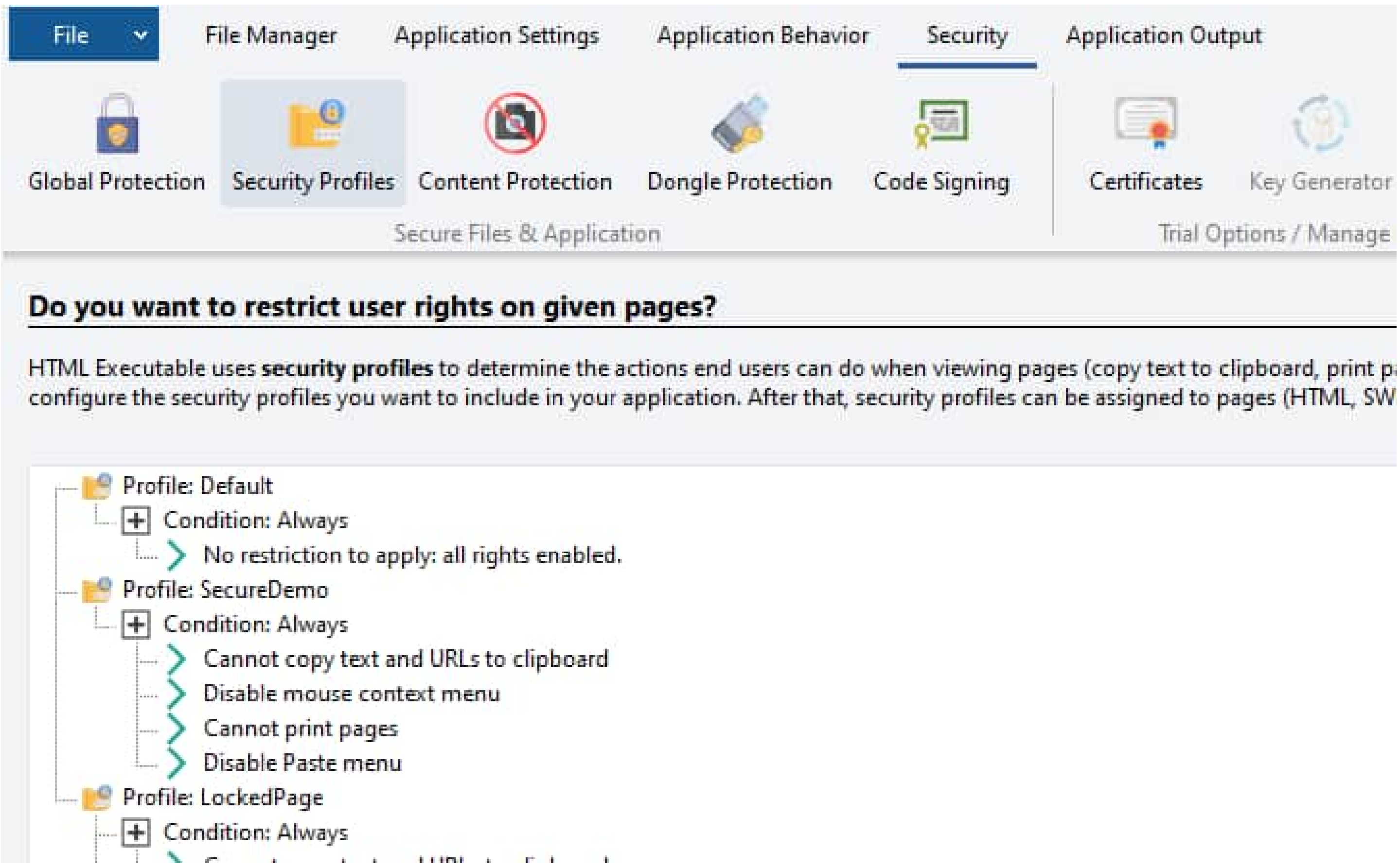
This option prevents end users from printing the PDF with a virtual printer: only physical printers are allowed. Otherwise, an end user could recreate the secured PDF file with a virtual PDF printer (which saves its output to a PDF file).

If a virtual printer is detected, the following error message is shown: `Printing is not allowed with the selected printer. Please choose another one and retry.` which can be customized thanks to the resource string named `SPrinterNotAllowed`.

HTML Executable succeeds in detecting most common virtual PDF printers. If you find a virtual printer not detected, you can report it to us.

4.4.2. Security Profiles

Security profiles are a powerful security feature in HTML Executable. They allow you to group pages of your ebook or publication into "profiles". These determine the **accessibility of a page based on certain conditions such as a password, trial publication, certificates, script function result, and so on**. They also dictate the **actions that end users are allowed to perform** when viewing the page, including copying text, printing pages, copying images, and more.



Do you want to restrict user rights on given pages?

HTML Executable uses **security profiles** to determine the actions end users can do when viewing pages (copy text to clipboard, print p
configure the security profiles you want to include in your application. After that, security profiles can be assigned to pages (HTML, SW

- Profile: Default
  - Condition: Always
    - No restriction to apply: all rights enabled.
- Profile: SecureDemo
  - Condition: Always
    - Cannot copy text and URLs to clipboard
    - Disable mouse context menu
    - Cannot print pages
    - Disable Paste menu
- Profile: LockedPage
  - Condition: Always

Description of Security Profiles

- **A security profile** contains one or more "conditions": when a condition is fulfilled (if .... is True), the associated actions are executed (then do .....).

- **Security profiles** are linked with HTML pages, PDF and DOCX documents. When a page is requested, the publication first loads its security profile, then analyses the different conditions. If any of these conditions are met, their corresponding actions are executed.

IMPORTANT: Security profiles only work for PDF documents if the built-in PDF viewer is activated.

Goal of Security Profiles

Security profiles can be employed for a variety of purposes. Here are some examples:

- You can create a security profile that password-protects pages. If an end user fails to provide the correct password when prompted, the page is not displayed. See the steps for this example.
- You could disable the print function for a group of pages.
- You can effortlessly disable the "Copy to Clipboard" functions.
- If you create trial or restricted ebooks or publications, you can **lock certain chapter pages in the demo version of your ebook**. You can then send a key to registered end users to unlock the remaining chapters.

For a detailed example of working with security profiles, including an animated tutorial, sample projects, and a complete ebook, visit https://www.htmlexe.com/samples. Be sure to grab a copy!

Managing Security Profiles

To manage security profiles, navigate to the "Application Behavior - Security Profiles" page.

Before you assign security profiles to HTML pages, you must first create them. These profiles are managed using the tree editor, as depicted below:

- To add a security profile, simply click 'Add' and select 'Add New Security Profile'. Each new security profile is created with a condition named "Always".
- To add a new condition, select the profile you wish to modify, press 'Add', and select 'Add New Condition'. You will be prompted to determine the condition (see conditions of security profiles).
- Finally, you can configure the actions, restrictions or page locking of a given condition by clicking 'Configure'.
- To remove an action, a condition, or a security profile, select it and press 'Remove'. Note that the 'Always' conditions and the 'Default' profile cannot be removed.

Associating Security Profiles to HTML Pages and PDF Documents

Once your security profiles are ready, go to the File Manager, select the HTML page(s) you want to modify and click 'Properties'. The File Properties editor will appear, select the Security tab and choose the security profile you want to assign to the selected HTML page(s) or PDF document(s).

Conditions of Security Profiles

Please see the dedicated topic about conditions of security profiles.

Actions, Restrictions, and Page Locking

Please see the dedicated topic about actions and restrictions of security profiles.

4.4.2.1. Security Profiles - Conditions

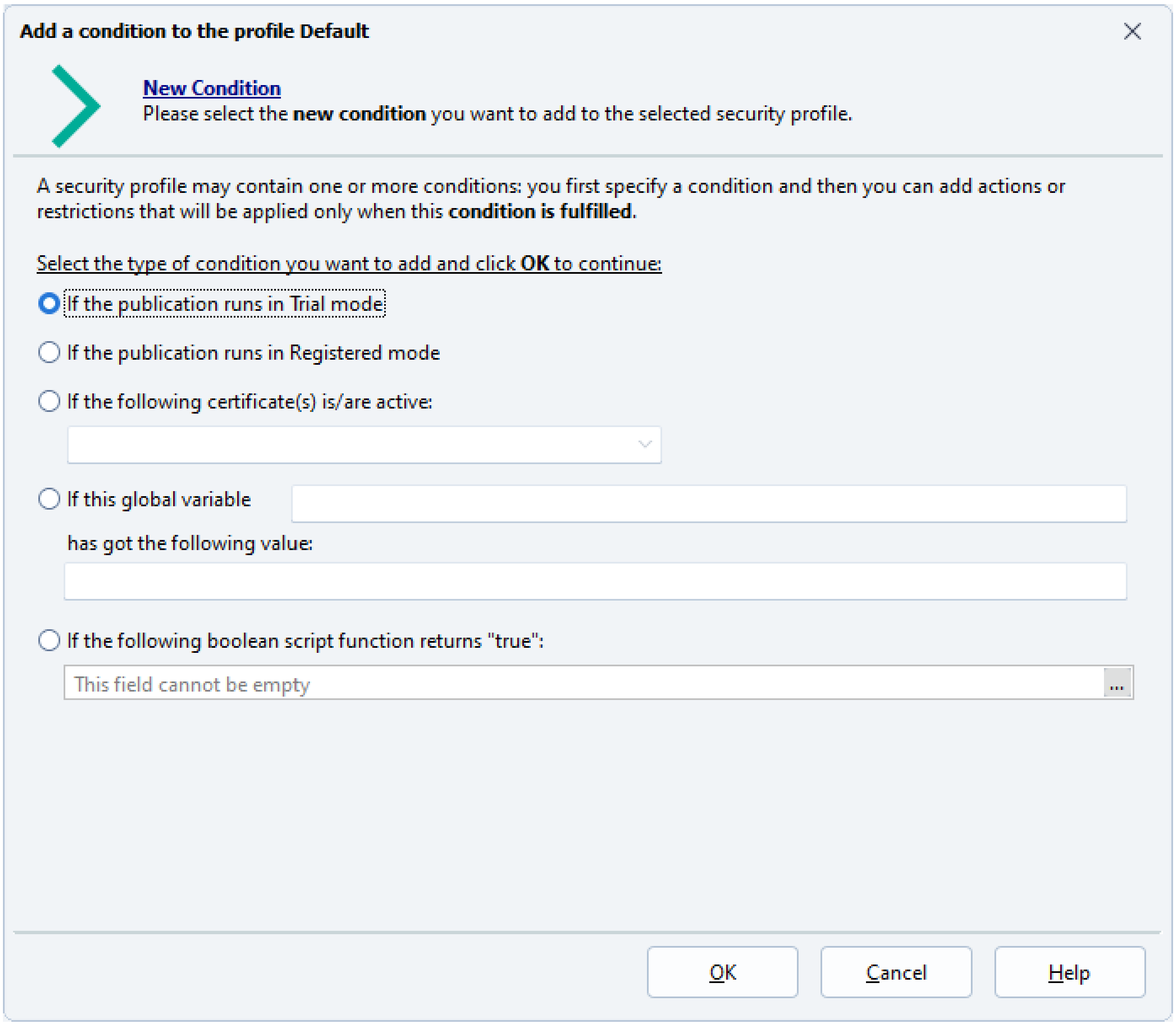
Working with Security Profiles and Conditions

Always-On Actions

Each security profile comes with a default **Always** condition. This ensures that any actions linked to this condition are executed without fail, providing a consistent user experience.

Adding New Conditions

When you're ready to add a new condition, you'll be greeted with a selection window:



Add a condition to the profile Default

New Condition
Please select the **new condition** you want to add to the selected security profile.

A security profile may contain one or more conditions: you first specify a condition and then you can add actions or restrictions that will be applied only when this **condition is fulfilled**.

Select the type of condition you want to add and click **OK** to continue:

[✓ If the publication runs in Trial mode]

If the publication runs in Registered mode

If the following certificate(s) is/are active:

If this global variable

has got the following value:

If the following boolean script function returns "true":

This field cannot be empty

Select the type of condition you want to create, fill in any required fields, and press **OK** to continue.

⚠ **WARNING:** The options for "Trial and Registered modes" and "following certificate(s) active" are only supported if you're creating a restricted publication.

Using Certificates for Conditions

The **following certificate(s) is/are active** condition provides a flexible tool for creating different registered editions of your publication. For instance, you can lock certain HTML pages, PDF or DOCX documents if the current certificate is the default one, indicating that the ebook is not registered. Moreover, by creating individual certificates for each edition, you can unlock a specific number of pages per edition.

To activate this option, use the list to specify which certificate(s) should be checked. The checkboxes allow you to easily select or deselect certificates.

Learn More About Certificates

Leveraging HEScript Boolean Functions

Creating a new condition is most efficient when using an HEScript Boolean function. These functions return either True or False, making them perfect for driving conditions. If the HEScript Boolean function returns true, the associated actions will trigger.

How to Create an HEScript Script

Using Global Variables

For those who prefer working with publication global variables, you have a third option. In this case, enter the global variable's name and the required value for the condition to be fulfilled. Keep in mind, this option is not case-sensitive.

Working with global variables

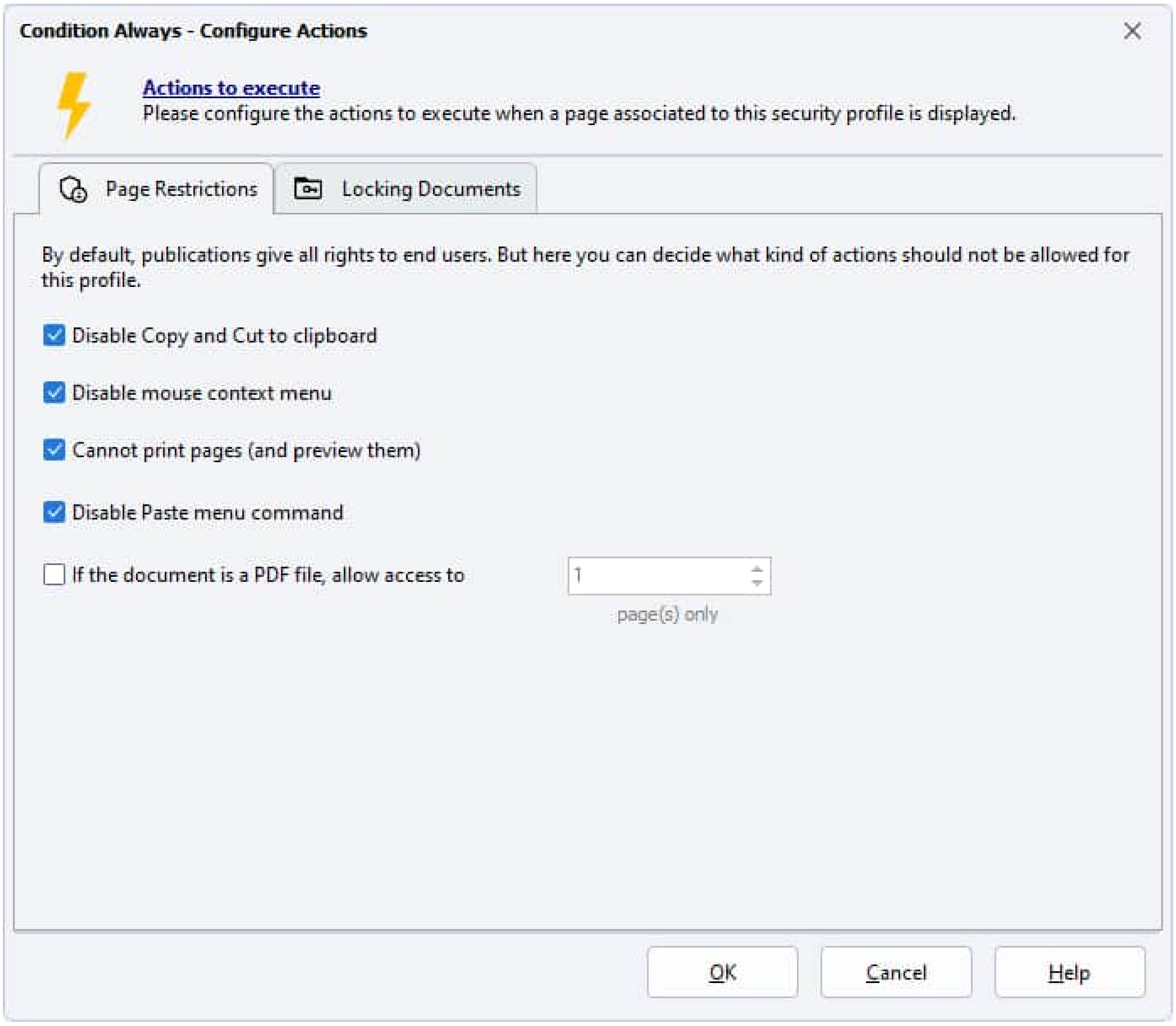
IMPORTANT: Remember that once a condition is created, it cannot be edited. If you need to modify an existing condition, you'll need to delete it and create a new one.

Return to Security Profiles

4.4.2.2. Security Profiles - Actions and Restrictions

In the realm of Security Profiles, each security profile condition can be **associated with multiple restrictions or actions**. These actions or restrictions are then executed or applied once the associated condition is met.

To configure these actions, a condition must first be selected. Following this selection, you can either click `Configure` or simply double-click on the condition, which will cause the configuration window to appear:


Condition Always - Configure Actions

Actions to execute

Please configure the actions to execute when a page associated to this security profile is displayed.

By default, publications give all rights to end users. But here you can decide what kind of actions should not be allowed for this profile.

- [✓] Disable Copy and Cut to clipboard
- [✓] Disable mouse context menu
- [✓] Cannot print pages (and preview them)
- [✓] Disable Paste menu command
- [ ] If the document is a PDF file, allow access to 1 page(s) only

In this window, you will find two tabs with different sets of options:

1. Page Restrictions

Page Restrictions allow you to control what actions end users can perform when viewing the current HTML page, PDF document or DOCX document (when using the built-in viewer). By default, ebook applications allow end users to perform a variety of actions such as selecting and copying text, printing pages, displaying the context menu, among other actions.

However, if you wish to impose restrictions, simply enable the desired option(s):

- **Prevent end users from selecting text**: This restricts users from selecting any portion of your HTML text or PDF document. The text will be displayed in a manner akin to print preview mode, but this does not prevent users from navigating through your publication. This feature is commonly employed when you wish to disable the "copy to clipboard" functionality.
- **Cannot copy text and URLs to clipboard**: If enabled, this feature prevents end users from selecting parts of your HTML documents or URLs and copying them to the clipboard.
- **Disable mouse context menu**: This feature disables the mouse context menu (accessed via right click), which allows users to access various commands like copy, print, and top page. You can also partially disable some commands using the context menu component.

- **Cannot print pages**: This restriction disables the ability for users to print pages from the publication. By default, end users can print HTML, PDF and DOCX documents from your publication, but enabling this feature removes this capability, along with the Print button from the toolbar.
- **Cannot copy images to clipboard**: Similar to the "Cannot Copy Text" feature, but in this case, it pertains to images and is only applicable to HTML Viewer publications.
- **Disable Paste command**: This removes the Paste command from the Edition and context menus.
- **If the document is a PDF file, allow access to X pages**: This option lets you limit the number of pages a user can access in PDF documents (only if the built-in PDF viewer is enabled). For instance, if you enter 2, only the first two pages will be accessible, while the remaining ones will be removed by the viewer.

2. Locking Pages

This feature allows you to prevent end users from accessing the current HTML page, PDF or DOCX document. When a page is locked, it cannot be displayed, and instead, an error page is shown. You have the option to specify which page should be displayed in such instances.

When an HTML page is being requested, the publication loads its associated security profile and executes all actions of fulfilled conditions. If one of the actions is of the "page is locked" type, then the HTML page is not displayed at all.

Notes:
1. If a "page is locked" action is placed in the "Always" condition of a security profile, any page associated with that profile will always be locked.
2. The "page is locked" actions can be used to restrict user access to your pages.

By playing with the different conditions of a security profile, you can determine whether a user should have access to a page or not.

See this sample about how to use a security profile to password protect pages (using "Locking Pages").

Return to Security Profiles.

4.4.3. Content Protection

Protecting your ebook or publication content is crucial to safeguarding your intellectual property and preventing unauthorized sharing and piracy. HTML Executable offers robust content security features that enable authors to protect their work effectively.

The Importance of Content Protection

As an author, your content represents your creativity, effort, and expertise. By implementing content protection measures, you can:

Prevent Unauthorized Access: License validation ensures that only users with valid registration keys can access your ebook, allowing you to control its distribution and protect against fraudulent use.

Counteract Piracy: Content security measures, such as preventing screen captures and blacklisting unauthorized software programs, help deter piracy and unauthorized distribution of your work.

💼 Safeguard Intellectual Property: Content protection safeguards your intellectual property rights, preserving the value of your work and ensuring you can monetize your efforts.

Enable Publication Content Protection

To prevent unauthorized screen captures, HTML Executable provides the option to utilize the Windows built-in content protection feature. This feature effectively blocks screenshot utilities, enhancing the security of your ebook:

⚠ **Warning:** This feature is only available on Windows 7 and later versions. Desktop Window Manager and desktop composition must be enabled. Refer to this link for instructions on enabling it.

Once enabled, if someone attempts to take a screenshot or capture the screen using video recording software, the window will appear black, ensuring the protection of your content.

Handling Content Protection Activation Failure

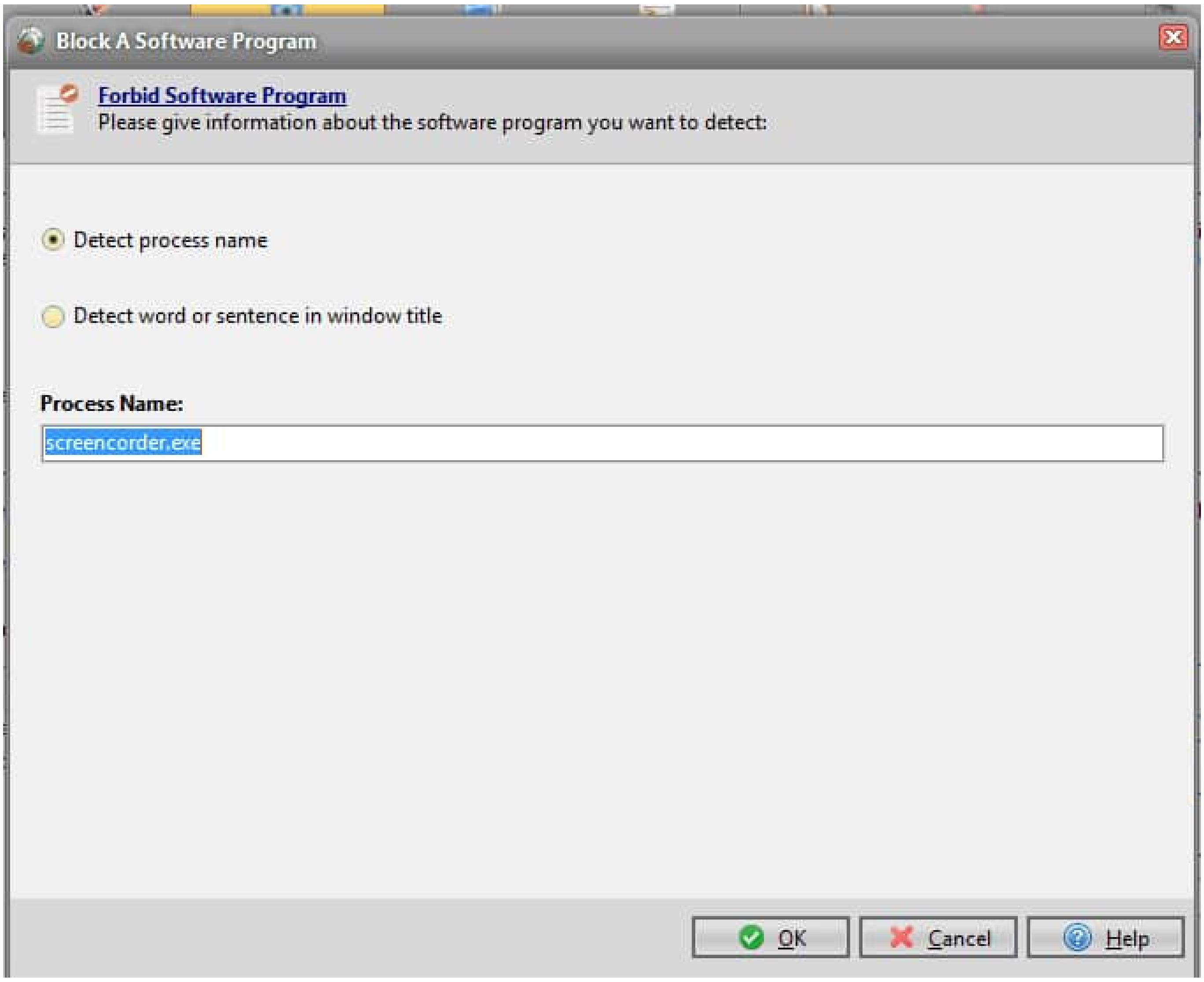
In cases where content protection cannot be activated, HTML Executable provides an error message and exits the publication. This ensures that your ebook is not run without the necessary content protection. You can customize the error message using the `SErrorCopyProtectFailed` resource string.

Forbid Software Programs / Blacklist

To further fortify content security, HTML Executable allows you to create a blacklist of software programs that are not allowed to run alongside your ebook. This includes screen capture utilities, screen recorders, and file monitoring software, among others.

You can manage the blacklist by adding specific process names or keywords present in window titles. If any of the listed software programs are detected at runtime, your ebook will display an error message and terminate. You can configure regular checks, with the blacklist being verified every X seconds.

➢ To add a software program, click **Add** and choose either the process name or word/text in the window title. Enter the corresponding text to detect, ensuring it is case insensitive.


Block A Software Program

Forbid Software Program

Please give information about the software program you want to detect:

(●) Detect process name

( ) Detect word or sentence in window title

Process Name: screencorder.exe

- Import and export functionality is available through **XML Tools**, allowing you to easily manage the blacklist.
- The initial list is populated with default values from the `defforbidprog.xml` file located in the HTML Executable installation folder. You can customize these defaults by exporting your current list to the `defforbidprog.xml` file.
- To clear the list, click **Clear**.
- To reset the list to the default values from the `defforbidprog.xml` file, click **Reset with default list**.

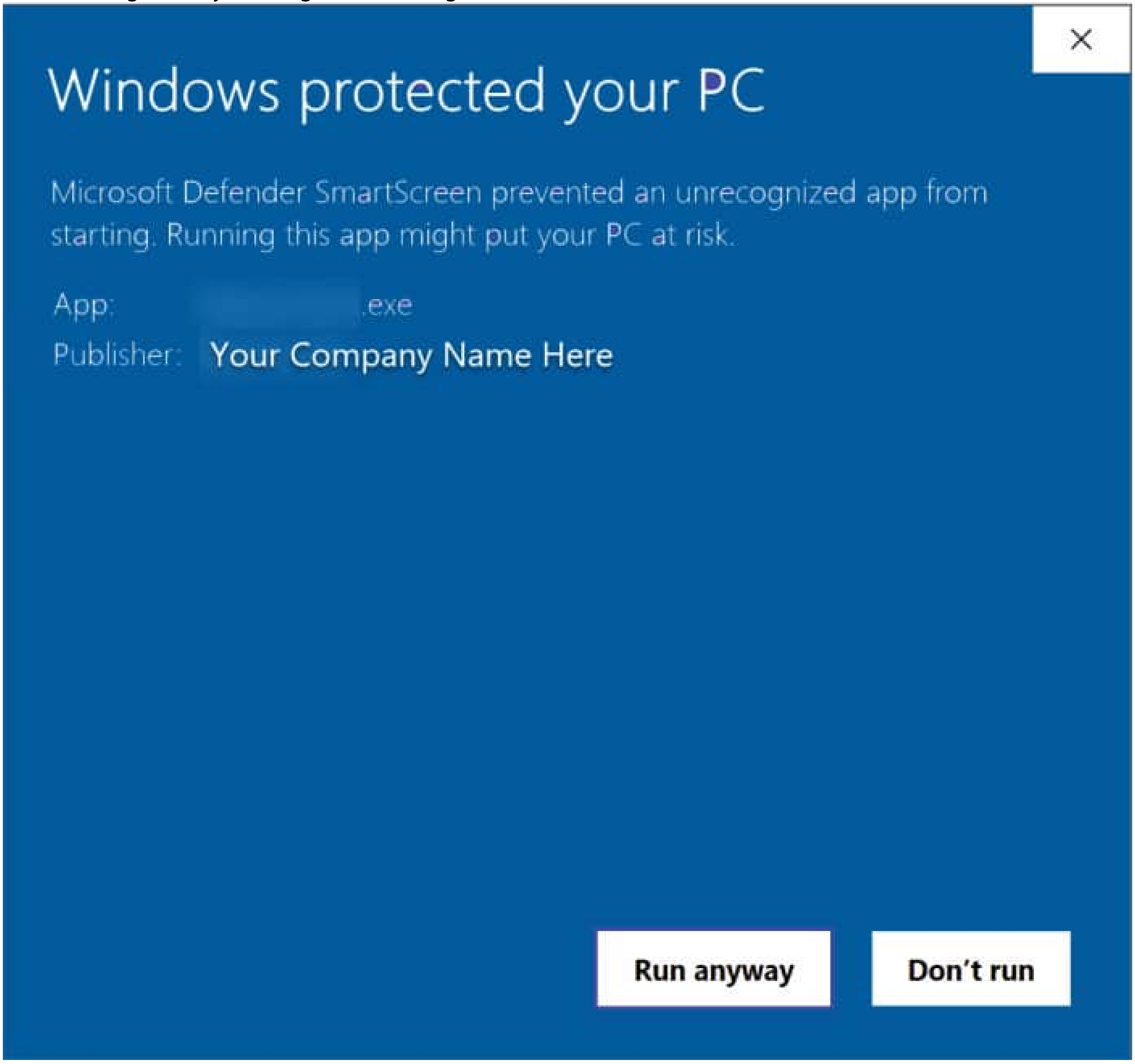
By leveraging these content protection features, you can ensure the integrity and security of your ebook, protecting your hard work and creativity from unauthorized use and distribution.

4.4.4. Code Signing (Digital Signatures)

Digitally signing your compiled publication or ebook .EXE file assures end users that the code has not been tampered with or altered since its release. Based on *Microsoft Authenticode® technology*, digital signing verifies the source of the code and its integrity. With HTML Executable, signing your .EXE files is straightforward, as it includes the necessary tools and supports modern signing methods like Azure Trusted Signing.

Benefits of Code Signing

For signed applications: End users see a certificate indicating the software's origin and authenticity, reducing security warnings and building trust:

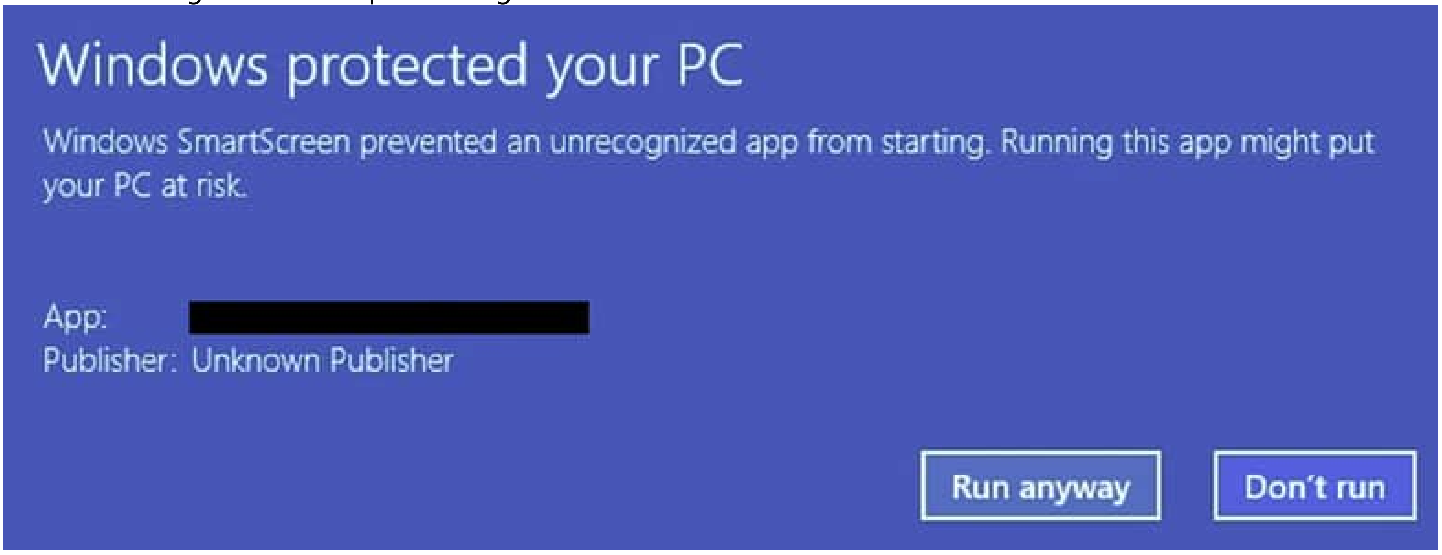

Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: .exe

Publisher: Your Company Name Here

For unsigned applications: Windows often displays a warning (like the SmartScreen filter), which may discourage users from proceeding:


Windows protected your PC

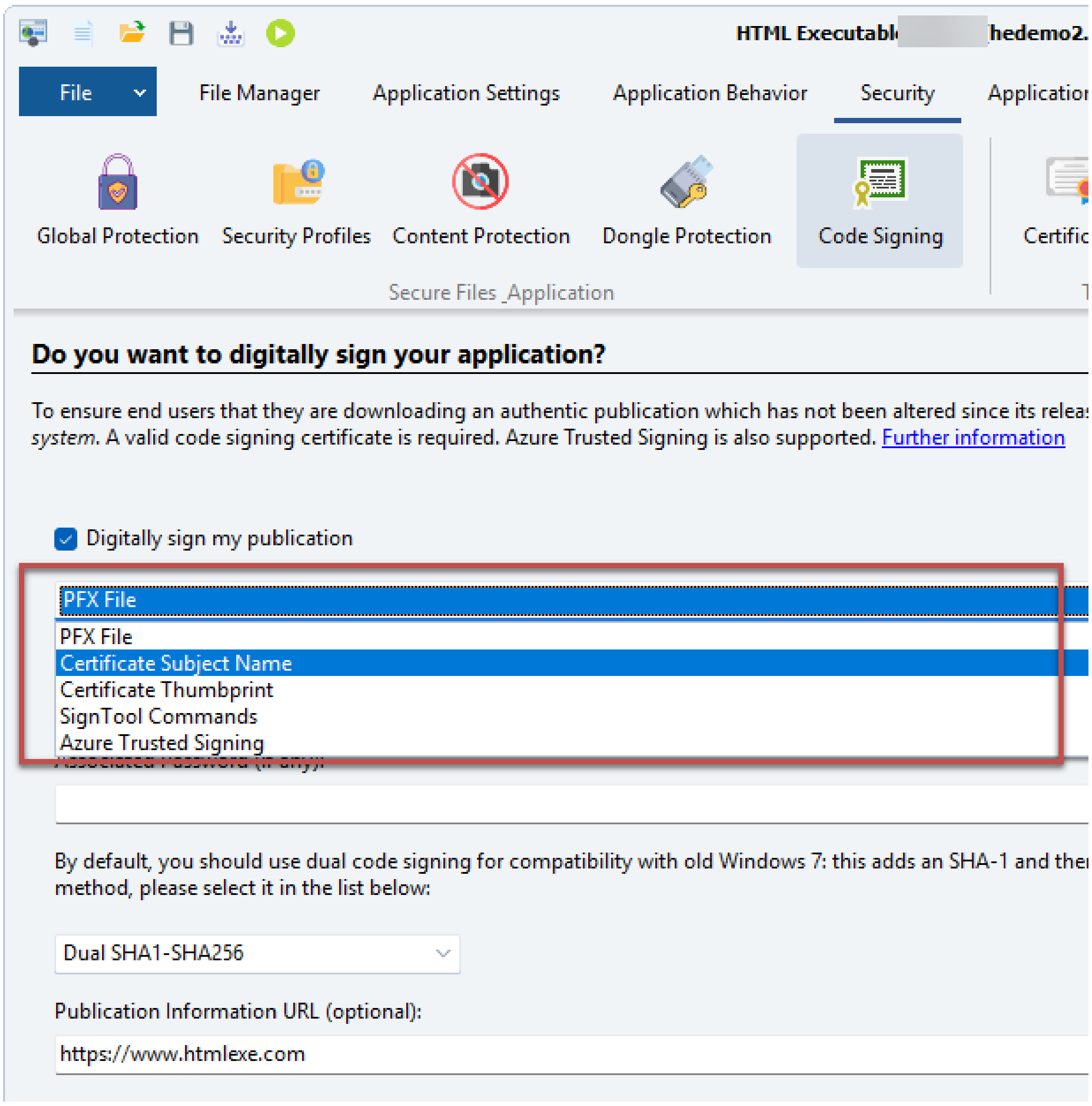
Windows SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: [redacted]

Publisher: Unknown Publisher

To enable digital signing in HTML Executable, go to **Security => Code Signing**, activate the "**Digitally sign**

my publication" option, and choose your signing method:



Do you want to digitally sign your application?

To ensure end users that they are downloading an authentic publication which has not been altered since its releas *system*. A valid code signing certificate is required. Azure Trusted Signing is also supported. Further information

☑ Digitally sign my publication

PFX File
- PFX File
- Certificate Subject Name
- Certificate Thumbprint
- SignTool Commands
- Azure Trusted Signing

By default, you should use dual code signing for compatibility with old Windows 7: this adds an SHA-1 and the method, please select it in the list below:

Dual SHA1-SHA256

Publication Information URL (optional):

https://www.htmlexe.com

Learn more about code signing with Authenticode in this Introduction to Code Signing.

Windows cannot sign EXE files larger than 2 GB. If your publication EXE exceeds this limit, consider options to reduce its size or use alternative distribution methods. See Output Format for more details.

How to Obtain a Code Signing Certificate

To sign your application, you need a valid code signing certificate from a trusted Certificate Authority (CA) such as Sectigo or Digicert. CAs offer different types of certificates, but only code signing certificates are compatible with Authenticode.
Traditionally, obtaining these certificates could be expensive. However, a more cost-effective solution is now available:

Azure Trusted Signing: Provided by Microsoft, Azure Trusted Signing offers several advantages, including potentially lower costs and no need for USB tokens for key storage. HTML Executable fully supports signing applications with Azure Trusted Signing.

Choose the signing method in HTML Executable:

- PFX File
- Certificate Subject Name (from local Certificate Store)
- Certificate Thumbprint (from local Certificate Store)
- **SignTool Commands** (for advanced scenarios such as cloud signing)
- **Azure Trusted Signing** (cloud-based signing)

Token-Based Certificates and Hardware Security Modules (HSMs)

As of June 1, 2023, industry standards require that code signing certificate private keys be stored on a hardware security module (HSM) or a token that meets FIPS 140-2 Level 2 (or Common Criteria EAL 4+) or an equivalent standard. This change aims to prevent misuse of stolen keys. The traditional PFX format for distributing private keys is being phased out for newly-issued public certificates.
HTML Executable supports signing with token-based certificates (when using PFX or Certificate Store options, if your token makes the certificate available this way). Ensure your token is connected during the signing process.

Steps to Sign Your Publication with a Code Signing Certificate (PFX / Store)

HTML Executable includes **GSignCode**, an integrated utility for signing publications. No third-party software installation is required for this method (unless your certificate provider requires specific drivers for a hardware token).

Configuring the Certificate

1) **Using a PFX File:** Select "PFX File" from the dropdown. Specify the path to your `.pfx` file and its associated password (if protected).
2) **Using the Windows Certificate Store:** Select "Certificate Subject Name" or "Certificate Thumbprint". Provide the respective identifier. HTML Executable will search for the certificate in the Current User or Local Computer store.
3) **Token-Based Certificates:** If your certificate is on a hardware token, ensure it's connected. If it's accessible via the Windows Certificate Store, use that option. GSignCode automatically adapts to the token's capabilities.

Using SignTool Commands

For advanced scenarios, you can instruct HTML Executable to use Microsoft's SignTool.exe.
Select **SignTool Commands** from the dropdown list.

You need to provide the command(s) that SignTool should execute. You can use placeholders:
{$OUTPUTFILE$}: Represents the full path to the executable file(s) to be signed.
{$OUTPUTFOLDER$}: Represents the path to the output directory.

Example command:
```
sign /a /fd SHA256 /tr "http://timestamp.digicert.com" /td SHA256 "{$OUTPUTFILE$}"
```

The path to SignTool.exe must be configured in Environment Options.

Using Azure Trusted Signing

HTML Executable seamlessly integrates with Azure Trusted Signing.

Select **Azure Trusted Signing** from the dropdown list.

You will need to provide:

Trusted Signing Account Endpoint: e.g., `https://eus.codesigning.azure.net` (choose your region).
Trusted Signing Account Name: Your Azure account name for trusted signing.
Certificate Profile Name: The name of your signing certificate profile in Azure.

Before using Azure Trusted Signing, ensure you have installed Microsoft Azure CLI and the Trusted Signing Client Tools. You must also be logged in via `az login`. The path to `Azure.CodeSigning.Dlib.dll` and `SignTool.exe` must be configured in Environment Options.

For a detailed guide on setting up Azure Trusted Signing, please refer to tutorials like this one.

Digest Algorithms

HTML Executable supports modern digest algorithms:

1) **SHA-256:** The default and recommended standard for security.
2) **Dual Code Signing (SHA1-SHA256):** This option combines SHA-256 and SHA-1 signatures to support older systems like Windows 7 or Vista, which may not fully support SHA-256 alone. When this is selected for GSignCode, it attempts to dual sign. On Windows 7, if dual signing is not explicitly chosen or possible, only SHA-256 is typically used by default for new signatures.

You can select the preferred digest algorithm from the dropdown list when using PFX/Store signing. For SignTool and Azure, the digest algorithm is usually specified in the command itself or handled automatically by the service.

Publication Information URL

Include a URL in your digital certificate to direct users to learn more about your product or company. If not specified, HTML Executable uses the default URL from the Icon / Version page.

Digital Signature Timestamp

A timestamp is added to your ebook or publication, ensuring that the embedded digital signature remains valid even after the signing certificate itself expires. Ensure that your system has an Internet connection during the signing process for time-stamping purposes.
Two timestamp servers are used: an Authenticode-compatible server and an RFC-3161-compatible server. You can configure their URLs in the Environment Options.

Troubleshooting Code Signing

If errors occur during code signing, refer to the compilation log for detailed messages. Ensure that your certificate is accessible and correctly configured, and verify your Internet connection for timestamping. For SignTool or Azure issues, ensure the respective tools are correctly installed and configured as per their documentation.

4.4.5. Dongle Protection

Protect your ebook or HTML application software with dongle protection! A dongle is a small hardware device that acts as an electronic "key" for your software. When using HTML Executable, you can **lock your ebook or publication to a dongle, ensuring that it will only run when the correct dongle is plugged in**.

By utilizing dongles, your users can run your ebook or publication on any computer while preventing unauthorized sharing or copying. Each time the publication is run, it will check for the presence of the dongle. If the right dongle is detected, the publication will open; otherwise, it will not run.

Compatible Dongles

HTML Executable currently supports Enky LC2 and Enky SL dongles distributed by HS-Security Ware GmbH. For more information about these dongles, visit their respective pages: Enky LC2 and Enky SL. To inquire about demonstrations, make purchases, or ask questions, please contact HS-Security Ware directly.

If you need support for other dongle models, please contact us.

Enky LC2 Dongles

The Enky LC2 dongle is a cost-effective software protection dongle with a matured 8-bit chip. It is a standard USB interface-based HID device that can be used without requiring drivers on Windows.

Configuration Steps for Enky LC2 Dongles

1) Turn on "Link the publication to a dongle."
2) Select the "Enky LC2 dongle" model from the list.
3) Enter the **Developer ID** received from HS-Security Ware when purchasing the dongles.
4) Enter a unique **Product ID** associated with your publication or ebook, ensuring that only Enky LC2 dongles with the correct product ID can be used.
5) Compile your publication.

Configuring an Enky LC2 Dongle

To link the dongle to your publication, you need to burn it first. Connect the dongle and click the "**Burn Dongle Now**" button. Once configured, you can distribute the dongle along with the publication EXE file to unlock your publication for customers.

⚠ Note: HTML Executable can automatically handle the burning process during compilation if a compatible dongle is plugged in.

Background Check

To ensure the protected publication or ebook runs only with the correct dongle, a background check can be enabled. The dongle will be **checked every X seconds**, freezing the publication if the dongle is not available. Users can choose to reinsert the dongle or exit the publication.

You can specify the X interval in seconds for the background check or leave it blank to disable. For example, setting it to "15" will check the dongle every 15 seconds.

Customizing Error Messages

You can customize error messages related to the dongle using the resource strings in the Localization page. The following resource strings are used: SDongleInvalidProductID, SDongleAPIError, SDongleNotFound, SDongleHardwareError, SDongleIncompatible, SDongleInvalidDongle, SDongleInvalidCheck.

Enky SL Dongles

The Enky SL dongle is a smart card-based security dongle designed for software licensing, software copy protection, and software piracy protection.

Configuration Steps for Enky SL Dongles

1) Turn on "Link the publication to a dongle."
2) Select the "Enky SL dongle" model from the list.
3) Enter the **Developer ID** received from HS-Security Ware when purchasing the dongles.
4) If you haven't received the product ID, you can generate one. Connect the dongle, click "**Modify**," and enter the seed data used to compute a secure and unique product ID. This product ID is also stored on the dongle.
5) Enter a unique **Application ID** associated with your publication or ebook, providing an additional security layer.
6) Choose the "License Index Module" to be used for your publication. Enky SL dongles support up to 256 different license modules.
7) Optionally, assign a maximum execution count, maximum number of execution days, or fixed expiration date for a given license module.
8) Compile your publication.

Configuring an Enky SL Dongle

Similar to Enky LC2 dongles, you need to burn the dongle to link it to your publication. Connect the dongle and click the "**Burn Dongle Now**" button. Once configured, you can distribute the dongle along with the publication EXE file to unlock your publication for customers.

⚠ Note: HTML Executable can automatically handle the burning process of Enky SL during compilation if a compatible dongle is plugged in.

Background Check

To ensure the protected publication or ebook runs only with the correct dongle, a background check can be enabled. The dongle will be **checked every X seconds**, freezing the publication if the dongle is not available. Users can choose to reinsert the dongle or exit the publication.

You can specify the X interval in seconds for the background check or leave it blank to disable. For example, setting it to "15" will check the dongle every 15 seconds.

Customizing Error Messages

You can customize error messages related to the Enky SL dongle using the resource strings in the Localization page. The following resource strings are used: SDongleInvalidProductID, SDongleAPIError, SDongleNotFound, SDongleHardwareError, SDongleIncompatible, SDongleInvalidDongle, SDongleInvalidCheck, SDongleLicenseError, and SDongleLicenseExpired.

Other Security Features

Dongle protection is compatible with other security features and trial options of HTML Executable.

4.5. Application Output

The ribbon "Application Output" in HTML Executable offers general settings for generating the final ebook and publication EXE file.

The following topics explain these features:

- Output Settings
- Output Format
- Portable Ebooks and Publications
- Loading Screen
- EXE Icon / Version
- Create Installer
- Web Update Settings

4.5.1. Output Settings

Configure the output settings for your ebook or publication made with HTML Executable. The output file is the executable file that you will distribute to your end users.

Output Path

In the "Output Path" field, specify the full path to the final publication file (directory+filename). This file should have an executable (.exe) extension.

Notes

- Save your project before compiling the publication.
- The publication file will be overwritten if it already exists.
- If the output folder doesn't exist, it will be created.
- Ensure that the output folder is not read-only.
- All resources and source files must be available during compilation.
- Make sure you have enough free disk space.

Publication Title

Provide a short and descriptive title for your publication. This title will be used in message boxes and window title bars, and it will appear in the Windows taskbar and task manager.

Publication GUID

As ISBN identifies books, the publication GUID allows the ebook / publication to store its settings on the end user's computer. To generate a new GUID, you can press the right button.

⚠ Note that you should not change a GUID once your project is started.

ⓘ If you want to share the same settings between different publications, give them the same GUID.

Reset Settings

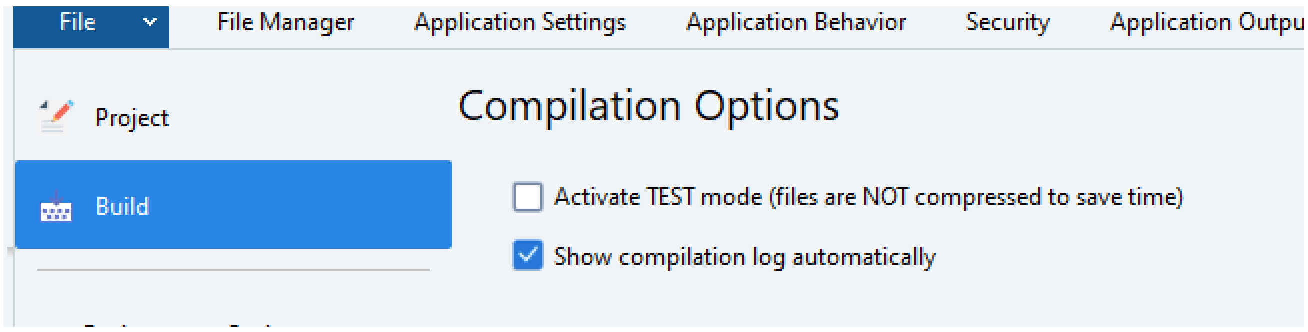
If you want to remove or reset the publication's settings stored on your computer, click "Reset Settings" and confirm.

Export Cleanup Utility

Generate a small cleanup program to remove traces left by the publication on different computers. This portable program can be run from a USB disk on any computer.

⚠ The generated program is specific to the current project and should not be distributed to others.

Compilation Options



Compilation Options

[ ] Activate TEST mode (files are NOT compressed to save time)

[✓] Show compilation log automatically

Activate TEST mode: Use this option for testing purposes only. It disables file compression, optimizations, and code signing to speed up the compilation. Publications built in test mode should never be distributed.

Show compilation log automatically: this will display the compilation log once the build operation is finished.

Compiling Your Ebook or Publication

Launch the compilation of your package using one of these shortcuts:

• Click in the toolbar.
• Use the menu commands from the Build menu.
• Press F9 for normal build or F10 for a full build with file compression.
• Use command line parameters.

Archive Caching: HTML Executable compresses files into a single archive during compilation. To speed up subsequent builds, the archive is cached. If the source files or compression options change, the archive is reconstructed. You can force a full rebuild by pressing F10 or selecting the "Build full publication" menu command.

The cached archive is automatically deleted when you close HTML Executable or switch to another project.

Refer to the Environment Options for more details on compression settings.

4.5.2. Output Format

HTML Executable is a powerful tool that allows you to transform your websites, ebooks, and PDF documents into stand-alone Windows applications in .EXE format.

Stand-Alone EXE Files with No Dependencies

HTML Executable is designed to generate **stand-alone Windows applications**. Your website is neatly packed into a **single .EXE file**, ready for distribution. This .EXE file **does not require any third-party software** such as the Microsoft .NET Framework or external DLLs. However, the code necessary to run stand-alone EXE files **takes about 15 MB uncompressed**. If size is not a concern for you, stick with the stand-alone option.

Compressing Executable Files for Easy Distribution

Planning to distribute your publication .EXE file on the Internet? We recommend compressing it to reduce its size and download time. Here are some ways to do it:

Compress the final .EXE file with UPX: UPX is a free executable packer available at https://upx.sourceforge.net. HTML Executable can call UPX automatically if you enable this option.

Note: For legal reasons, UPX is not shipped with HTML Executable. You have to download the program from https://upx.sourceforge.net and unzip the archive into the "UPX" subfolder of your HTML Executable installation directory (e.g., C:\Program Files (x86)\HTML Executable XXXX\UPX, where XXXX is the version).

Distribute the final .EXE file in an installer: You can use the installer creator Paquet Builder to compress your .EXE file into an installer or Setup program, which is optimized for online distribution.

See Create Installer.

Protect your EXE file with a third-party EXE compressor or protector: Stand-alone executable files made with HTML Executable can be secured with third-party software protection systems.

Managing Publication Data and Search Index Data

HTML Executable creates a single EXE file by default. However, if you have large source files, you might easily reach the EXE size limit (**4 GB**). To bypass this limit, you can enable **Keep publication data outside the EXE file**. HTML Executable will create two files: your EXE file and a second file with the same name, but a different extension (.hedata).

Warning: The companion data file (.hedata extension) is bound to the primary publication EXE file. If you try to replace the .hedata file with another, an error will occur, and the program will close.

If you have a large amount of source files, the search index data might exceed the free memory limit available for 32-bit programs (**2 GB**). To address this problem, enable **Keep the search index data outside the EXE file**. HTML Executable will store the search index in two companion files with the same name as your publication EXE, but different extensions (.searchi.dat and .searchx.dat).

⚠ **Warning**: The two companion files are bound to the primary publication EXE file and must be distributed with it. Otherwise, the search engine will not work.

Managing Chromium Embedded Framework Runtime Files 💻

If you enable "Do not compile CEF files into the EXE", HTML Executable will not compile the mandatory Chromium Embedded Framework runtime files into your application EXE file (saving approximately 70 MB in size). However, the application will not work on a computer if these files are not found. The following error message will be shown: `missing CEF3 files!`

You can use this option only if you are sure to install these files yourself. For instance, these files can be set up automatically if you use the hecefruntimeXXXX.exe installer (where XXXX is the version) available in the Redist subfolder of the HTML Executable installation. Alternatively, you can find them in the CEFRuntime subfolder of the HTML Executable installation. The files must be made available in the shared cache folder on the end user's computer.

Do Not Cache CEF Files Locally

Since the Chromium Embedded Framework runtime files have a large size (approximately 240 MB uncompressed), it normally takes several seconds for the application to decompress them to memory at startup. To avoid this decompression step and gain loading time, the application will decompress and **store Chromium Embedded Framework runtime files into a shared cache folder on the end user's computer the first time it is run**. If the runtime files already exist in the cache, they are used directly, and no decompression is necessary.

If you do not want your application to store Chromium Embedded Framework runtime files (for instance, to save disk space on the end user's computer), you can enable "**Do not cache CEF files locally**". However, the application will take longer to load each time it is run.

The cache folder is the same for all applications made with HTML Executable. By default, the following cache folder path is used: C:\ProgramData\GDG Software\HTML Executable App Cache\CEFXXXX, where XXXX represents the version number of the Chromium Embedded Framework used by HTML Executable.

A different cache folder name will be used if you configured a custom name for the application's storage folder.

4.5.3. Portable Ebooks and Publications

HTML Executable offers the unique capability to create **portable versions of your publications and ebooks**. With a portable publication:

- No installation is required, your publication is stored on a removable device such as a USB flash drive/stick, enabling it to be used on multiple computers.
- User preferences and publication settings are stored with the software (i.e., they are written to the USB drive). Thus, your users will keep their preferences even if they start the publication on another PC. The Windows registry is not used.

Portable publications create one or two **data file(s)** at least in addition to their EXE file:

- The state file (user preferences, global variables...) is named: `[name of the EXE].userpref`
- The license file (trial and registration information) is named: `[name of the EXE].license`

These files are saved in the same folder as the EXE file. Consequently, the storage device **should not be read-only**. If the publication is unable to write the files in the same folder as its EXE file, it will use the default location on the hard disk. The default location is a subfolder in the User Data directory that you may customize.

If you don't want the publication to write the state file, enable "**Do not create a user preference file**". This is useful if nothing should be written to the storage device.

If you do not create a portable version, you may now specify the **name of the folder where a publication should store its settings**. By default, it will be a subfolder in the User Data directory. You can obtain the full path at runtime from the global variable HEPubStorageLocation.

ⓘ You can explore the local storage folder by clicking the corresponding button.

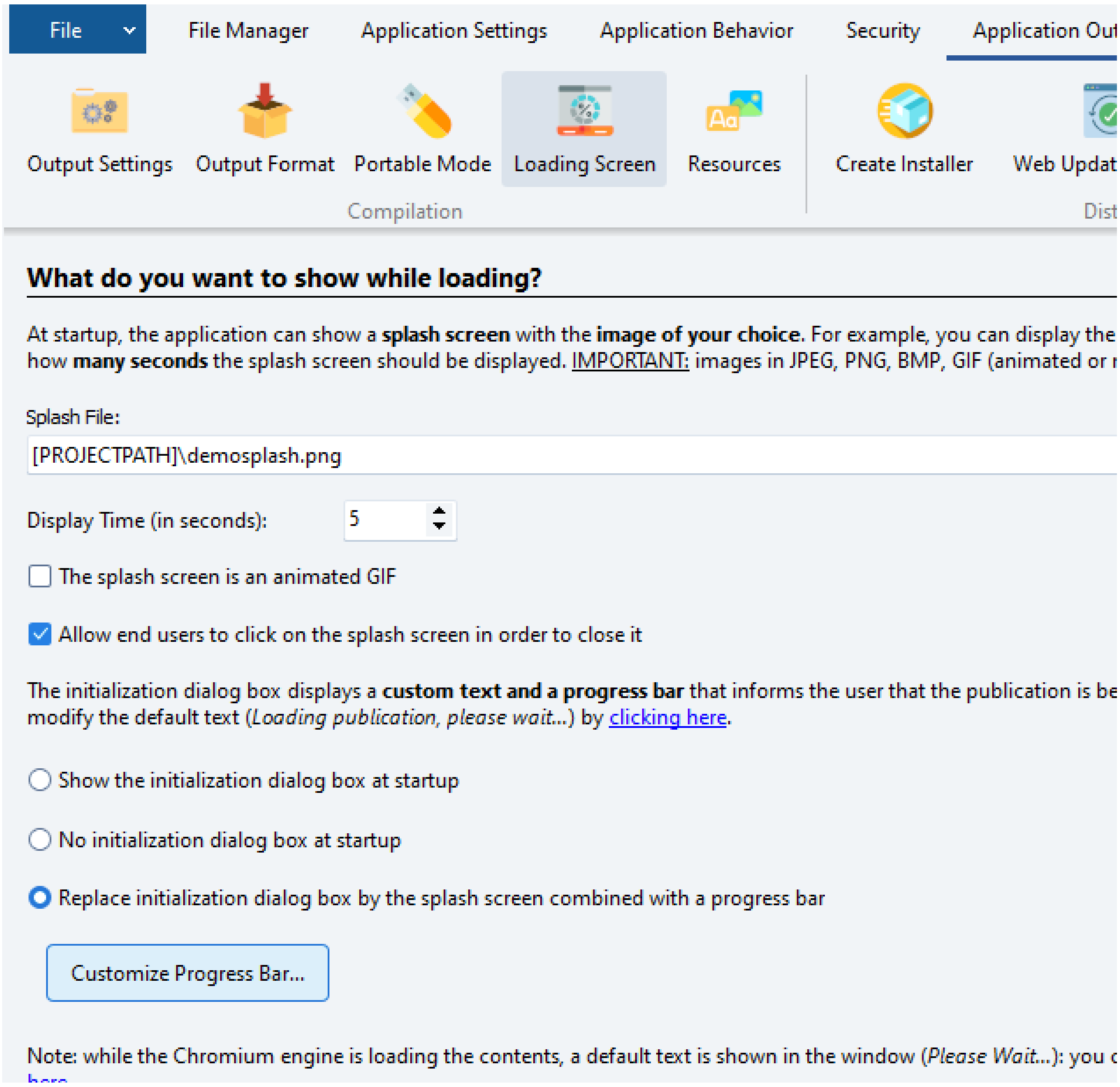
⚠ **Warning**: Avoid using expiration features for a portable publication. In fact, portable publications store their settings on the USB disk. Consequently, trial settings are saved on the disk too: an end user could easily reset his trial period by removing the settings files. For portable publications, you should forbid the entire access or use Security Profiles to partially lock your publication.

Want to lock a Portable Publication to a Specific USB Disk?

4.5.4. Loading Screen

Customizing Loading Screen for HTML Executable Applications

During their initialization phase, applications created with HTML Executable can show a **splash screen** (an image briefly displayed at the beginning), and/or a **"Please wait..." dialog box**. These features can enhance the user experience, making the startup phase more interactive while informing users that the application is being initialized.



What do you want to show while loading?

At startup, the application can show a **splash screen** with the **image of your choice**. For example, you can display the
how **many seconds** the splash screen should be displayed. IMPORTANT: images in JPEG, PNG, BMP, GIF (animated or r

Splash File:
[PROJECTPATH]\demosplash.png

Display Time (in seconds): 5

☐ The splash screen is an animated GIF

☑ Allow end users to click on the splash screen in order to close it

The initialization dialog box displays a **custom text and a progress bar** that informs the user that the publication is be
modify the default text (*Loading publication, please wait...*) by clicking here.

○ Show the initialization dialog box at startup

○ No initialization dialog box at startup

● Replace initialization dialog box by the splash screen combined with a progress bar

Customize Progress Bar...

Note: while the Chromium engine is loading the contents, a default text is shown in the window (*Please Wait...*): you c
here

Splash Screen

The splash screen is a great way to display your company's logo or any imagery related to your application's content.

To enable the splash screen, **you must specify the image file to be used** (absolute path required) and determine how long it should be displayed (in seconds).

HTML Executable supports a variety of image formats including PNG, BMP, GIF (including animated GIFs), JPEG, and TIF.

Don't forget to tick the option named "The splash screen is an animated gif" if you choose an image that is an animated GIF, otherwise it will be displayed statically.

HTML Executable supports non-rectangular alpha-blended (semi-transparent) splash screens if you use 32-bit PNG files. This can give a unique look to your application.

Additionally, you have the option to **allow users to close the splash screen by clicking on it**, which is generally recommended for a better user experience.

Please take note of the following:

- Try to use small splash screens: large splash screens may take longer to draw because, in some cases, data needs to be decompressed first (like for PNG or JPEG).
- The splash screen file is not stored in the project file. It should be available as an external file when the application is being compiled. Path variables like **[PROJECTPATH]** are allowed.

"Please Wait" Dialog Box

The "Please wait..." dialog box is another feature to enhance the user experience. It displays a custom text of your choice, such as "Loading application, please wait...".

To change the displayed text, navigate to the Language tab and modify the resource string: `SLoadingPublication`.

You can also opt not to display this dialog box by enabling the "**No initialization dialog at startup**" option.

"Please Wait" Panel at Startup (CEF only)

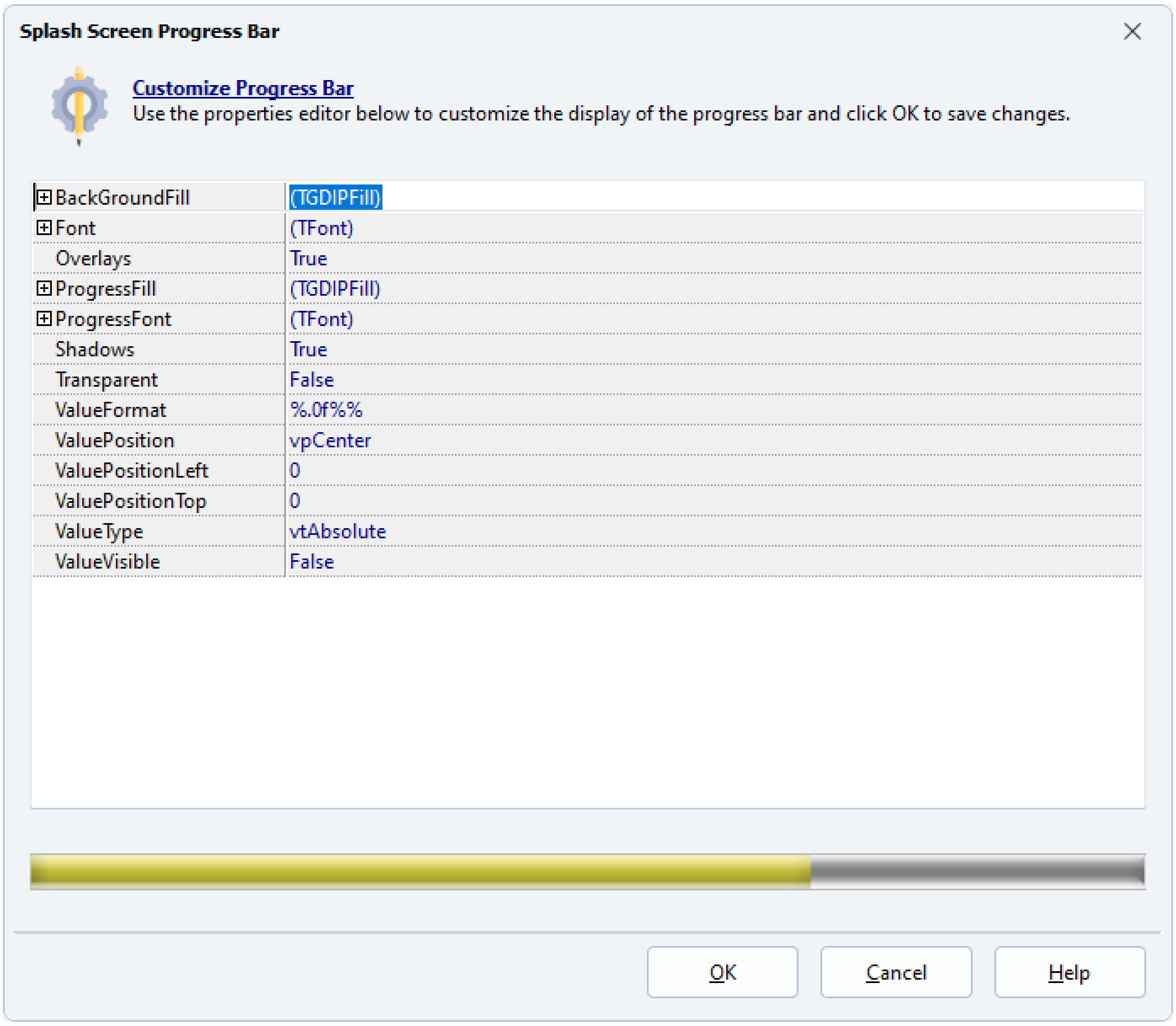
Instead of a blank window at startup, HTML Executable can show a panel bearing a "Please wait" text while the application's contents are loading. You can customize this text by editing the resource string named `SCEFPleaseWait`.
Only for the CEF rendering engine.

Splash Screen with Progress Bar

The splash screen can also serve as a progress indicator while loading the application. With this option, a progress bar is displayed at the bottom of the splash screen, letting users know that the application is busy loading.

This option is not compatible with "Display Time". Moreover, the splash screen in progress mode can't be closed by users.

You can customize the progress bar's colors by clicking on **Customize Progress Bar** and editing the available properties.


Splash Screen Progress Bar

Customize Progress Bar
Use the properties editor below to customize the display of the progress bar and click OK to save changes.

⊞ BackGroundFill	(TGDIPFill)
⊞ Font	(TFont)
Overlays	True
⊞ ProgressFill	(TGDIPFill)
⊞ ProgressFont	(TFont)
Shadows	True
Transparent	False
ValueFormat	%.0f%%
ValuePosition	vpCenter
ValuePositionLeft	0
ValuePositionTop	0
ValueType	vtAbsolute
ValueVisible	False

4.5.5. EXE Icon / Version

Customizing Icons and Version Information in HTML Executable Applications

When compiling your HTML application or ebook, HTML Executable generates a **single executable (.EXE) file that's ready to be distributed**. A standard feature of most Windows executable files (PE format) is a **resource section** that houses important assets like version information, icons, cursors, string tables, and more. HTML Executable provides you with the ability to modify some of these resources, allowing you to incorporate your own logo and copyright information.

Changing the EXE File Icon

HTML Executable offers the capability to **replace the default icon of the executable file**. By indicating the **path to an alternative icon file** (which must be a proper icon file with a .ico extension), you can customize your application's icon to suit your needs. HTML Executable can support any icon image, whether it's 32x32 with 16 colors or 48x48 with 256 colors.

For straightforward icon creation or extraction, consider using GConvert, a companion tool that facilitates extracting icons or converting images into icons.

Icons with 32-bit PNG data are also compatible.

Remember, the icon file should be accessible as an external file during the compilation of the publication. Path variables such as **[PROJECTPATH]** are permitted. You can find some usable icon files in the **Resources subfolder** of HTML Executable.

Adding Custom Version Information

The version information in an executable file is a specialized resource section that holds critical details about the file, such as its version number, targeted operating system, original filename, copyright information, and more. This information is integrated into the compiled code, and users can view it by right-clicking the program icon and selecting **Properties** (or by pressing ALT+ENTER in Explorer).

HTML Executable lets you inject your personalized version information into the EXE file. This includes:

- **Company Name**: The name of your company.
- **Legal Copyright**: Your own copyright statement, such as "Copyright 2023 by Your Company. All rights reserved."
- **File Description**: A description of your application's content.
- **Web Homepage**: A URL to your company's website or homepage.
- **Version number**: The current release number of your .exe file. The format must be **X.X.X.X** where X represents a numerical value, such as 1.20.34.45.
- **Product number**: The release number of your application, following the same format as the version number.
- **Legal Trademarks**: Text that will appear in the "Legal Trademarks" field. This field is editable only if you've purchased the "No Branding" option.

These pieces of information, along with the default logo file, can be configured in the Environment Options. With this setup, your preferred settings will be applied every time you create an application.

In conclusion, HTML Executable provides you with the flexibility to tailor the ebook's appearance and information of your HTML applications. With simple .EXE customization, your application can effectively represent your brand and provide essential details to your users.

4.5.6. Create Installer

HTML Executable generates single executable files ready for delivery. You just need to distribute the application .exe file to your users/customers. But after having built your application, you may also want to distribute it over the Internet: you may want to package your compiled application into Zip archive or even an **installer**. Installers are interesting because they can install several applications, additional files like readme, create shortcuts in the Windows Start menu, and offer an uninstaller to let end users remove any trace of your application from their computer.

Generate an Installer

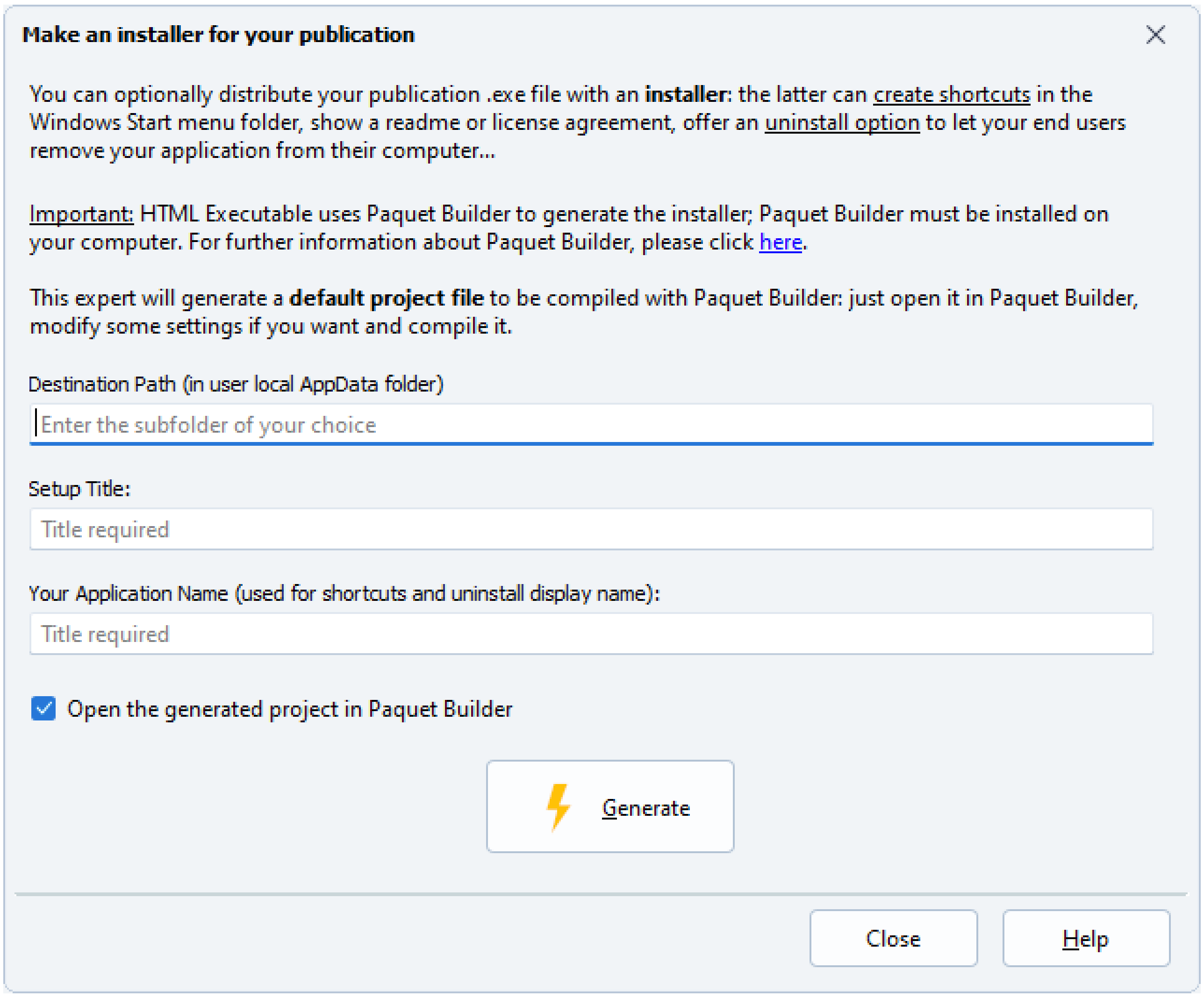
HTML Executable uses our software program **Paquet Builder** to generate custom and compact Installer programs for publications. It must be installed on your computer before you can use this feature.

Paquet Builder is a mix between a 7z Self-Extracting archive maker and an Installer routine generator. Thanks to its exhaustive feature set, you can create flexible and compact self-extractors for professional file and software delivery. Package up any document or program files, visually construct simple or sophisticated multi-language distribution and installation packages; generate updates and patches; wrap multimedia presentations or several Windows Installer MSI setups into single .exe files ready for delivery over the Internet.

Learn about the installer creator Paquet Builder

How does it work?

You will get a window with these fields:



Make an installer for your publication

You can optionally distribute your publication .exe file with an **installer**: the latter can create shortcuts in the Windows Start menu folder, show a readme or license agreement, offer an uninstall option to let your end users remove your application from their computer...

Important: HTML Executable uses Paquet Builder to generate the installer; Paquet Builder must be installed on your computer. For further information about Paquet Builder, please click here.

This expert will generate a **default project file** to be compiled with Paquet Builder: just open it in Paquet Builder, modify some settings if you want and compile it.

Destination Path (in user local AppData folder)

Enter the subfolder of your choice

Setup Title:

Title required

Your Application Name (used for shortcuts and uninstall display name):

Title required

☑ Open the generated project in Paquet Builder

First, you need to fill in the three fields **Destination Path**, **Setup Title** and **Your Application Name**: "Destination Path" is the default folder where you want your ebook to be installed, "Setup Title" is the title that will appear on all Installer windows, and the last one is obvious.

Secondly, press **Generate** to create the project. HTML Executable will then launch Paquet Builder to let you modify the new project.

HTML Executable will create a default project for Paquet Builder: you can then edit this project with the installer creator and of course compile it to create the installer package.

5. For Developers

5.1. Extend Functionality with HEScript

Introduction to Scripting with HEScript - Enhancing Functionality

Dive into the dynamic world of HTML Executable ebooks and publications! These creations are not just static pages; they are powered by scripts and feature a **built-in script engine**. Each publication is governed by a suite of scripts, generated and compiled into p-code by HTML Executable. When a publication is launched, it displays an interactive web browser-like environment where your end users can explore your HTML pages and interact with your content.

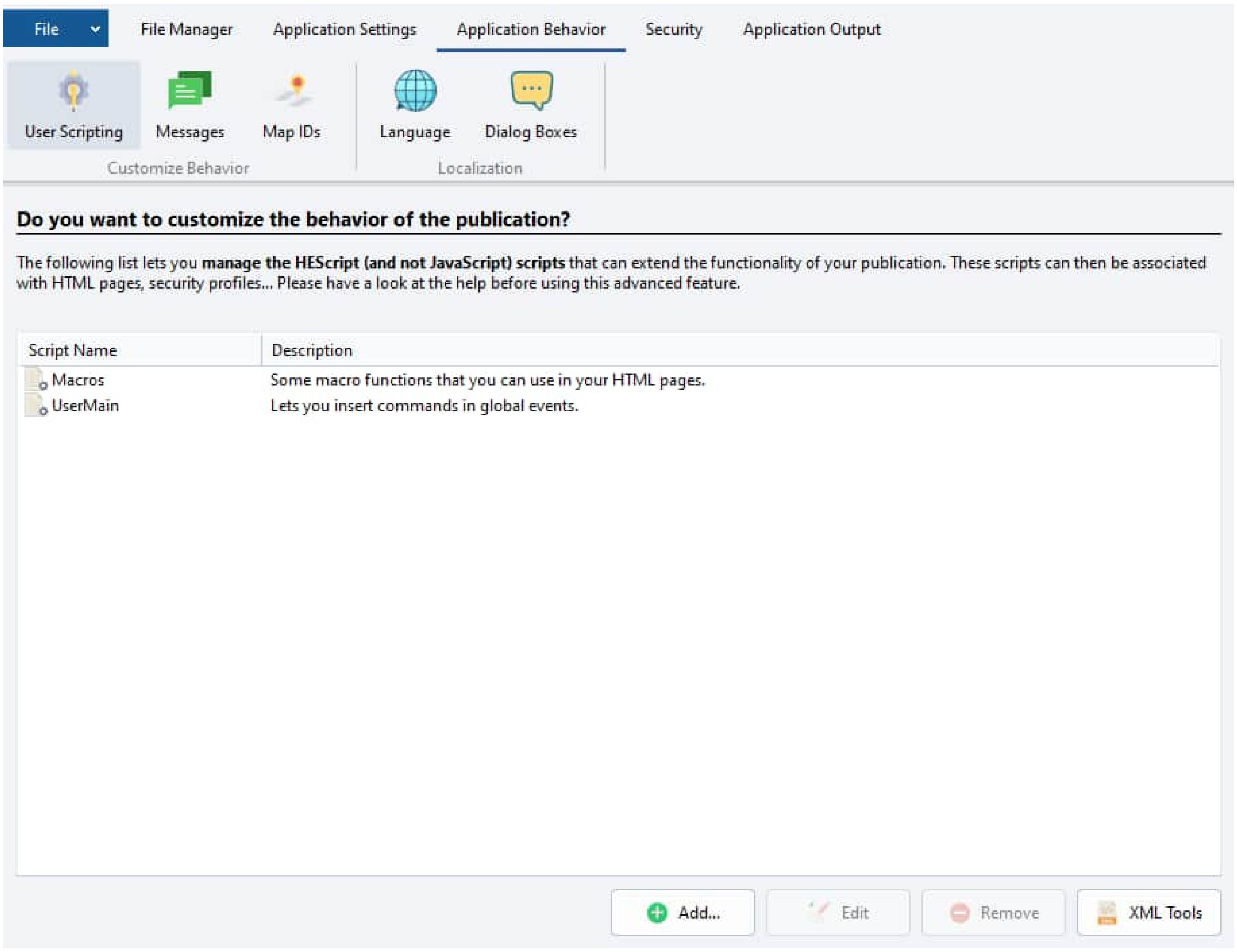
This dynamic nature allows you to **extend the functionality of your publications** by writing and invoking your own script functions. This powerful feature opens up a world of possibilities, enabling you to customize your publications to your heart's content.

Note: You don't need to delve into scripting to use HTML Executable and compile publications. Scripting is an advanced feature designed for those who wish to exercise full control over their publications.

Unveiling HEScript - The Script Language

The scripting language employed by HTML Executable is known as **HEScript**. It's built on the Object Pascal language syntax (akin to Embarcadero® Delphi and FreePascal) with a few minor modifications.

Unlike JavaScript, HEScript functions cannot be written directly into HTML pages. This is because HEScript scripts need to be compiled into pseudo-code first. Therefore, you'll need to use the User Script Manager to write and manage your scripts.



Do you want to customize the behavior of the publication?

The following list lets you **manage the HEScript (and not JavaScript) scripts** that can extend the functionality of your publication. These scripts can then be associated with HTML pages, security profiles... Please have a look at the help before using this advanced feature.

Script Name	Description
Macros	Some macro functions that you can use in your HTML pages.
UserMain	Lets you insert commands in global events.

☑ Each script that will be compiled into the publication's script collection is **listed**. You'll see the **name** of the script along with an optional **description**. Each script must have a **unique** name (similar to a namespace).

Here's a simple script example:

```
 // UserMain
 // This script contains special functions related to some of the events triggered by the publication.
 // You can then optionally add your own commands to these functions.

function OnBeforeNavigate(NewURL, TargetFrame: String): Boolean;
begin
 //Before the publication displays a page. Set Result to True to stop the operation.
 Result := False;
end;

procedure OnNavigateComplete;
begin
 // When a page has been displayed.
end;
```

☑ A script file is essentially a **collection of procedures or functions**. Each script file has a unique name; each procedure/function also has a unique name within its script file, following these rules:

- Only alphanumeric characters may be used; no spaces.
- Each name must be unique within a given script file, but you can have two different script files that

contain procedures or functions with the same name.

Key Points to Remember:

- Script files function like namespaces. When you call a procedure/function or assign it to an event, you'll need to specify the **script name**, followed by a **dot** (.), and then the **name** of the procedure/function, i.e., `[scriptname].[functionprocedurename]`. For example: `UserMain.OnNavigateComplete`
- Each script is managed by an **independent script engine**. Local variables are confined to their respective functions/procedures. To share data *between different HEScript files* or to *persist data across application sessions*, you'll need to use global variables.
- Using local variables within functions and procedures is a standard and recommended programming practice for managing data temporarily and maintaining code clarity.

Please see these topics too:

- Extend Functionality with HEScript
  - HEScript Script Editor
  - HEScript Function Reference
  - Run and Call HEScript From JavaScript And HTML
  - UserMain Script And Script Templates
  - Quickly Add HEScript Code
  - Sample Scripts
    - How to prompt end users for their name once and store it?
    - How to password protect pages?
    - How to call a javascript function from HEScript (toolbar, menubar)?
    - How to call DLL functions?
    - How to open an external file
    - How to open subfolders of a given folder or disk?
    - How to prompt for a password for closing ebook?
    - How to run an executable file or app?
    - How to dynamically modify the Table of Contents?
    - samplescript3
    - How to open and save files
- HTML Executable JavaScript API
- Special Targets for External Links
- Application Global Variables
- Publication Command Line Arguments
- Command Line - Directives
- customhtmldlg
- technicalnotes
- About Cookies

5.1.1. HEScript Script Editor

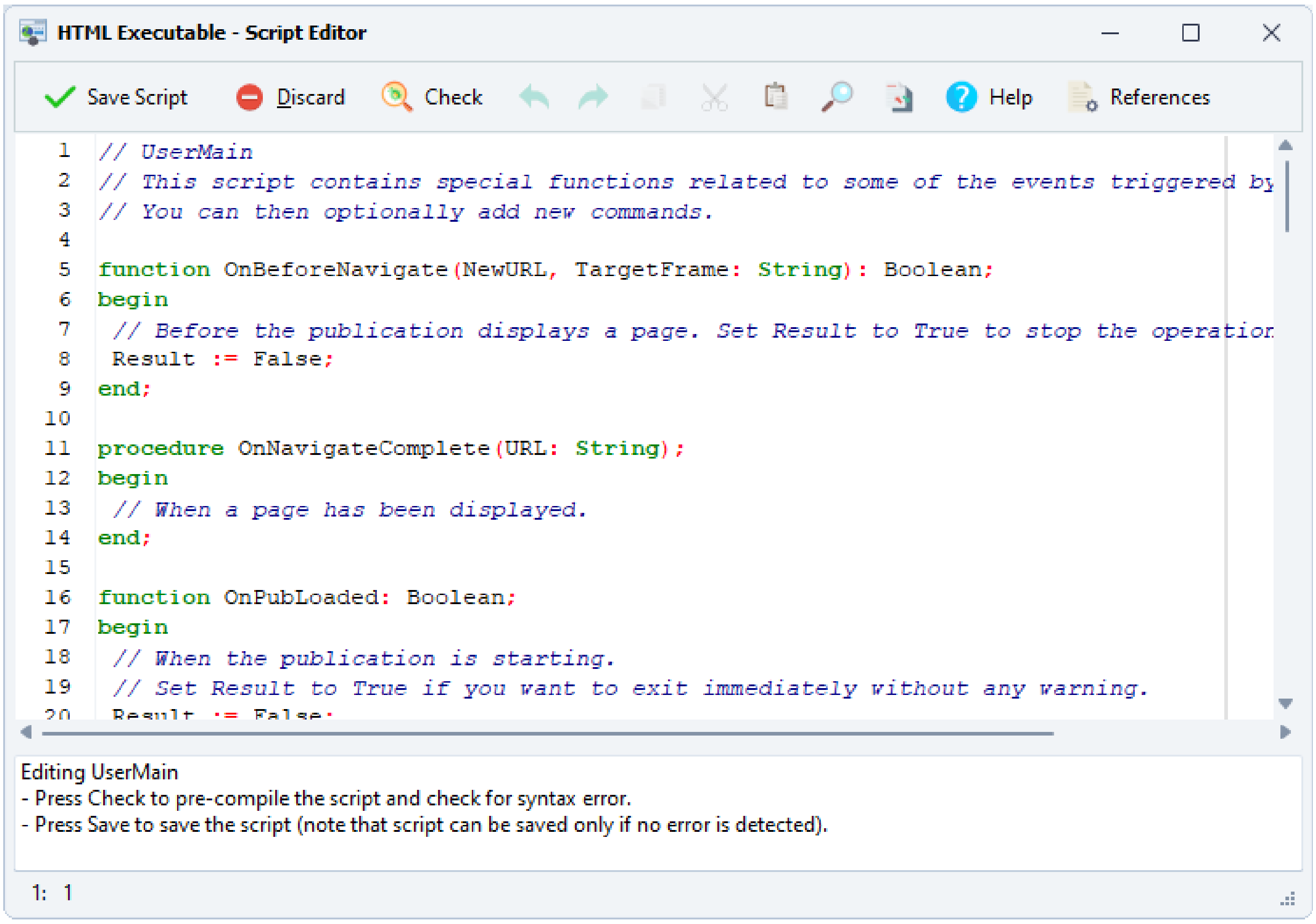
Using the HTML Executable Script Editor

☑ The Script Editor is your canvas for creating, removing, importing, and exporting HEScript scripts with the Script Manager.

☑ To dive into a script, simply select it in the Script Manager and click **Edit**. You can also double-click on the desired script for quick access. The script editor will also automatically appear when you import a script.

Alert: The script editor is your friend, but it's not foolproof. Just because a script is syntax-error-free doesn't mean it will work perfectly! Always test your ebooks and publications to ensure your scripts are functioning as expected!

Here's what the script editor looks like:



```
// UserMain
// This script contains special functions related to some of the events triggered by
// You can then optionally add new commands.

function OnBeforeNavigate(NewURL, TargetFrame: String): Boolean;
begin
 // Before the publication displays a page. Set Result to True to stop the operation
 Result := False;
end;

procedure OnNavigateComplete(URL: String);
begin
 // When a page has been displayed.
end;

function OnPubLoaded: Boolean;
begin
 // When the publication is starting.
 // Set Result to True if you want to exit immediately without any warning.
 Result := False;
```

```
Editing UserMain
- Press Check to pre-compile the script and check for syntax error.
- Press Save to save the script (note that script can be saved only if no error is detected).
```

1: 1

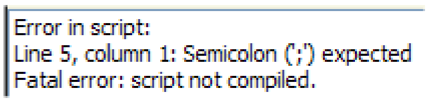
The script unfurls in the main edit box, ready for your modifications.

Understanding the Behavior

☑ The script editor is smart - it automatically highlights the syntax. Pascal keywords stand out in bold, while comments take a backseat in italic.

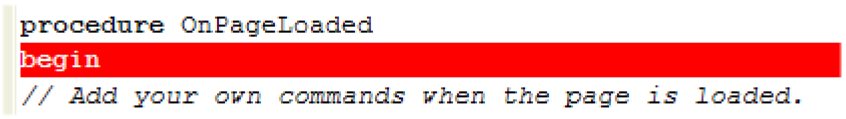
☑ Feel free to modify the script as you see fit. To check for errors, click **Check** in the toolbar. This pre-compiles the script to catch any syntax errors, making it easy to spot and fix issues.

If the script is error-free, you'll see the message: "Script successfully compiled". If there's an error, you'll receive an error message with the reason and location in the source code:



```
Error in script:
Line 5, column 1: Semicolon (';') expected
Fatal error: script not compiled.
```

The line containing the error will be marked in red:


```
procedure OnPageLoaded
begin
// Add your own commands when the page is loaded.
```

Simply fix the problem and click Check again to clear the error.

Once you're happy with your modifications, save the script by clicking **Save**. The script is first pre-compiled (as if you clicked Check) and if it's error-free, the editor closes and the script is saved. If an error is found, the script editor will stay open! If you want to close the editor without saving, click **Discard**.

For security reasons, only error-free scripts can be saved to the project. That's why scripts are pre-compiled when you click Save or when scripts are imported.

The Help button is your gateway to these help topics.

Further Reading

Introduction to Scripting

Script Function Reference

5.1.2. HEScript Function Reference

Welcome to the HEScript Function Reference. This comprehensive guide will empower you to harness the full potential of HEScript in controlling your ebook or application's behavior and interacting with the HTML Executable runtime module and Windows environment.

Introduction

HEScript scripts are tailored to enhance your application's capabilities. They provide a wide array of internal functions and macros that can be seamlessly integrated into your scripts.

Notes:

- Some internal functions, although not listed here, can be utilized in system HTML pages. Feel free to reach out to us for more information.
- Explore additional references to Pascal Object on DelphiBasics.

Interface Functions

MessageBox

The `MessageBox` function allows you to display user-friendly message boxes, akin to the Windows API MessageBox.

function MessageBox(const Text: String; const Title: String; const Flags: Integer): Integer;

Text: Content of the message box.
Title: Title of the message box.
Flags: A set of bit flags that dictate the content and behavior of the box. For detailed information, check here.
Flag examples:
MB_OK
MB_OKCANCEL
MB_YESNOCANCEL
MB_YESNO

Additionally, you can use:

MB_ICONSTOP
MB_ICONINFORMATION
MB_ICONWARNING
Result (integer): Returns IDOK, IDCANCEL, etc., based on the user's selection.

MessageDlg

Display a customizable message box with the `MessageDlg` function.

function MessageDlg(const Message: String; const Title: String; DialogType: TMsgDlgType; Buttons: TMsgDlgButtons): Integer;

Message: Content of the message box.
Title: Title of the message box.
DialogType: Enumerated values like mtWarning, mtError, mtInformation, mtConfirmation, or mtCustom.
Buttons: Flags defining the message box buttons, such as mbYes, mbNo, mbOK, mbCancel. Example: [mbYes, mbNo].

Example:

MessageDlg("My message", "My title", mtInformation, [mbOK]);

For more details, refer to DelphiBasics.

Note: Avoid calling this function during initialization or when the publication exits.

SetUIProp

`SetUIProp` sets the value of a specified property for a control.

procedure SetUIProp(const id, propname, propval: String);

id: Name of the control.
propname: Name of the property.
propval: New value (always a string).

GetUIProp

Retrieve the value of a specified property for a control with the `GetUIProp` function.

```
function GetUIProp(const id, propname: String): String;
```

id: Name of the control.
propname: Name of the property.
result: The property's value (in string format). If the control is not found, an error may occur.

ChangeStatusBar

Modify the status bar text with the `ChangeStatusBar` function.

```
procedure ChangeStatusBar(const Text: String; Reset: Boolean);
```

Text: New text to display in the status bar.
Reset: If true, the text will be set as the default.

ShowLeftPanel

Control the display of navigation panels using the `ShowLeftPanel` function.

```
procedure ShowLeftPanel(const id: Integer; OpenIt: Boolean);
```

Id: Index of the panel you wish to display.
OpenIt: If false, the navigation panel is hidden (regardless of the Id). If true, the panel is displayed.

Possible ID values:

1: Search
2: Table of Contents (TOC)
3: unused
4: Favorites

Please note that the navigation panel must be enabled in HTML Executable to ensure proper functionality.

GetLeftPanel

Retrieve the index of the currently displayed navigation panel using the `GetLeftPanel` function.

```
function GetLeftPanel: Integer;
```

Result: Index of the currently displayed panel. Returns 0 if no panel is displayed.

ShowAboutBox

Display the About box with the `ShowAboutBox` function.

```
procedure ShowAboutBox;
```

ManageWaitForm

Show and hide a "Please wait..." dialog box using the `ManageWaitForm` function.

procedure ManageWaitForm(Show: Boolean; Text: String);

> **Show:** True to show the dialog box, False to hide it.
> **Text:** Text to display on the dialog box.

After calling `ManageWaitForm(True, "...")`, don't forget to call `ManageWaitForm(False, "...")` to hide the dialog box.

Navigation Functions

GoToPage 🌐

Navigate to the specified URL or execute a hescript command with the `GoToPage` function.

procedure GoToPage(const Name, Window: String);

> **Name:** Partial or full URL, hescript command, or internal protocol (e.g., ghe://heserver/).
> **Window:** the name of the window for the navigation. Cabn be one of the special HTML Executable targets.

UseMapID 🗺

Display the HTML page corresponding to the given map ID using the `UseMapID` function.

procedure UseMapID(const ID: Integer);

> **ID:** Map ID of the HTML page. More information here.

ReloadPage 🔃

Refresh the entire page with the `ReloadPage` function.

procedure ReloadPage;

GetCurrentHTMLPagePath 📄

Retrieve the full URL of the currently displayed HTML page with the `GetCurrentHTMLPagePath` function.

function GetCurrentHTMLPagePath: String;

> **Result:** The full URL of the current page.

ExitPublication 📱

Terminate the publication with the `ExitPublication` function.

procedure ExitPublication;

NavigateCommand 🚀

Execute common navigation commands with the `NavigateCommand` function.

procedure NavigateCommand(const Command: Integer);

Command: One of the following values:
- 0: Return to the previous page (back).
- 1: Move to the next page (forward).
- 2: Unused
- 3: Copy selected text to clipboard.
- 4: Select the entire page.
- 5: Paste text from clipboard.
- 6: Page zoom in.
- 7: Page zoom out.
- 8: Unused
- 9: Cut selected text to clipboard.
- 10: Show developer tools (or hide them)

SeqGoPrevious

Navigate to the previous page in a Browse Sequence.

procedure SeqGoPrevious;

SeqGoNext

Navigate to the next page in a Browse Sequence.

procedure SeqGoNext;

ExecuteHTMLScript

Execute a script function in an HTML page (JavaScript).

procedure ExecuteHTMLScript(const CommandLine, Page: String);

CommandLine: The JavaScript code to run,
Page: Leave empty.

Example: To call this simple JavaScript function:

```
<script language="JavaScript">
<!--
function Say(what)
{
    window.alert(what)
}
-->
</script>
```

Use this:

ExecuteHTMLScript("Say('this is what I say')", "JavaScript");

RefreshTOC

Refresh the Table of Contents.

procedure RefreshTOC;

Especially useful when working with the Visibility HEScript function.

LoadTOCfromXMLFile

Load the Table of Contents from an XML file previously saved with HTML Executable.

procedure LoadTOCfromXMLFile(SourceFile: String);

> **SourceFile:** Full path to an existing XML file. If the XML file is stored inside the publication, use UnpackTemporaryResource first.

If SourceFile is empty, the publication loads the default Table of Contents.

LoadPDFFromFile

Load an external PDF file directly into the publication's built-in PDF viewer (requires the built-in PDF viewer option to be turned on).

procedure LoadPDFFromFile(PDFPath: String);

> **PDFPath:** Full path to the PDF file you want to open. The PDF file must exist.

PDFViewerCommand

Send commands to the built-in PDF viewer.

function PDFViewerCommand(Command: Integer; Param: Integer): Integer;

> **Command:** A unique identifier for the command to be sent. See the list of all available commands.
> **Param:** An integer parameter related to the command sent.
> **Result:** Result from the built-in PDF viewer.

PDFViewerCommandStr

Send commands to the built-in PDF viewer.

function PDFViewerCommandStr(Command: Integer; Param: Integer): Integer;

> **Command:** A unique identifier for the command to be sent.
> **Param:** An integer parameter related to the command sent.
> **Result:** Result from the built-in PDF viewer.

These functions provide advanced control and interaction with the publication's content, including the ability to execute scripts, navigate sequences, and work with external files.

Management Functions

GetGlobalVar

Get the value of a global variable.

function GetGlobalVar(const Name, DefaultIfNotFound: String): String;

> **Name:** Name of the global variable.
> **DefaultIfNotFound:** Value to return if the global variable is not found.
> **Result:** The value of the variable if it exists; otherwise, the value of DefaultIfNotFound.

Use global variables to store or exchange data between different scripts.

SynchronizeGlobalVar

If several instances of a publication are running, call this function to force all global variables (only the persistent ones) to get the same value as the ones of the publication that made the call.

procedure SynchronizeGlobalVar;

Persistent global variables can be used to share data between several instances of a publication.

GetString

Return the value of a resource string.

function GetString(const ID: String): String;

> **ID:** Name of the resource string.
> **Result:** The value of the resource string, or empty if not found.

GetManualHardwareID

Return a unique system ID based on hardware specifications.

function GetManualHardwareID(method: Integer): String;

> **method:** Method identifier to compute the unique ID.
> 0: Serial number of the first hard disk.
> 1: Manufacturer-allocated number used to identify the physical media.
> 2: CPU specifications such as CPU ID.
> 3: Manufacturer-allocated number of the first hard disk.

If an error occurs, an empty value is returned.

WriteRegStr

Write a string entry in the Windows registry.

procedure WriteRegStr(Root: Integer; Key, Ident, Value: String);

Root: Root key ID (see list below).
Key: Name of the subkey.
Ident: Name of the entry.
Value: String value to be written.

List of root key IDs:

0: HKEY_CLASSES_ROOT
1: HKEY_CURRENT_CONFIG
2: HKEY_CURRENT_USER
3: HKEY_DYN_DATA
4: HKEY_LOCAL_MACHINE
5: HKEY_USERS

WriteRegDW

Write an integer entry in the Windows registry.

procedure WriteRegDW(Root: Integer; Key, Ident: String; Value: Integer);

Root: Root key ID (see list above).
Key: Name of the subkey.
Ident: Name of the entry.
Value: Integer value to be written.

ReadRegStr

Read the value of a string entry from the Windows registry.

function ReadRegStr(Root: Integer; Key, Ident, Default: String): String;

Root: Root key ID (see list above).
Key: Name of the subkey.
Ident: Name of the entry.
Default: Default return value if the entry is not found.
Result: The value of the string entry.

ReadRegDW

Read the value of an integer entry from the Windows registry.

function ReadRegDW(Root: Integer; Key, Ident: String; Default: Integer): Integer;

Root: Root key ID (see list above).
Key: Name of the subkey.
Ident: Name of the entry.
Default: Default return value if the entry is not found.
Result: The value of the integer entry.

GetViewerSerial

Return the unique ID of the runtime module used to run this publication.

function GetViewerSerial: String;

HTML Functions

ShowHTMLBlockID

Provides an easy way to control the display of a block of HTML text. Works with HTML tags such as `<div>`, `<p>`, `<ul>`, `<form>`, `<table>`, and more. The block tag must contain an `ID=` attribute to identify it.

procedure ShowHTMLBlockID(const ID: String; Visible: Boolean);

> **ID:** ID of the HTML block.
> **Visible:** Setting Visible to False hides the block, and setting it to True displays the block.

This function works both in HTML Viewer and Internet Explorer (IE) browser publications.

SetFormControlValue

Sets the value of a form field.

procedure SetFormControlValue(const ControlName, PropertyName, PropertyValue: String);

> **ControlName:** Name of the control or form field. There is no need to indicate which form the field belongs to, but your controls should have unique names on an HTML page.
> **PropertyName:** Which property should be set.
> **PropertyValue:** The new value to give to the property.

GetFormControlValue

Gets the value of a form field.

function GetFormControlValue(const ControlName, PropertyName, DefValue: String): String;

> **ControlName:** Name of the control or form field. There is no need to indicate which form the field belongs to, but your controls should have unique names on an HTML page.
> **PropertyName:** The property whose value should be read.
> **DefValue:** The value if the control is not found.
> **Result:** Returns the value of the property in string format.

ShowFindDialog

Displays the Find Text dialog (search in the current HTML page only).

procedure ShowFindDialog(const FindText: String);

> **FindText:** Default text to search for. Parameter only used with HTML Viewer publications.

PrintPages

Prints the current HTML page.

procedure PrintPages;

Prints the current document. The standard Print Dialog will be shown.

StrToHTML

Generates the HTML code for the given string.

function StrToHTML(const Str: String): String;

Str: The string you want to convert to HTML.

Program / File / Folder Functions

OpenFile

Opens an external document or program file.

function OpenFile(const Filename, Parameters: String; State: Integer): Integer;

Filename: Full path to the file you want to open or run.
Parameters: Optional command line parameters.
State: Window state. May be SW_SHOWNORMAL, SW_SHOWMAXIMIZED, SW_SHOWMINIMIZED, SW_HIDE...
Result: Successful if greater than 31.

For executable files, you may also use RunAProgram.

OpenFileAdv

Opens an external document or program file (advanced: lets you specify the operation to execute).

function OpenFileAdv(const Filename, Parameters, Verb: String; State: Integer): Integer;

Filename: Full path to the file you want to open or run.
Parameters: Optional command line parameters.
Verb: Specifies the action to be performed: open, runas, print...
State: Window state. May be SW_SHOWNORMAL, SW_SHOWMAXIMIZED, SW_SHOWMINIMIZED, SW_HIDE...
Result: Successful if greater than 31.

For executable files, you may also use RunAProgram.

UnpackTemporaryResource

Extracts a file from the publication's data (it must have been compiled) to a temporary file and returns the path to this file. The file is normally removed when the publication closes.

function UnpackTemporaryResource(const SourceName: String): String;

SourceName: Virtual path to the file you want to extract from the publication's data.
Result: Full path to the extracted temporary file on the user's hard disk.

You can use this function to extract and run special files that cannot be handled by the publication, such as video, executable, etc. Useful with OpenFile.

ExportAFile

Same as the previous function, except that it will display a Save As dialog box, and the file is not temporary. Acts as when you download a file from the Internet.

```
function ExportAFile(const SourceName: String; PromptTitle, DefFilename: String): String;
```

SourceName: Virtual path to the file you want to extract from the publication's data.
PromptTitle: The title of the Save As dialog box.
DefFilename: The default filename that should be used. It can be ".".
Result: Full path to the final unpacked file.

The file is not removed! Useful if you want to let your end users "download" files from your publication.

GetTemporaryFilename

Returns the full path to a temporary file.

```
function GetTemporaryFilename: String;
```

This function may be useful for functions like SaveFormResultsToFile.

RunAProgram

Executes the specified executable program file.

```
function RunAProgram(const Filename, Params, WorkingDir: String; Wait: Boolean; DispWindow: Integer):
Boolean;
```

Filename: Full path to the .exe file you want to run.
Params: Optional command line parameters.
WorkingDir: The path to the folder that should be set as the system current one.
Wait: Indicates whether the publication "sleeps" until the end of the program's execution.
DispWindow: Window state. May be SW_SHOWNORMAL, SW_SHOWMAXIMIZED, SW_SHOWMINIMIZED, SW_HIDE...
Result: True if the program was run successfully.

It does not work with document files.

OpenFileDialog

Displays the standard File Open dialog box.

```
function OpenFileDialog(const aTitle, aFilename, aDefaultExt, aFilter, aInitialDir: String): String;
```

aTitle: Title of the dialog box.

aFilename: Default filename.
aDefaultExt: Default extension.
aFilter: File extension filter.
aInitialDir: Initial directory.

Result: Returns the full path to the selected file. If canceled, returns an empty string.

Warning: The Main Window must be available (do not use this function in an event like OnPubLoaded).

SaveFileDialog 🗁

Displays the standard File Save As dialog box.

```
function SaveFileDialog(const aTitle, aFilename, aDefaultExt, aFilter, aInitialDir: String): String;
```

aTitle: Title of the dialog box.
aFilename: Default filename.
aDefaultExt: Default extension.
aFilter: File extension filter.
aInitialDir: Initial directory.

Result: Returns the full path to the selected file. If canceled, returns an empty string.

Warning: The Main Window must be available (do not use this function in an event like OnPubLoaded).

SelectDirectory 🗁

Displays a browse folder dialog box.

```
function SelectDirectory(const Caption: String; const Root: String): String;
```

Caption: Indicative text asking for the folder.
Root: The root folder. Optional: leave it empty to use the Desktop as the root folder.

Result: Returns the full path to the selected folder. If canceled, returns an empty string.

UnpackVirtualResource 🖫

Extracts a file from the publication's data (it must have been compiled) to the virtual memory storage and returns the path to this file. The file only exists in memory: it is not on the hard disk. Nevertheless, it works as if it was on the hard disk at the specified path.

```
function UnpackVirtualResource(const SourceName: String; const DestFile: String): String;
```

For advanced users only.

SourceName: Virtual path to the file you want to extract from the publication's data.
DestFile: The path where the virtual resource should be unpacked to.

Miscellaneous Functions

ReplaceString

Returns a string with occurrences of one substring replaced by another substring.

function ReplaceString(const Original, OldPattern, NewPattern: string): String;

> **Original:** Original string with old patterns.
> **OldPattern:** String to replace.
> **NewPattern:** The new string that takes the place.
> **Result:** The new string with all replaced patterns.

Similar to StringReplace in Delphi SysUtils.

RegisterPub

Registers a Trial publication. Only working if you have a Trial publication. You should then call ActivateCertificate in order to apply changes.

procedure RegisterPub(name, data1, data2, key: String);

> **Name:** User name.
> **Data1, Data2:** Optional. Used to generate the key.
> **Key:** The key specified by the user.

This function will replace the current registration information items with the provided ones. For security reasons, it does not check the validity of the key: at the next startup, the publication will run with the certificate whose key was made for.

ActivateCertificate

Forces the trial publication to activate the certificate that matches the current registration items saved by RegisterPub.

function ActivateCertificate: String;

Should only be used after a call to RegisterPub, and followed by ExitPublication. Returns the name of the selected certificate. Empty if no certificate matches.

GetCurrentCertificate

Returns the name of the current certificate. Only working if you have a Trial publication.

function GetCurrentCertificate: String;

SetDefaultCertificate

Sets the current certificate to the default one.

procedure SetDefaultCertificate;

This function causes the publication to come back to the Default certificate. If the default certificate has an expiration period, this period is automatically expired.

MD5OfAString

Converts the string to UTF-8 if not ANSI and computes the MD5 hash sum.

function MD5OfAString(const Str: String): String;

> **Str:** String you want to get the MD5 sum of. Conversion from Unicode to UTF8 is done automatically.
> **Result:** String format of the MD5 sum.

MD5OfAStringUnicode

Computes the MD5 hash sum of a Unicode string.

function MD5OfAStringUnicode(const Str: UnicodeString): UnicodeString;

MD5OfAFile

Gets the MD5 hash sum of the specified file.

function MD5OfAFile(const Path: String): String;

> **Path:** Full path to the file you want to get the MD5 sum of.
> **Result:** String format of the MD5 sum.

InputBox

Prompts your end users for a query (a dialog box is shown).

function InputBox(const Query, Title, Default: String): String;

> **Query:** The prompt text. Tells your end users what you are asking.
> **Title:** Title of the dialog box.
> **Default:** The default value to display in the field.
> **Result:** Returns what end users answered or an empty value if they chose “Cancel”.

ParamStr

Returns the index-th parameter passed to the program using the command line.

function ParamStr(index: Integer): String;

> **Index:** Index of the parameter you want to obtain.

Note: Use double quotes to wrap multiple words as one parameter (such as long file names containing spaces).

StartTimer

Starts a timer to trigger an event, either one time or repeatedly, after a measured interval. Write the code that you want to occur at the specified time inside the UserMain's OnTimer function event.

procedure StartTimer(const TimerName: String; const Interval: Cardinal);

TimerName: Name of the custom timer. Cannot be blank.
Interval: Determines the amount of time, in milliseconds, that passes before the timer initiates another OnTimer event. For example, 1000 for one second. Note: A 0 value is valid, however, the timer won't call an OnTimer event for a value of 0.

StopTimer

Stops a timer that was created by StartTimer.

```
procedure StopTimer(const TimerName: String);
```

TimerName: Name of the custom timer. NOT CURRENTLY USED. Just leave empty.

StrUnicodeToUtf8

Converts a normal Unicode string to a UTF8 encoded string.

```
function StrUnicodeToUtf8(const Str: String): String;
```

Str: The string you want to convert.

StrUtf8ToUnicode

Converts a UTF8 encoded string to a normal Unicode string.

```
function StrUtf8ToUnicode(const Str: String): String;
```

Str: The string you want to convert.

SetClipboardText

Set the contents of the Windows Clipboard to the text from Data.

```
procedure SetClipboardText(const Data: String);
```

Data: The text data to put in the clipboard.

GetClipboardText

Retrieves the contents from the Windows Clipboard as text.

```
function GetClipboardText: String;
```

RandomRange

Generates a random Integer number within the range RangeFrom to RangeTo inclusively.

```
function RandomRange(const RangeFrom, RangeTo: Integer): Integer;
```

To delve further into HEScript and expand your scripting capabilities, you can explore the following resources:

Introduction to Scripting

Additionally, if you want to understand how to use the Script Manager effectively:

Using the Script Manager

These resources will provide you with more insights into HEScript and its practical applications.

5.1.3. Run and Call HEScript From JavaScript And HTML

Script Usage Scenarios

Explore the power of scripting in HTML Executable, and take your applications to the next level!

Scripts in HTML Executable offer a wide range of possibilities, including:

- Defining security profiles with Boolean conditions
- Creating HTML links that execute scripts
- Implementing JavaScript functions
- Responding to global application events

In this comprehensive guide, we'll walk you through each of these use cases, providing clear instructions and even working demonstrations for a deeper understanding.

Leveraging Scripts with Security Profiles

Security profiles allow you to define "Boolean function result" conditions that call specific functions within a script. When you add such a condition to a security profile, you can choose the Boolean function you wish to execute. If the function returns true, the condition is considered fulfilled, and its associated actions and restrictions are enforced.

Harnessing the Power of HTML Links

HTML Executable enables you to call HEScript procedures and functions independently via HTML links. To do so, simply use the `hescript://` prefix instead of `http://` to signal the runtime module to execute an HEScript procedure or function. The format as follows:

```
hescript:[scriptname].[functionprocedurename]
```

Here's an example:

```
<a href="hescript:MyScript.Procedure1">Click here to execute my first function</a>
```

You can even pass string parameters using a `|` separator. Just remember to properly encode non-ASCII characters of parameters. For instance, replace spaces in arguments by `%20`.

For URL encoding, you can use this online tool.

Another Demonstration

Let's take a look at another example. In this case, we'll use string parameters in our links. Here's the script procedure:

```
procedure MySecondDemo(cond: String);
var
S: String;
begin
S := "first";
if cond="2" then S := "second";
MessageBox("The " + S + " link was clicked", "Second Demo",
MB_OK+MB_ICONINFORMATION);
end;
```

You can create two links with different parameters:

```
<a href="hescript://demo1.MySecondDemo|1">This is the 1st link</a>
```

- This is the 1st link
- This is the 2nd link

Utilizing JavaScript Integration

Finally, you can seamlessly integrate HEScript with JavaScript. The global `htmlexe` JavaScript object allows communication between JavaScript scripts and HEScript: use these two JavaScript methods: `htmlexe.RunHEScriptCom` or `htmlexe.GetHEScriptCom` to communicate with HEScript scripts

WARNING: JavaScript implementation works asynchronously when invoking HEScript functions..

Calling HEScript procedures with JavaScript

ⓘ **When you don't expect a result from an HEScript procedure,** you can use `htmlexe.RunHEScriptCom`.

```
htmlexe.runHEScriptCom('[scriptname].[functionprocedurename]|param1|param2|....|paramN');
```

This works exactly as the hescript:// method for HTML links explained above; you have just to replace it by htmlexe.RunHEScriptCom(".....");

Examples:

```
htmlexe.RunHEScriptCom("hescript://UserMain.Procedure1");
```

```
<script>
$(function() { $("#btnjs1").click( function() {
htmlexe.RunHEScriptCom('hescript://UserMain.Procedure1'); } ); });
</script>
<button id="btnjs1" class="btn btn-primary">Test Procedure 1</button>
```

Calling HEScript string functions with JavaScript

ⓘ **When you expect the result from an HEScript string function**, you must use `htmlexe.GetHEScriptCom` to execute and read the result with JavaScript. This function does not return the value itself: you must pass a callback JavaScript function that will receive the result. The callback should be a simple JavaScript function with a single parameter that will receive the HEScript function's result.

```
htmlexe.GetHEScriptCom('[scriptname].[functionprocedurename]|param1|param2|....|paramN',
callbackJS);
```

Example:
```
// Callback function
function DemoCallback2(content) { $('#result').text(content); }

// Main code:
htmlexe.GetHEScriptCom('hescript://UserMain.ReturnDate', DemoCallback2);
```

Once the UserMain.ReturnDate function returns the result, the JavaScript callback named DemoCallback2 is fired.

Example:
```
<script>  // Callback function
 function DemoCallback2(content) { $('#result').text(content); }

 $(function() { $("#btnjs2").click( function() {
htmlexe.GetHEScriptCom('hescript://UserMain.ReturnDate', DemoCallback2); } ); });
</script>
<button id="btnjs2" class="btn btn-primary">Tell me the date</button>
```

Other examples

Open the HTML Executable's Main Demonstration to see working examples

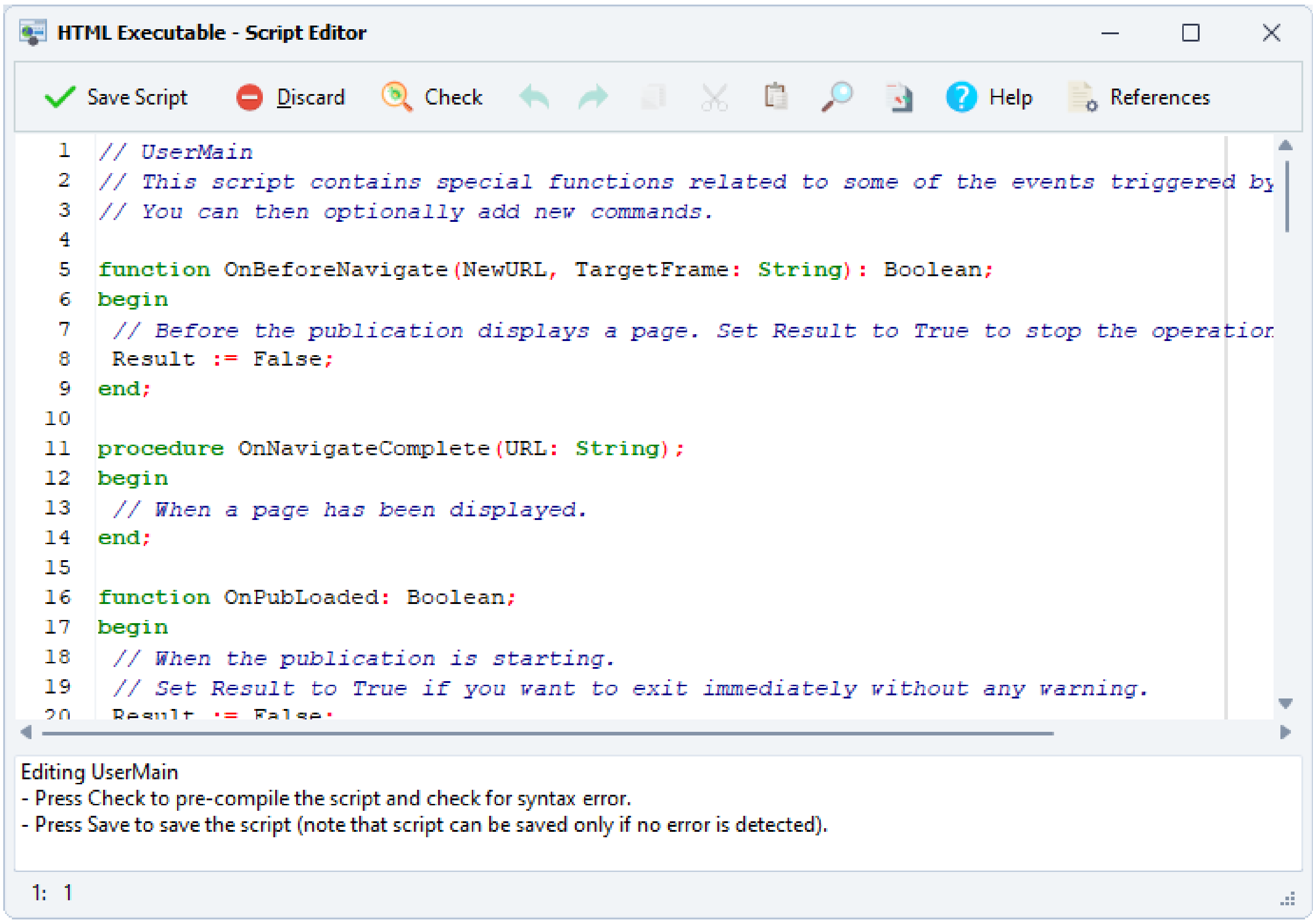
Introduction to Scripting

5.1.4. UserMain Script And Script Templates

In this topic we'll explore pre-defined HEScript scripts, including the UserMain script, and delve into their global application events.

About the UserMain Script

The UserMain script is an integral part of every project initiated with HTML Executable. It plays a pivotal role by housing numerous global events that enable you to inject special commands and tailor your application's behavior.



```
// UserMain
// This script contains special functions related to some of the events triggered by
// You can then optionally add new commands.

function OnBeforeNavigate(NewURL, TargetFrame: String): Boolean;
begin
 // Before the publication displays a page. Set Result to True to stop the operation
 Result := False;
end;

procedure OnNavigateComplete(URL: String);
begin
 // When a page has been displayed.
end;

function OnPubLoaded: Boolean;
begin
 // When the publication is starting.
 // Set Result to True if you want to exit immediately without any warning.
 Result := False;
```

```
Editing UserMain
- Press Check to pre-compile the script and check for syntax error.
- Press Save to save the script (note that script can be saved only if no error is detected).
```

1: 1

Here are some key events provided by the UserMain script:

OnBeforeNavigate (NewURL, TargetFrame: String): Boolean

This event triggers just before the application displays an HTML page. You can set the result to True to halt the operation. NewURL represents the full URL to be displayed, and TargetFrame denotes the frame where the page appears, if any.

OnNavigateComplete (URL: String)

It fires when a page has finished loading. URL represents the full URL of the loaded page.

OnPubLoaded: Boolean

This event occurs when the application starts, just before displaying the homepage. Set Result to True if you wish to exit immediately without any warning.

OnPubBeingClosed

Triggered when the application is about to terminate.

OnStartSearching (what: String);

When a full search is initiated. "what" contains the exact user query.

OnDisplayWindow (WindowName: String)

This event comes into play when a window, specified by WindowName, is displayed. It applies to both main and secondary windows, including pop-ups.

OnStartMainWindow

Occurs just before the main window is displayed, right before showing the homepage.

OnMinimizeToTray

When the main window is minimized to tray.

OnRestoreFromTray

When the main window is restored from tray.

OnCloseWindow (WindowName: String)

This event triggers when a user closes a window.

OnWindowCloseQuery (WindowName: String): Boolean

If a user attempts to close a window, this event is called. You can set Result to False to prevent the window from closing, or True to allow it. Note that this event is not created by default and must be added manually to the UserMain script.

OnTimer (TimerName: String): Boolean

This event is triggered by a timer created using the StartTimer HEScript function. It activates when a specified amount of time, determined by the StartTimer function, elapses. Set the result of this function to True if you want to disable the timer or use StopTimer to achieve the same result.

OnInvalidPasswordAtStartup: Boolean

This event occurs when an invalid global password is provided at startup. Set to True if you prefer the application not to display an error message and exit.

OnExpiredPublication: Boolean

Triggers when the global expiration date is reached. Set to True if you wish to bypass error messages and exit.

OnKeyNotValidated: Boolean

Occurs when the validation of a registration key failed. Set to True if you want the application to exit.

OnKeyValidation: Boolean

Lets you decide whether validation of the registration key should be performed at startup or not. Set Result to True if you want the publication to be validated, False otherwise.

OnPDFDisplay(PDFPath: String)

This event is called when the PDF viewer loads and displays the PDF whose path is given by PDFPath. Only for the built-in PDF viewer.

OnWordViewComplete(WordURL: String);

This event is called when the Word Viewer loads and displays the DOCX whose URL is given by WordURL. Only for the built-in DOCX Viewer.

OnTrayIconClick / OnTrayIconDblClick

This event is called when the user clicks / double-clicks the tray icon.

OnWebUpdateNewVersionFound: Boolean

This event occurs when a new version is detected by the web update feature at publication startup. Set Result to True if you do not want the Web Update dialog box to be displayed and perform actions yourself

OnPrintPage

Fires when the user prints the current page.

OnFullScreenModeChange(FullScreen: Boolean);

Signals a change in full-screen mode, where FullScreen indicates whether full-screen mode is enabled (true) or disabled (false).
Use this to show or hide your custom controls thanks to the SetUIProp HEScript function.

function OnPermissionRequested(URI: string; PermissionKind: Integer; UserInitiated: Boolean; StateDefault: Integer): Integer;

It is triggered when a resource or action requires user permissions (e.g., accessing the microphone or camera).
The event has four parameters:

URI: A string representing the URL or URI associated with the permission request.
PermissionKind: An integer indicating the type of permission being requested. Possible values are:
`0`: Unknown permission.
`1`: Microphone access.
`2`: Camera access.
`3`: Geolocation access.
`4`: Notifications.
`5`: Other sensors access.
`6`: Clipboard read access.
UserInitiated: A Boolean indicating whether the request was initiated by the user (`true`) or programmatically (`false`).
StateDefault: An integer representing the default permission state to apply if no custom logic is provided. Possible values are:
`0`: Use the default state.
`1`: Allow the permission.

`2`: Deny the permission.

The event must return an integer specifying the chosen permission state. By default, the function returns the value passed as **StateDefault**, but this can be customized by you.

Warning: This event is only called when using the **WebView2 engine.**

A sample showing the user of this event (activating the webcam without permanent permission) is available

Explore more about scripting in HTML Executable:

Introduction to Scripting

Using the Script Manager

Script Function Reference

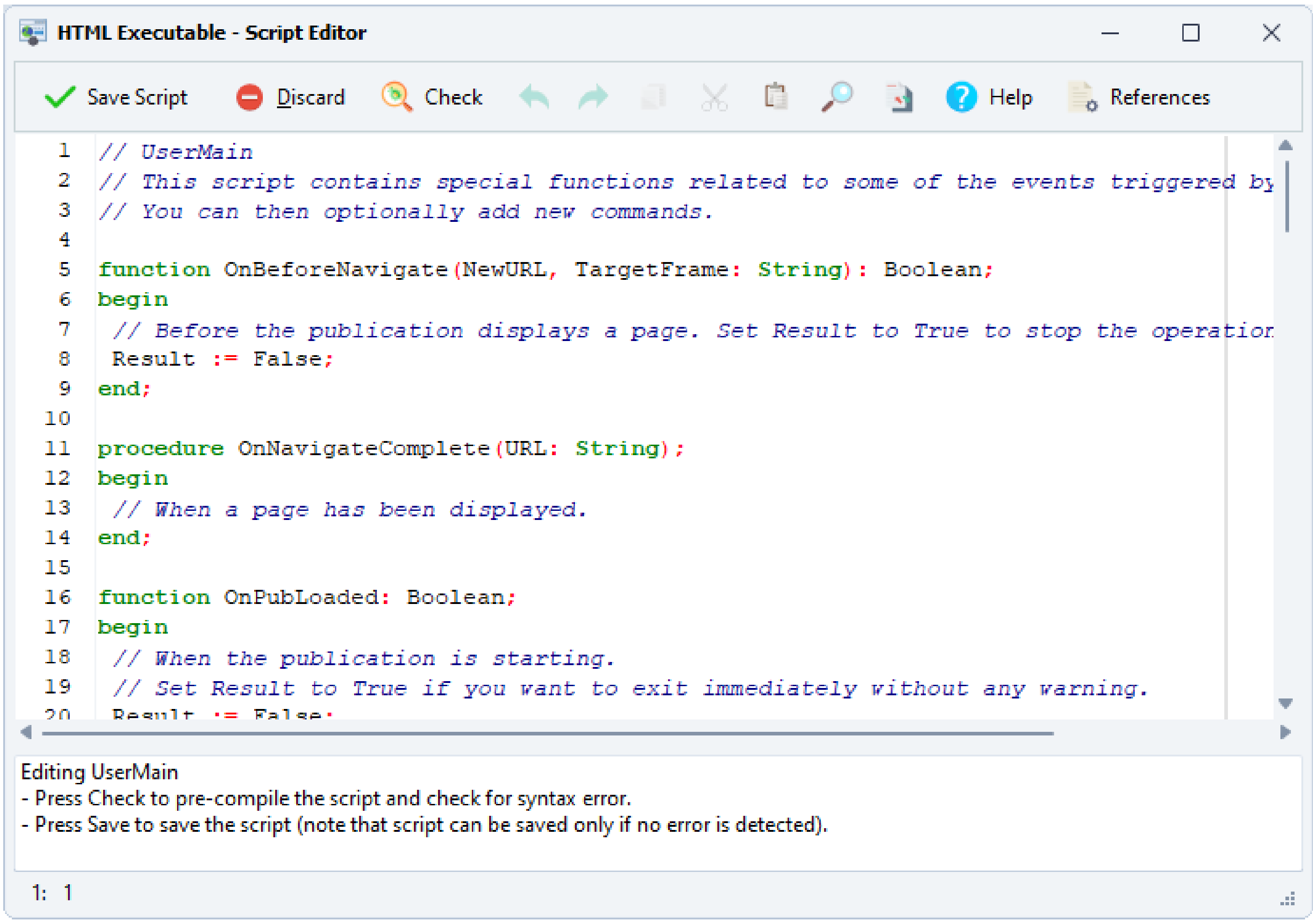
5.1.5. Quickly Add HEScript Code

This tutorial will walk you through the process of extending the functionality of your ebook application using HEScript. Whether you're a seasoned developer or just starting out, this guide will make the process of adding HEScript functions or procedures to your application a breeze.

Adding HEScript Code to Your Application

Suppose you've found some useful HEScript code on our forum that you'd like to incorporate into your application. Here's how you can do it:

Open HTML Executable and navigate to **Application Settings => Scripting**.
Double-click on `UserMain` in the list to open the script editor.



```
// UserMain
// This script contains special functions related to some of the events triggered by
// You can then optionally add new commands.

function OnBeforeNavigate(NewURL, TargetFrame: String): Boolean;
begin
 // Before the publication displays a page. Set Result to True to stop the operation
 Result := False;
end;

procedure OnNavigateComplete(URL: String);
begin
 // When a page has been displayed.
end;

function OnPubLoaded: Boolean;
begin
 // When the publication is starting.
 // Set Result to True if you want to exit immediately without any warning.
 Result := False;
```

Editing UserMain
- Press Check to pre-compile the script and check for syntax error.
- Press Save to save the script (note that script can be saved only if no error is detected).

1: 1

☞ For a comprehensive overview of the script editor and its features, check out our Script Editor Guide.

Paste the entire function you want to use into the editor. For example, the following HEScript code returns the path to the "My Documents" directory:

```
function GetDocPath: String;
begin
Result := GetSpecialFolderPath(16);
end;
```

Click **Save Script**. Voila! Your script is ready to use!

Executing Your HEScript Code 🚀

You can use JavaScript or HTML links to invoke your HEScript functions or procedures.

☞ Visit our JavaScript-HEScript Guide for examples and syntax details.

For more information, explore the following topics:

- Extend Functionality with HEScript
  - HEScript Script Editor
  - HEScript Function Reference
  - Run and Call HEScript From JavaScript And HTML
  - UserMain Script And Script Templates
  - Sample Scripts

- How to prompt end users for their name once and store it?
    - How to password protect pages?
    - How to call a javascript function from HEScript (toolbar, menubar)?
    - How to call DLL functions?
    - How to open an external file
    - How to open subfolders of a given folder or disk?
    - How to prompt for a password for closing ebook?
    - How to run an executable file or app?
    - How to dynamically modify the Table of Contents?
    - samplescript3
    - How to open and save files
- HTML Executable JavaScript API
- Special Targets for External Links
- Application Global Variables
- Publication Command Line Arguments
- Command Line - Directives
- customhtmldlg
- technicalnotes
- About Cookies

5.1.6. Sample Scripts

This page only contains some sample scripts. Please go to the forum for more samples, articles.

Please select a sample below to show it:

- How to prompt end users for their name once and store it?
- How to password protect pages?
- How to call a javascript function from HEScript (toolbar, menubar)?
- How to call DLL functions?
- How to open an external file
- How to open subfolders of a given folder or disk?
- How to prompt for a password for closing ebook?
- How to run an executable file or app?
- How to dynamically modify the Table of Contents?
- samplescript3
- How to open and save files

For more information, explore the following topics:

- Extend Functionality with HEScript
  - HEScript Script Editor
  - HEScript Function Reference
  - Run and Call HEScript From JavaScript And HTML
  - UserMain Script And Script Templates
  - Quickly Add HEScript Code
  - Sample Scripts
    - How to prompt end users for their name once and store it?
    - How to password protect pages?
    - How to call a javascript function from HEScript (toolbar, menubar)?

5.1.6.1. How to prompt end users for their name once and store it?

This guide will walk you through a script that accomplishes the following objectives:

- Prompts the end user to enter their name the first time the publication runs.
- Stores the user's name into a persistent global variable.
- Displays the user's name in an HTML page.

Getting Started

We will be using a global variable named "TheUserName".

Step 1: Writing the Script

Navigate to the Script Manager, double click on "UserMain" and copy/paste this script in the OnPubLoaded function event:

```
function OnPubLoaded: Boolean;
var S: String;
begin
// Checks whether the user was already prompted.
// If the TheUserName global variable already has a value, then we do not
// prompt the user again.
if GetGlobalVar("TheUserName", "") <> "" then exit;
// Prompts the user then:
S := InputBox("Welcome!"#13#10"Please enter your name:", "What is your name", "");
// If the user does not give a name, set it to "Mysterious user"...
if S = "" then
S := "Mysterious user"
else
begin
// Stores the result only if the user has given a name.
// So he/she will still be prompted the next time.
SetGlobalVar("TheUserName", S, True);
```

```
// True means that the global variable is stored.
end;
// When the publication is starting and before the homepage is displayed.
// Set Result to True if you want to exit immediately without any warning.
Result := False;
end;
```

Step 2: Displaying the Name in an HTML Page

Use this JavaScript code:

```
<script>
function DemoCallback(content) {
    document.write(content);
}
</script>
<script>
htmlexe.GetGlobalVariable('TheUserName', '', DemoCallback);
</script>
```

Additional Notes

You can customize this example as per your needs. For instance, instead of using a dialog box, you could display an HTML page at startup.

5.1.6.2. How to password protect pages?

This guide on how to utilize security profiles and scripting in HTML Executable. This guide will walk you through a practical example, demonstrating how to prompt users for a password to access certain pages, cache the password, and use a security profile to protect HTML pages with this password. Let's dive in!

The Objective

Our script aims to:

- Prompt the user to provide a password to view certain pages.
- Cache the password so it is only prompted once.
- Use a security profile to mark which HTML pages should be protected using this password.

The outcome? If a user attempts to view our password-protected pages, they will be prompted for the password. If the correct password is provided, access to the pages is granted. If not, an error page is displayed.

Step-by-Step Guide

Step 1: Writing the Script

We'll start by writing a Boolean function that can be used as a condition for our security profile. Head over to the Script Manager, create a blank script named "MySecurity1", and copy/paste the following code:

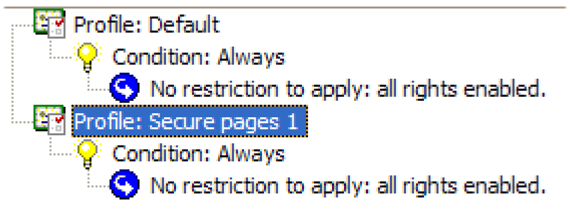
```
// MySecurity1
// A script to password protect pages.

// Our secret password:
const
 OurPassword="reality";

function CheckSecurity1: Boolean;
var
 S: String;
begin
{ This function is called by the security profile.
If it returns True, we configure the actions so that
the user may NOT view the pages.
If it returns False, then the secure pages may be displayed. }
// 1) Check if the password was cached.
   S := GetGlobalVar("CachePassword1", "");
   if S="" then
   begin
//  The password was never asked.
// So we ask it.
S := InputBox("Password protected page"#13#10 +
"This page is password protected; please enter the password in order to continue",
 "Security", "");
   end;
// 2) Check whether the password is correct.
Result := S <> OurPassword;
// Cache the password if valid.
if not Result then SetGlobalVar("CachePassword1", S, True);
// Result = true => pages locked.
// Result = false => pages unlocked.
end;
```

This code provides us with a Boolean function that we can use in a security profile. Don't forget to save the script!

Step 2: Creating a Security Profile ◐

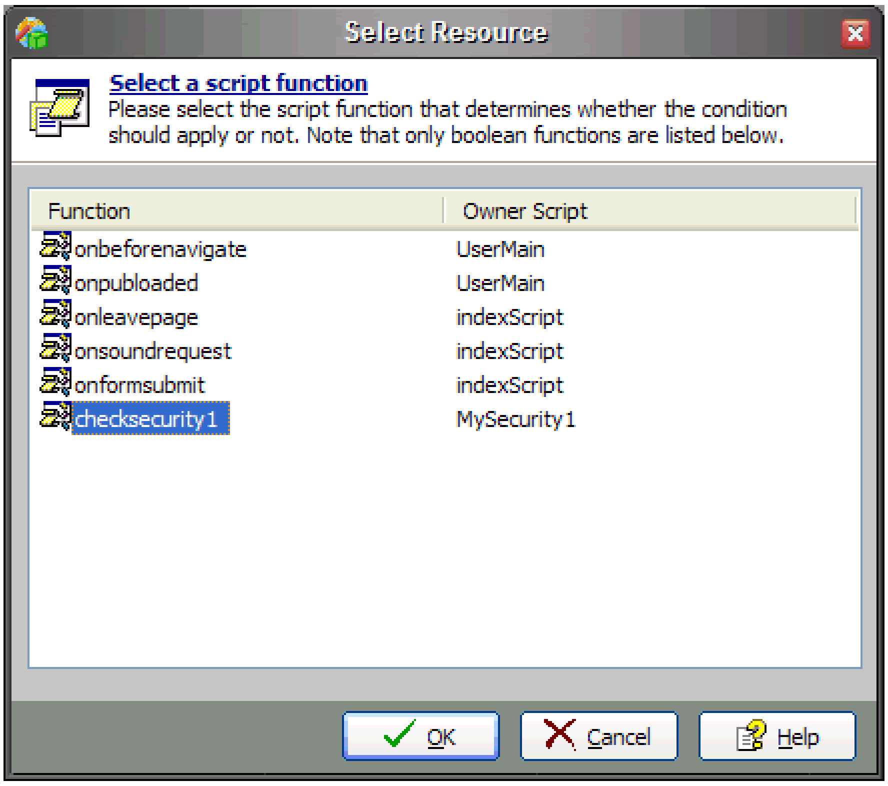
Next, we'll create a security profile:

- Go to the Security ribbon and select "Security Profiles".
- Click "Add" and select "Add New Security Profile".
- Enter "Secure pages 1" as the profile's name.



```
Profile: Default
    Condition: Always
        No restriction to apply: all rights enabled.
Profile: Secure pages 1
    Condition: Always
        No restriction to apply: all rights enabled.
```

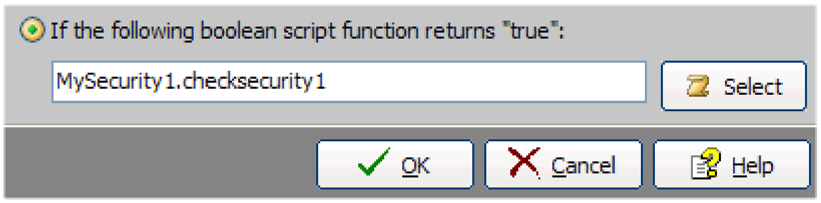
- Click "Add" and select "Add new condition".
- Click on "If the following boolean script function returns true" and select "checksecurity1":


Select Resource

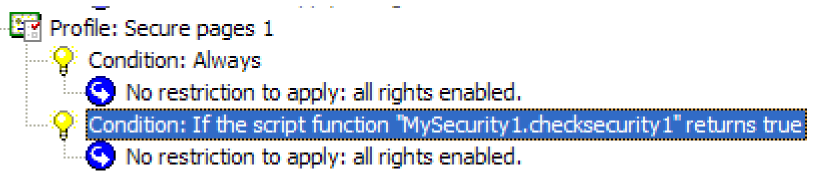
Select a script function

Please select the script function that determines whether the condition should apply or not. Note that only boolean functions are listed below.

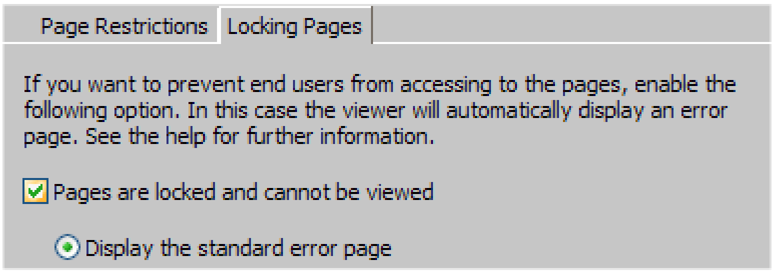
Function	Owner Script
onbeforenavigate	UserMain
onpubloaded	UserMain
onleavepage	indexScript
onsoundrequest	indexScript
onformsubmit	indexScript
checksecurity1	MySecurity1


If the following boolean script function returns "true":

MySecurity1.checksecurity1


```
Profile: Secure pages 1
    Condition: Always
        No restriction to apply: all rights enabled.
    Condition: If the script function "MySecurity1.checksecurity1" returns true
        No restriction to apply: all rights enabled.
```

➢ Finally, click "Configure" (select the condition in order to enable the Configure button). In the "Actions to execute" window, select the "Locking Pages" tab. Then enable "Pages are locked and cannot be viewed":

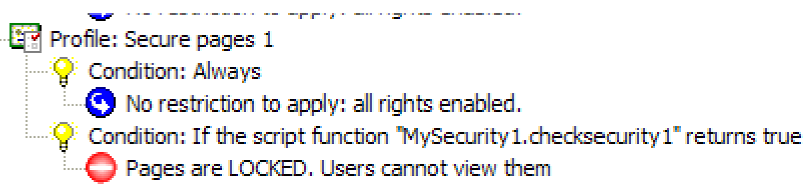


If you want to prevent end users from accessing to the pages, enable the following option. In this case the viewer will automatically display an error page. See the help for further information.

[✓] Pages are locked and cannot be viewed

[●] Display the standard error page

Your security profile is now ready!


```
Profile: Secure pages 1
    Condition: Always
        No restriction to apply: all rights enabled.
    Condition: If the script function "MySecurity1.checksecurity1" returns true
        Pages are LOCKED. Users cannot view them
```

Step 3: Protecting Your Pages

Now, let's select the pages we want to protect:

- Go to the File Manager and select the page(s) you want to protect.
- Click on "Properties" and in the File Properties window, select "Security".
- In the "Selected Security Profile" list, look for the security profile we just created, i.e. "Secure Pages 1".

Final Step: Test Your Publication

You can now compile your publication and test it.

Notes

Feel free to customize this example:
- Instead of using a prompt dialog box, you could create a login system.
- You can use anything else than a password to protect your files. Just modify the Boolean function.

5.1.6.3. How to call a javascript function from HEScript (toolbar, menubar)?

This guide explains how to call JavaScript functions from a button, menu item, or any general HEScript code in HTML Executable. Let's get started!

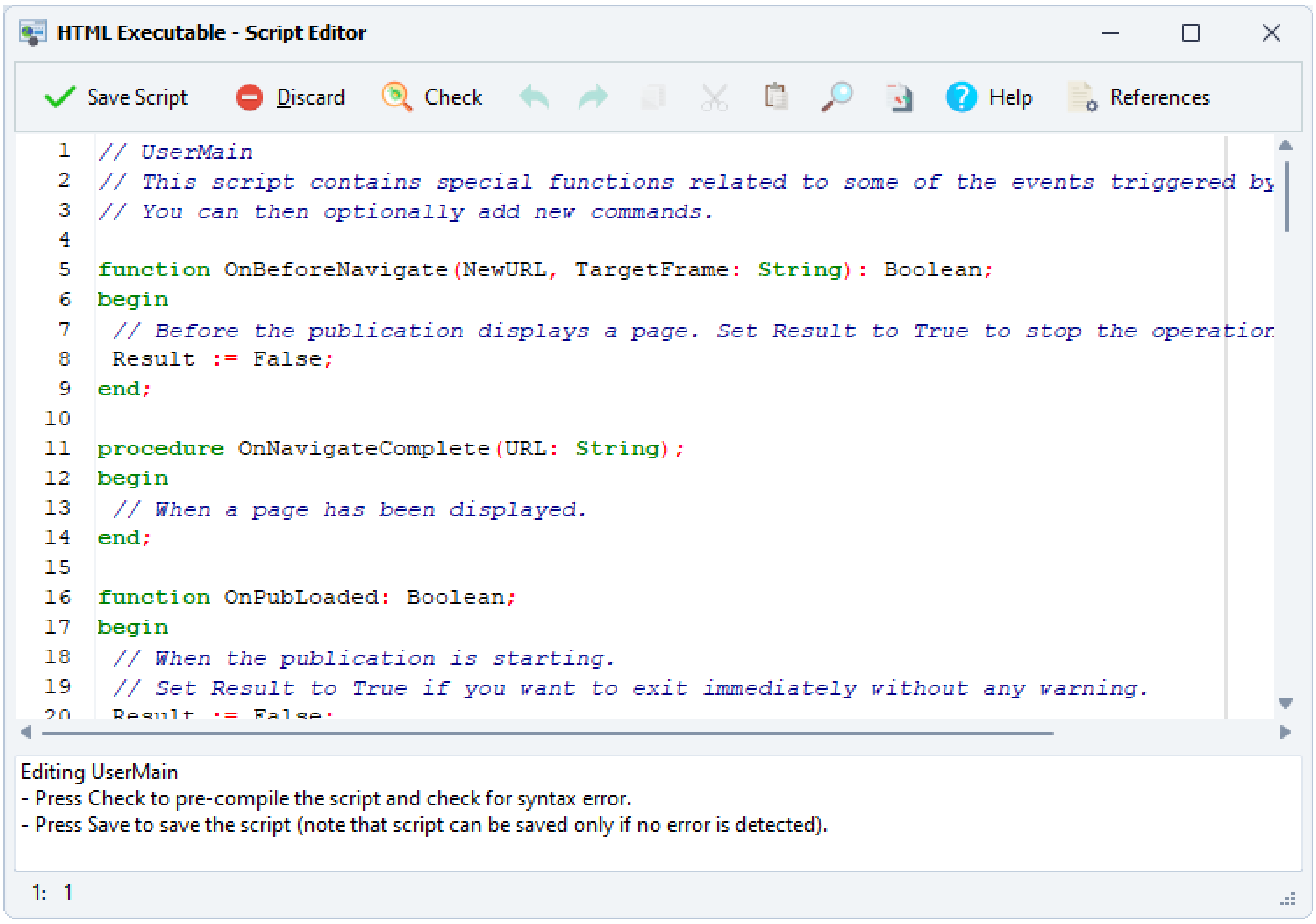
The Objective

You want to call a JavaScript function from your button, menu item, or any general HEScript code. How can you do that? The answer is simple: use the built-in HEScript function named `ExecuteHTMLScript`.

Step-by-Step Guide

Step 1: Open Your HTML Executable Project

Open your HTML Executable project and navigate to the Script Manager. Double-click the "UserMain" script to open the script editor:



```
// UserMain
// This script contains special functions related to some of the events triggered by
// You can then optionally add new commands.

function OnBeforeNavigate(NewURL, TargetFrame: String): Boolean;
begin
 // Before the publication displays a page. Set Result to True to stop the operation
 Result := False;
end;

procedure OnNavigateComplete(URL: String);
begin
 // When a page has been displayed.
end;

function OnPubLoaded: Boolean;
begin
 // When the publication is starting.
 // Set Result to True if you want to exit immediately without any warning.
 Result := False;
```

Editing UserMain
- Press Check to pre-compile the script and check for syntax error.
- Press Save to save the script (note that script can be saved only if no error is detected).

1: 1

Step 2: Use the ExecuteHTMLScript Function 🖉

In the script editor, use the following code if you want to call your JavaScript code in the default homepage at startup:

```
procedure OnStartMainWindow;
begin
  // When the main window is going to be displayed (just before the homepage is shown).
  ExecuteHTMLScript("sayHello('world')","");
end;
```

Don't forget to save your script!

Step 3: Include the Correct JavaScript Function 🗐

Ensure that you have the correct JavaScript function in the default homepage (or any HTML page). For instance:

```
<script>
function sayHello(name) {
  alert('Hello ' + name + '!');
}
</script>
```

Final Step: Compile Your Project 🧪

Compile your project and that's it! You've successfully called a JavaScript function from your HTML Executable project.

5.1.6.4. How to call DLL functions?

Extending Your Publication with DLL Functions

This guide will walk you through the process to extend your HTML Executable application by importing procedures and functions from external DLL files.

The Objective 🎯

The script engine in HTML Executable allows you to import procedures and functions from external DLL files, enabling you to extend your publication by calling your own DLL functions.

How to Import a DLL Function in HEScript? 📝

Importing a DLL function in HEScript is accomplished using the `external` keyword.

Syntax 🖉

The syntax for importing a DLL function is as follows:

```
function functionName(arguments): resultType; [callingConvention]; external
"libName.dll" [name 'ExternalFunctionName'];
```

For instance, the following declaration:

```
function MyFunction(arg: integer): integer; external 'CustomLib.dll';
```

imports a function called `MyFunction` from `CustomLib.dll`. The default calling convention, if not specified, is `register`.

HEScript allows you to declare a different calling convention (`stdcall`, `register`, `pascal`, `cdecl` or `safecall`) and to use a different name for the DLL function, like the following declaration:

```
function MsgBox(hWnd: Pointer; lpText, lpCaption: String; uType: Cardinal):
Integer; stdcall; external "user32.dll" name 'MessageBoxW';
```

This imports the `MessageBoxW` function from `User32.dll` (Windows API library), named `MsgBox` to be used in the script, like:

```
procedure TestDLL;
var
  S: String;
begin
  S := InputBox("What do you want to show?", "Query", "");
  MsgBox(0, S, "You entered:", MB_OK+MB_ICONINFORMATION);
end;
```

⚠ Please note that pointer and out parameters are not handled by the script engine.

5.1.6.5. How to open an external file

Welcome to a comprehensive guide . This guide on how to open document files located outside your HTML Executable ebook will walk you through the process in a clear and concise manner. Let's get started!

The Objective 🎯

You can open document files located outside your ebook, but you need to make some changes to your HTML code. The best approach is to use an HEScript script function. This function would be invoked from your HTML code and will open any file you specify with a parameter.

> 👉 Please note that the file to be opened must be in the same folder as the output EXE file or in a subfolder.

Step-by-Step Guide 📝

Step 1: Create the HEScript Function 🖉

Open your HTML Executable project and navigate to the Script Manager. Double-click the "UserMain" script to open the script editor. Add the following function code:

```
procedure OpenAFile(Filepath: String);
var
 EbookPath, MyFile: String;
 res: integer;
begin
 EbookPath := GetGlobalVar("HEPublicationPath", "");
 MyFile := IncludeTrailingPathDelimiter(EbookPath) + FilePath;
 res := OpenFile(MyFile, "", SW_SHOWNORMAL);
 if res < 32 then
 begin
  MessageBox("Error while opening: " + MyFile + ". Error code: " + inttostr(res),
"Launch Error", MB_OK+MB_ICONERROR);
  exit;
 end;
end;
```

Don't forget to save your script!

Step 2: Edit the HTML Page 🗐

Edit the HTML page with the links that should open the file. Replace the links with the following HTML code:

```
<a href="hescript://UserMain.OpenAFile|PATH TO YOUR DOCUMENT">Open my document</a>
```

Replace `PATH TO YOUR DOCUMENT` with the relative path to the file you want to open (do not remove the `|` character).

Please note that non-ASCII characters of the path for the URL should be encoded. URL encoding replaces unsafe ASCII characters with a `%` followed by two hexadecimal digits. For instance, spaces in filenames should be replaced by `%20`. You can find an online application for character conversion at: W3Schools

Examples

- If you want a link that opens `mydocument.txt`, use this HTML code:

```
<a href="hescript://UserMain.OpenAFile|mydocument.txt">Open my document</a>
```

- If your file is inside a subfolder, use a path relative to the folder that contains the publication .EXE file. For instance, to launch the file `my folder\my document.mp3`:

```
<a href="hescript://UserMain.OpenAFile|my%20folder\my%20document.mp3">Open the MP3</a>
```

Compile your project and that's it!

5.1.6.6. How to open subfolders of a given folder or disk?

This tutorial, which explains how to access subfolders within a compiled website on a CD using HTML Executable, will guide you through the steps in a straightforward and concise fashion. Let's begin!

The Objective

You want to create a compiled website on a CD, with the .exe file in the root of the CD. This website needs to open subfolders of the CD in Windows Explorer. This can be achieved by making some changes to your HTML code. The best approach is to use an HEScript script function. This function would be invoked from your HTML code and will open the subfolder you specify with a parameter.

Step-by-Step Guide

Step 1: Create the HEScript Function

Open your HTML Executable project and navigate to the Script Manager. Double-click the "UserMain" script to open the script editor. Add the following function code:

```
procedure OpenSubFolder(FolderName: String);
var
 EbookPath, MyFolder: String;
begin
 EbookPath := GetGlobalVar("HEPublicationPath", "");
 MyFolder := EbookPath + FolderName + "  ";
 OpenFile(MyFolder, "", SW_SHOWNORMAL);
end;
```

Don't forget to save your script!

Step 2: Edit the HTML Page

Edit the HTML page with the links that should open the subfolder. Replace the links with the following HTML code:

```
<a href="hescript://UserMain.OpenSubFolder|FOLDER NAME">Open Folder</a>
```

Replace `FOLDER NAME` with the name of the folder after the `|` character. Please note that folders shouldn't have spaces (or use the `%20` character like for URLs).

For instance, if you have this CD structure:

ROOT
Mycompiledwebsite.exe (the compiled .exe file)
-- Folder1 (a subfolder)
-- Folder2 (another subfolder)

If you want a link that opens "Folder1", use this HTML code:

```
<a href="hescript://UserMain.OpenSubFolder|Folder1">Open Folder 1</a>
```

Compile your project and that's it!

5.1.6.7. How to prompt for a password for closing ebook?

This is a tutorial on how to request a password to enable users to exit your HTML Executable ebook. This tutorial will guide you through the steps in a clear and concise manner. Let's begin!

The Objective 🎯

You want to prompt for a password to allow users to close your ebook. This can be achieved by making some changes to your HEScript script.

Step-by-Step Guide 📝

Step 1: Open Your HTML Executable Project 🗁

Open your HTML Executable project and navigate to the Script Manager. Double-click the "UserMain" script to open the script editor.

Step 2: Replace the Existing OnWindowCloseQuery Block Code 🖉

In the script editor, replace the existing `OnWindowCloseQuery` block code with the following code:

```
function OnWindowCloseQuery(WindowName: String): Boolean;
var
 S: String;
begin
 // Occurs when the user wants to close the window.
 // Note: set to False if you do not want to close the window.
 S := InputBox("To exit, please enter the password", "Security", "");
 Result := S = "OurPassword"; // replace OurPassword by what you want. Do not
remove quotes.
end;
```

Don't forget to save your script!

Compile your project and that's it!

👉 Please note that the ebook will be automatically closed without password if the user shuts down their computer.

5.1.6.8. How to run an executable file or app?

This comprehensive tutorial on How to Execute an External Program or a Program Embedded within your HTML Executable Publication will walk you through these processes in a clear and concise manner. Let's get started!

Running an External Program File 💻

We use the internal HEScript function named `RunAProgram` to execute a program. We only need to know the path to the program file we want to launch. Fortunately, HTML Executable has a global variable named `HEPublicationPath` pointing to the path of the folder that contains the publication's exe file.

Here's an example script:

```
procedure RunTutor;
var
 EbookPath, MyProgram: String;
begin
 EbookPath := GetGlobalVar("HEPublicationPath", "");
 MyProgram := EbookPath + "hehvdemo.exe";
 RunAProgram(MyProgram, "", EbookPath, false, SW_SHOWNORMAL);
end;
```

First, we get the path to our publication stored into the `EbookPath` variable. Then we add the filename of our executable program and we pass the result to the `RunAProgram` function.

Running a Program that was Compiled in the Publication 📑

You can include the program file directly into your publication .exe. In that case, you need to extract the program file before running it. Use the `_heopenit` target or the following code:

```
procedure RunACompiledProgram;
var
 MyProgram: String;
begin
 MyProgram := UnpackTemporaryResource("programfilename.exe");
 RunAProgram(MyProgram, "", ExtractFilePath(MyProgram), false, SW_SHOWNORMAL);
end;
```

This code uses the `UnpackTemporaryResource` internal function that extracts a compiled file from the publication to a temporary location. Then, you just need to execute the file.

5.1.6.9. How to dynamically modify the Table of Contents?

HTML Executable: Modifying the Table of Contents at Runtime 📖

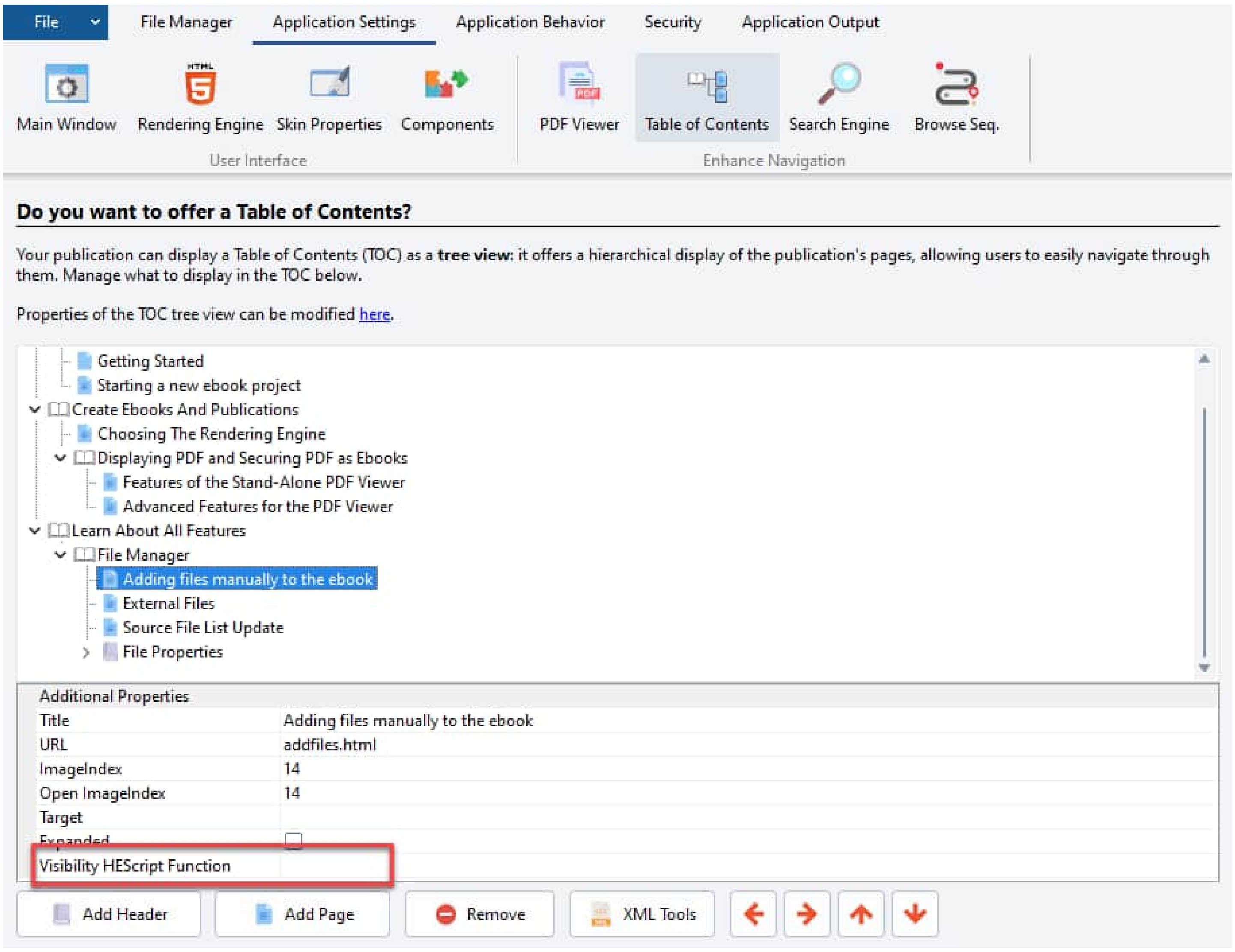
Welcome to a comprehensive guide on how to modify the Table of Contents (TOC) in your HTML Executable ebook or publication at runtime using HEScript. This guide will walk you through the process in a clear and concise manner. Let's get started! 🚀

The Objective 🎯

This topic explains how you can show and hide TOC items at runtime. For instance, you might want to display some TOC items only if the publication works in Registered mode.

Associating HEScript Functions to TOC Items 📝

In the TOC editor, there is a property named "Visibility HEScript Function". This property can point to a Boolean HEScript function that defines whether the TOC entry is visible or not.



Do you want to offer a Table of Contents?

Your publication can display a Table of Contents (TOC) as a **tree view**: it offers a hierarchical display of the publication's pages, allowing users to easily navigate through them. Manage what to display in the TOC below.

Properties of the TOC tree view can be modified here.

- Getting Started
- Starting a new ebook project
- Create Ebooks And Publications
  - Choosing The Rendering Engine
  - Displaying PDF and Securing PDF as Ebooks
    - Features of the Stand-Alone PDF Viewer
    - Advanced Features for the PDF Viewer
- Learn About All Features
  - File Manager
    - Adding files manually to the ebook
    - External Files
    - Source File List Update
    - File Properties

Additional Properties	
Title	Adding files manually to the ebook
URL	addfiles.html
ImageIndex	14
Open ImageIndex	14
Target	
Expanded	☐
Visibility HEScript Function	

The syntax should be: `[scriptname].[booleanfunctionname]`

When the publication loads the TOC, if a TOC item is associated with a Boolean function, the latter is executed.

If its result is True, the TOC item is displayed; otherwise, if the result is False, the TOC item is hidden.

To refresh the TOC, you can use the `RefreshTOC` HEScript built-in macro.

Example

This sample hides TOC items if the publication works in Trial mode, and shows them if it is in Registered mode.

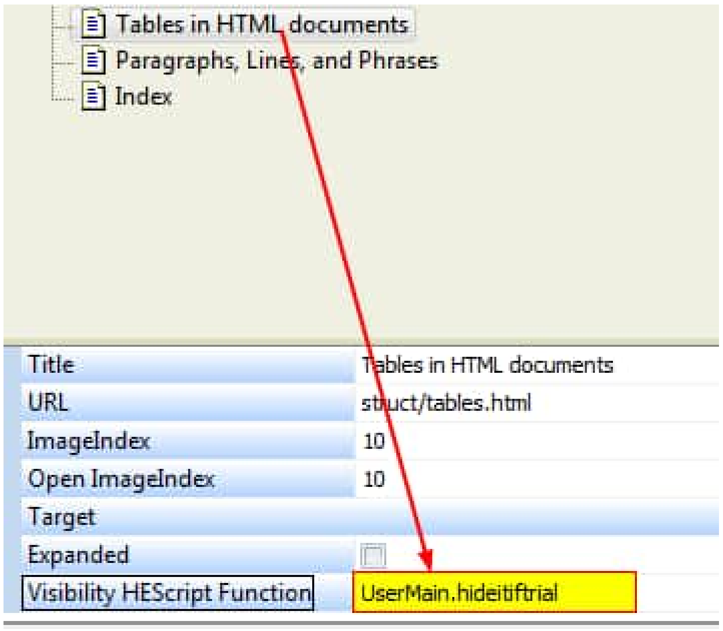
Step 1: Edit the UserMain Script

Go to the Script Manager, edit the "UserMain" script and copy/paste the following code:

```
function HideItIfTrial: Boolean;
begin
// Returns False to hide the TOC item if HEPubRegistered is set to 1 (Registered)
Result := GetGlobalVar("HEPubRegistered", "0") = "1";
end;
```

Step 2: Choose the TOC Item

Close the script editor, go to the Table of Contents page and choose the TOC item you want. Enter the reference to the script function: `UserMain.HideItIfTrial` or select it with the "..." button. Press Enter to save changes.



Title	Tables in HTML documents
URL	struct/tables.html
ImageIndex	10
Open ImageIndex	10
Target	
Expanded	☐
Visibility HEScript Function	UserMain.hideitiftrial

Step 3: Enable the Trial Publication Option

Make sure you have enabled the option to make a Trial publication and compile it.

5.2. HTML Executable JavaScript API

In this topic, we explore the powerful JavaScript Internal API provided by HTML Executable for your ebooks and applications. This API allows you to control your applications using JavaScript, offering seamless integration with your HTML pages.

HTML Executable features its own script language called HEScript, enabling you to manage your applications effectively. In addition to HEScript, you can harness the power of JavaScript to control your application through the **htmlexe** object.

To close your application programmatically, utilize the following JavaScript snippet:

```
htmlexe.RunHEScriptCode('ExitPublication;');
```

⚠ **Warning:** The JavaScript extension employs asynchronous callbacks. Consequently, the functions listed below do not return results directly. The actual result is obtained through a callback mechanism, as explained below.

List of htmlexe Methods

Method Name	**Prototype**	**Description and Comments**
CloseCurrentWindow	`procedure htmlexe.CloseCurrentWindow();`	Closes the current window, which may prompt end users.
GetGlobalVariable	`function htmlexe.GetGlobalVariable(name, defvalue, callback)`	Retrieves the value of a global variable by name. If not found, the defvalue is returned. The result is obtained through the specified callback function. Similar to GetGlobalVar HEScript function.
SetGlobalVariable	`procedure htmlexe.SetGlobalVariable(name, value, isstored)`	Sets the value of a global variable with the option to make it persistent. Similar to SetGlobalVar HEScript function.
RunHEScriptCom	`function htmlexe.RunHEScriptCom(comline)`	Executes an HEScript function or procedure and returns its result (for functions). This is akin to the hescript:// protocol with the added ability to specify the command line via `comline` and a default value in case the function is not found.
RunHEScriptCode	`function htmlexe.RunHEScriptCode(code)`	Compiles and executes HEScript code specified by `code` directly.

GetHEScriptCom	`function htmlexe.GetHEScriptCom(comline, callback)`	Executes an HEScript function and retrieves its result. Similar to the hescript:// protocol, but with the ability to specify the command line via `comline`. The result is obtained through the specified callback function.
GoToPage	`procedure htmlexe.GoToPage(url, window);`	Navigates the application to the specified URL in the named window.
GetString	`function htmlexe.GetString(name, callback);`	Retrieves a resource string by its name. The result is obtained through the specified callback function. Useful for localization.

Examples

Calling an HEScript Function from JavaScript

You can call an HEScript function from JavaScript using this function:

```
function executecom(sname) {
    return htmlexe.RunHEScriptCom(sname);
}
```

For example, to call the HEScript procedure `UserMain.Procedure1`, use either of the following:

```
javascript:executecom("UserMain.Procedure1")
```

```
javascript:htmlexe.RunHEScriptCom("UserMain.Procedure1")
```

Similar to the hescript:// protocol, parameters can be passed like this:

```
javascript:htmlexe.RunHEScriptCom("UserMain.Procedure1|Param1|Param2")
```

Displaying a Resource String in HTML

To display a resource string in an HTML page, use the following code:

```
<script language="JavaScript">

function DemoCallback(content) {
  document.getElementById('demo').outerHTML = '<p>' + content + '</p>';
 }
</script>

<div id="demo"></div>

<script language="JavaScript">
htmlexe.GetString("[Name]", DemoCallback);
```

```
</script>
```

Replace `[Name]` with the desired resource string's name.

Retrieving the Value of a Global Variable

```
<script language="JavaScript">
function DemoCallback(content) {
    alert(content);
}
</script>
<script language="JavaScript">
htmlexe.GetGlobalVariable('[Name]', DemoCallback);
</script>
```

Replace `[Name]` with the name of the global variable. Refer to the list of pre-defined global variables for more information.

Calling an HEScript String Function and Retrieving Its Result

JavaScript code:

```
<script language="javascript">

function DemoCallback(content) {
    alert(content);
}

htmlexe.GetHEScriptCom('usermain.getit|password1', DemoCallback);

</script>
```

This code calls the HEScript string function defined in UserMain:

```
function getit(what: String): String;
begin
  Result := MD5OfAString(what);
end;
```

Closing the Application with JavaScript and RunHEScriptCode

To close the application programmatically, use:

```
htmlexe.RunHEScriptCode('ExitPublication;');
```

For more information, explore the following topics:

- Extend Functionality with HEScript
  - HEScript Script Editor
  - HEScript Function Reference
  - Run and Call HEScript From JavaScript And HTML
  - UserMain Script And Script Templates
  - Quickly Add HEScript Code
  - Sample Scripts

5.3. Special Targets for External Links

Hyperlinks may have a target. HTML Executable provides you with **some pre-defined targets** that you can use not only for hyperlinks in your publications, but also for menu items, TOC entries...

Syntax:

```
<a target="[target]" href="[destination URL]">Your link</a>
```

List of predefined targets:

- `_heopenit`: extracts the file specified by [destination URL] (it must be the virtual path starting from the root without the protocol: `MyPath\MyFile.ext`) and runs the associated program to open this file. The file is then deleted when the publication is closed.
- `_blank`: this target creates a popup window.
- `_heexternal`: forces the application to open the destination URL in the default web browser (Internet Explorer or Mozilla Firefox for instance).
- `_henewinstance`: opens a new stand-alone instance of the application that will display the specified URL.
- `_hemain`: opens the destination in the main window.
- _parent: refers to the parent window.

How to open a URL in the default web browser

This will start the end user's default browser to view gdgsoft.com:

```
<a target="_heexternal" href="https://www.gdgsoft.com">View gdgsoft.com</a>
```

How to open a new instance

This will start a new instance and display "index.htm":

```
<a target="_henewinstance" href="index.htm">My link</a>
```

How to view a document file such as Word, Excel and so on

`_heopenit` can be used for **any document file** like executable program files, text files, Microsoft Office® files, etc...

```
<a target="_heopenit" href="mydocument.docx">Open this Word document</a>
```

5.4. Application Global Variables

Global variables play a crucial role in HTML Executable, allowing HEScript and JavaScript scripts to share data and store values within your application. They are a powerful tool for enhancing interactivity and storing/sharing data.

Managing Global Variables

You can manage global variables using two different methods:

- **HEScript**: Utilize the SetGlobalVar and GetGlobalVar functions for seamless control.
- **JavaScript**: Employ the SetGlobalVariable and GetGlobalVariable methods to manipulate global variables in your scripts.

Global variables offer improved security and functionality compared to cookies, making them an ideal choice for data sharing in your publication.

Properties

Here are some key properties of global variables:

- **Unique Names**: Global variable names must consist of alphanumeric characters without spaces. Avoid using names starting with "HE" to prevent conflicts with internal variables used by HTML Executable.
- **Persistence**: Global variables can be either persistent or temporary. Persistent variables are saved in the publication's state file and loaded in subsequent sessions, while temporary variables exist only for the current session. You can set a variable's persistence using the SetGlobalVar method.
- **Instance Isolation**: Global variables are not shared between multiple running instances of a publication. However, you can synchronize global variables across all instances by using the SynchronizeGlobalVar function.

Pre-defined Global Variables

HTML Executable provides several pre-defined global variables for various purposes. Here's a list of some commonly used ones:

Global Variable Name	Description
HELasterrormessage	When an error occurs, this variable contains the error message.
DefWinTitle	The title of the main window.
HomePage	The index page's filename.
HEPubStorageLocation	The path where the publication stores its data. You can customize

HEPubTempPath	The path to a temporary location where the publication stores its time.
HEPublicationDiskInfo	Contains the device ID returned by Windows of the USB disk on w
CurPageTitle	The title of the page which is currently displayed.
FwdButtonEnabled	Is the forward button enabled? (true/false)
BackButtonEnabled	Is the back button enabled? (true/false)
HEPublicationFile	The full path to the publication's .exe file (including filename)
HEPublicationPath	The full path to the folder that contains the publication's .exe file trailing backslash (e.g. `C:\MyPath\`).
HEPubRegistered	Boolean (0 or 1). Indicates whether the publication is registered o a Restricted Publication.
HEPubUserName	Contains the name of the registered user. Only available in a Restr
HEPubUserData1	Contains the company name of the registered user. Only available
HEPubExpired	Boolean (0 or 1). Indicates whether the trial period is over or not.
HEPubDays	Contains the number of remaining days in the trial period. Only av period, based on number of days.
HEPubRuns	Contains the number of times the publication has been started in Publication with a trial period, based on number of runs.
HEPublicationOnUSB	Boolean (0 or 1). Tells you whether the publication is on a USB dri GetManualHardwareID(1) must be invoked first. For example, usin publication may be run or not.
HEShowPDFBookmarks	Boolean (0 or 1). Enables the display of the navigation bookmark bookmarks are automatically listed. By default, it is set to 0.
HEDongleLicCount	For enkySL dongles. Contains the trial frequency value.
HEDongleTrialDay	For enkySL or enkyCT dongles. Contains the trial days value.
HEDongleExpirationDate	For enkySL or enkyCT dongles. Contains the authorization expirati

⚠ Please note that you should only read the values of these variables and avoid setting their values yourself.

With these additional global variables, you have more tools at your disposal to customize and enhance your HTML Executable ebooks and applications.

5.5. Publication Command Line Arguments

HTML Executable ebooks and publication apps have support for some **command line arguments**: if you want to call your publication from a third-party application and display a specific page for instance, or just run it using the Start|Run option of Windows.

The following command line switches are supported: **page, mapid, setproxy, ignoreuserpos**, and **deactivate**. But you can **add your own switches** too thanks to HEScript: see below.

Command line is especially useful with the "**Only one instance of the publication**" option available in Security -> Global Protection. For instance:

- if you launch your publication again with different command line arguments, the second and subsequent invocation of the EXE doesn't start a new instance, but uses the one that's already running and only changes the page displayed in it according to the command line.
- if you create an HTML exe application myapp.exe, you can get myapp.exe to start with a certain page by using `myapp.exe mapid 1024` where 1024 is the mapid for the page you want. If you then call `myapp.exe mapid 1025` the first instance of the app will show the page with mapid 1025.

page

Syntax:

PUBFILE.EXE page [page to display]

Use this switch to specify the virtual path to the HTML page that should be displayed by the publication.

Examples:

MYPUBFILE.EXE page comments.htm

PUBFILE.EXE page "my path page1.htm"

When you have spaces, be sure to enclose the path in quotes ("").

mapid

Syntax:

PUBFILE.EXE mapid [map ID of the page to display]

This switch can be used only if a map file is created. Specify the Map ID (integer) of the page you want to display.

Example:

MYPUBFILE.EXE mapid 1014

deactivate

Syntax:

PUBFILE.EXE deactivate

Only recognized if you **enable deactivation for a certificate** which lets the end user uninstall their registration data.

setproxy

Syntax: PUBFILE.EXE setproxy Lets the end user change his proxy server for activation, deactivation or validation.

ignoreuserpos

Syntax:

PUBFILE.EXE ignoreuserpos

Normally, the end user can set size and move the publication window around. For greater comfort, the publication stores those parameters for the next time it is run. However, some users will move the window to a second monitor and, for the next running, will forget the window was on this second monitor (for instance, the latter is turned off...), so it will be considered as "lost" for them.

This command line option will open the publication screen-centered, ignoring the position previously saved. Thus, the window will be accessible.

add your own switches

If you work with HEScript, it is possible to read the parameters passed to the publication .exe file when the latter is run.

You can do this using the ParamStr internal function.

5.6. Command Line - Directives

This page is suited for advanced users: it describes how you can use HTML Executable to create and compile publications (even silently) without navigating through the steps, but with **command line switches** (useful for daily and automated builds for instance).

Understanding Command Line Switches

Command line switches are specified with a forward slash and are sometimes followed by a value. For instance, `/C` is a valid switch. If you are specifying files or folders with spaces in them, you should enclose them in quotation marks. For example, `/U "C:\Program Files\My Folder" /C`.

Managing Project Files with Command Line Options

You can pass parameters to HTML Executable when you launch it manually (for instance with the "Run" command from the Windows Start menu). For example, the following command line opens a project file: `HEBUILD.EXE "c:\mywork\myproject\myproject.hepx"`. HTML Executable will open and read the settings from the project. All you have to do is press the Compile button and the publication will be created.

HTML Executable supports several command line switches for project files:

- `/b`: Loads the project and shows the Output page.
- `/c`: Forces HTML Executable to compile the publication whose project file was specified.
- `/q`: Forces HTML Executable to exit after a successful compilation.
- `/s`: Forces HTML Executable to run silently (no progress bar).

You can combine these switches. For example, `HEBUILD.EXE "c:\mywork\myproject\myproject.hepx" /c/q/s` will force HTML Executable to compile the publication silently and then exit.

Directive Files: An Introduction

HTML Executable introduces a specific file type called "directive file". These files, given the extension .hed, are text-based and contain instructions for HTML Executable to create a publication. Directive files are useful for external applications which need to create publications or to create automatic building processes.

How Do Directive Files Work?

Directive files work like the old Windows configuration (.ini extension) files. They contain at least two sections: "General" and "Source".

The "General" Section ☑

The "General" section includes several parameters:

- **Title** (required): Defines the title for the application.
- **NewProjectFile**: Specifies the path to the project file that HTML Executable should create when compiling the application.
- **SourceFolder**: Indicates the path to the source folder.
- **DefHomepage**: Path to the default homepage of your publication.
- **HTML5Engine** (required): Specifies the HTML5 rendering engine. Values: 0: WebView2 (Microsoft Edge engine), 1: Latest CEF (Chromium), 2: CEF (Chromium V87) with old Flash support, 3: CEF V109 with MP4 support (requires MP4 patent licensing).
- **Output** (required): Defines the path to the executable file that HTML Executable will create.
- **ProjectTemplate**: If you would like to build your project starting from an existing project, then specify the full path to this project file.
- **TOCXML**: Full path to an external XML file with a Table of Contents template previously exported with HTML Executable.
- **SubFolders**: Defines whether HTML Executable should include sub-folders when adding folders and wildcards.
- **KeepExistingProjectFiles**: 0 or 1. Tells HTML Executable whether it should keep the existing files or reset the file list when adding files specified by the directive file.
- **OutputLog**: If you would like to save the compilation log to an HTML file, then specify the full path to this file.
- **StartPrompt**: Modify the prompt to be displayed at startup.
- **EndPrompt**: Modify the prompt to be displayed at end.
- **WinTitle**: Points to the title of the main window.
- **MainDlgText**: Sets the text for the main dialog box (SFX publications only).
- **FileDesc**: Sets the publication's file description in the Version Information resource.
- **FileVerNum**: Sets the publication's file version number in the Version Information resource.
- **ProdVerNum**: Sets the publication's product version number in the Version Information resource.
- **MergeRuntime**: If set to 1, then the runtime module will be merged into the publication .exe file.
- **ResetGUID**: If set to 1, then HTML Executable generates a new GUID for the publication.
- **GlobalPassword**: Optionally defines the global password for the publication.
- **ExpirationDate**: Optionally defines an expiration date. Must be in the following format: yyyy/mm/dd.

The "Source" Section ☑

The "Source" section is an additional list of files that should be included. It can contain wildcards. It must be an ordered list: each entry begins with its number in the list and points to a path or a file. Please note that all files from the source folder are already automatically added. You do not need to specify them with the Source section.

Executing Directive Files Using HTML Executable

Without using the command line, select the "File|Open Directive File" menu command. Otherwise, you need to specify the path to the directive file thanks to the command line. For example, the following command lines will execute a directive file: `HEBUILD.EXE "c:\mywork\myproject\file.hed"`. HTML Executable will open and read the settings from the directive file. All you have to do is press the Compile button and the package *c:\mywork\myproject\output\myarc.exe* will be created.

In addition, HTML Executable supports command line switches for directive files:

> `/c`: Forces HTML Executable to compile the publication according to the directive file specified previously.
> `/s`: Will hide HTML Executable when compiling (silent compilation).
> `/q`: Will force HTML Executable to exit after a successful compilation.

You can combine these switches. For example, `HEBUILD.EXE "c:\mywork\myproject\file.hed" /c/s/q` will lead HTML Executable to compile c:\mywork\myproject\output\myarc.exe silently and then exit.

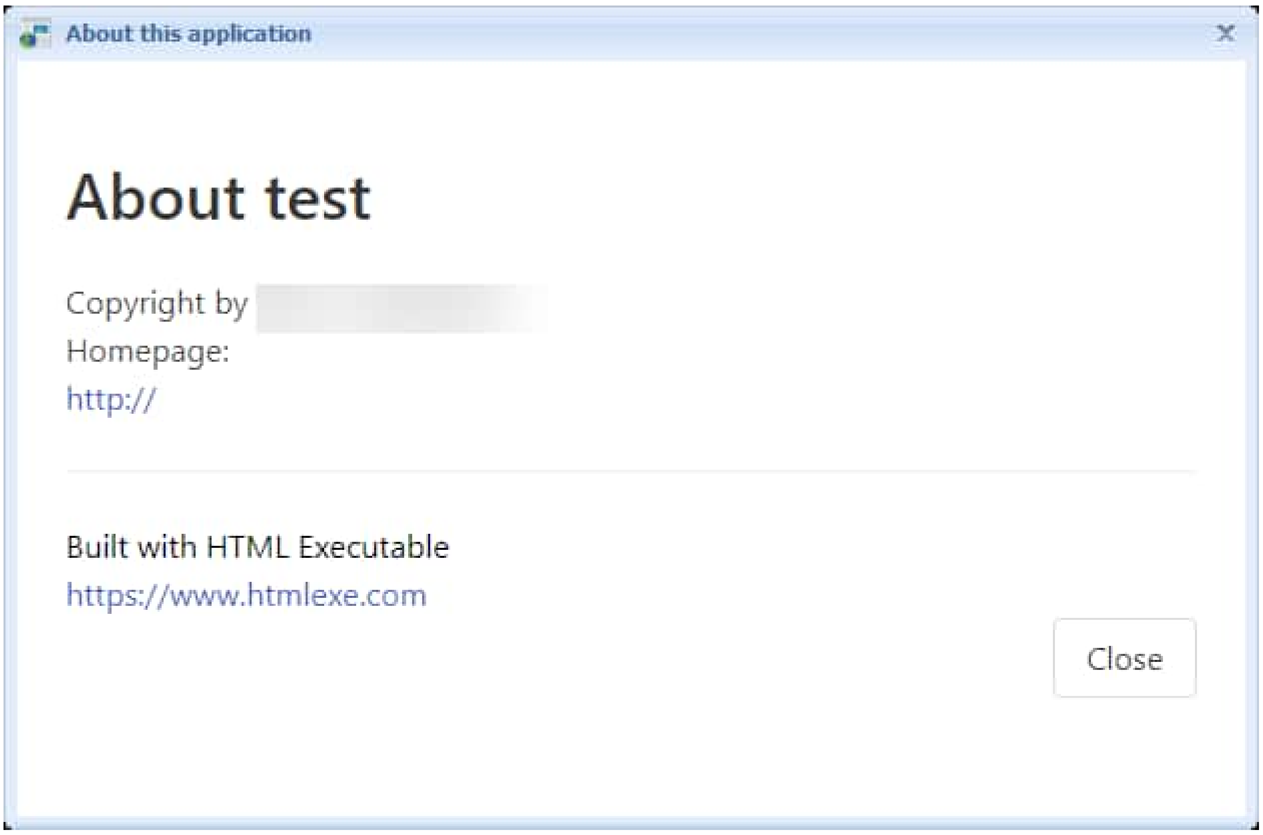
6. Frequently Asked Questions

At any time, if you have more questions, do not hesitate to **post your questions to the** user forum where you can share your feedback too.

We will update this FAQ section in future versions according to your needs.

6.1. How to modify the About box of my ebook?

In an ebook, you can display its About box by selecting "Help|About". You will get a window similar to this:


About this application

About test

Copyright by
Homepage:
http://

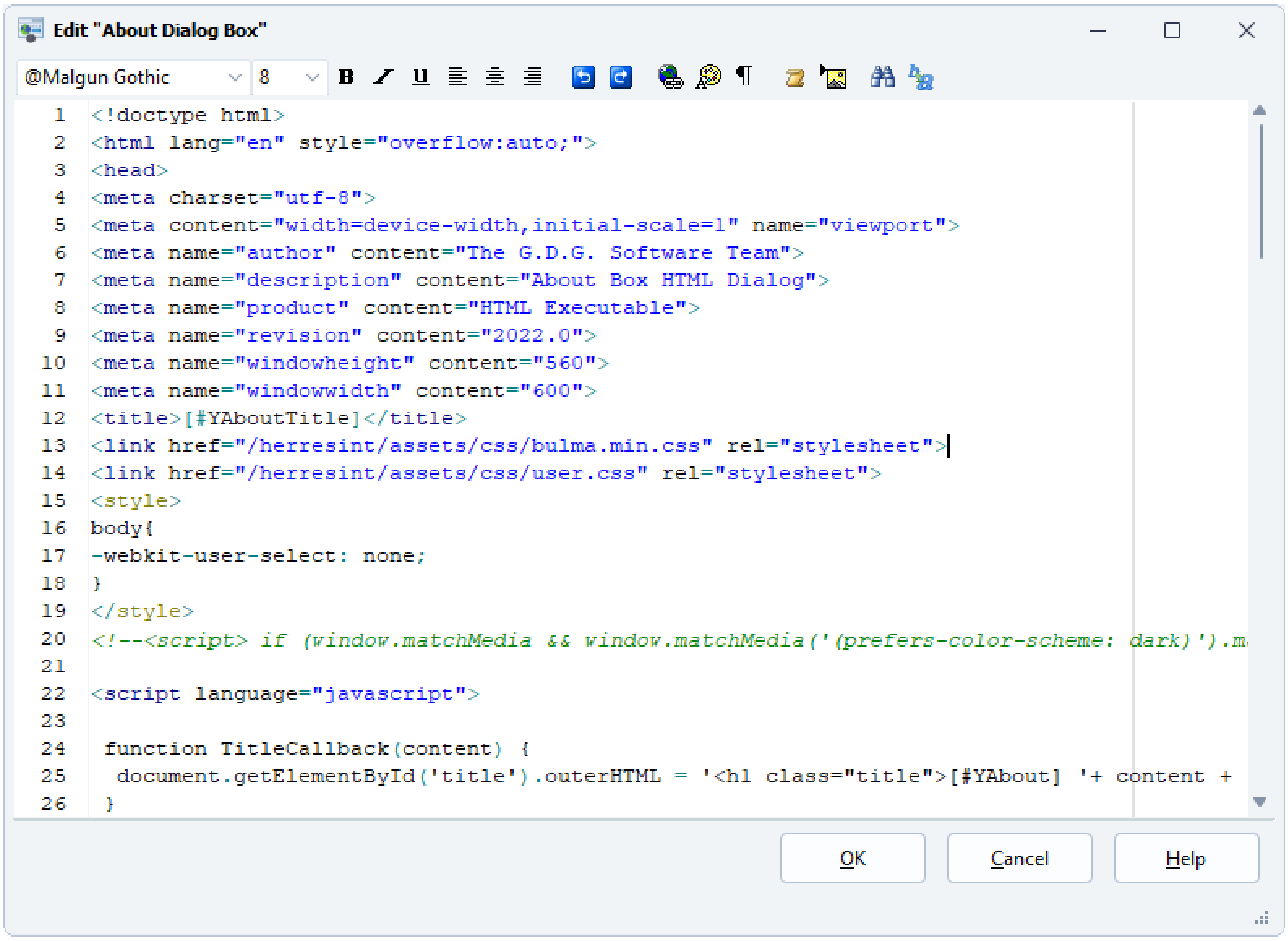
Built with HTML Executable
https://www.htmlexe.com

Each time you start a new project, HTML Executable creates a default About dialog box. But you can of course **customize it** and insert your own text, images, links... The About box is actually an HTML page.

Steps to modify the About box

In HTML Executable, open your project and

- Select "Application Behavior" and "Dialog Boxes".
- Double-click the "**About Dialog Box**" to open the HTML editor.



```
<!doctype html>
<html lang="en" style="overflow:auto;">
<head>
<meta charset="utf-8">
<meta content="width=device-width,initial-scale=1" name="viewport">
<meta name="author" content="The G.D.G. Software Team">
<meta name="description" content="About Box HTML Dialog">
<meta name="product" content="HTML Executable">
<meta name="revision" content="2022.0">
<meta name="windowheight" content="560">
<meta name="windowwidth" content="600">
<title>[#YAboutTitle]</title>
<link href="/herresint/assets/css/bulma.min.css" rel="stylesheet">
<link href="/herresint/assets/css/user.css" rel="stylesheet">
<style>
body{
-webkit-user-select: none;
}
</style>
<!--<script> if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').m

<script language="javascript">

 function TitleCallback(content) {
  document.getElementById('title').outerHTML = '<h1 class="title">[#YAbout] '+ content +
 }
```

- You can now modify the HTML code for the About dialog box. As you can see, the HTML code can contain some HEScript and JavaScript codes. You should leave this code unmodified unless you know what you do.
- Click **OK** to save the HTML code and compile your project.

Note: only users of the Professional/Commercial editions may edit the HTML code for the About box.

Insert a company logo or custom image

Since the dialog box is an HTML page, you can include any image you want.

Just use the following HTML code for instance:
```
<img src="https://heserver/hebox.png" align="right" hspace="8">
```

If your image file lies in a subfolder (e.g. `myfolder`), use the correct virtual path.

```
<img src="https://heserver/myfolder/hebox.png" align="right" hspace="8">
```

The `https://heserver/` protocol tells HTML Executable that the image is compiled inside the publication.

6.2. How do I download and compile my website into an ebook?

Learn how to download and compile your website into a single executable file for desktop using HTML Executable.

Prerequisites

In order to compile your website into a single working executable file for the desktop, HTML Executable needs source files (HTML pages, graphics, scripts...) to be available on your local computer.

Downloading Your Website

A variety of offline website download tools exist, such as the free **HTTrack** available at HTTrack's official website. HTTrack allows you to download a World Wide Web site from the Internet to a local directory, building recursively all directories, getting HTML, images, and other files from the server to your computer. HTTrack arranges the original site's relative link-structure.

☑ First, create a local mirror copy of your website with HTTrack in a folder on your computer.

Compiling Your Website into an Ebook

☑ Start HTML Executable, choose New Project, and select the index page of your website in the local folder created by HTTrack.

Please note, you may prefer to keep your website online and create a custom-branded web browser for your end users to navigate.

Happy compiling! 🎉

6.3. Can I use HTML Executable to load up my existing website and create a custom browser?

Creating a Web Browser with HTML Executable

If you're looking to have HTML Executable load up your website just like a web browser, without compiling or preloading it, you're in the right place! This is particularly useful if your website is constantly changing and you don't want to hard code anything. Let's get started! 🚀

Steps to Create Your Own Web Browser

- Create a single HTML page that redirects to your website with a JavaScript redirection for instance.
- Start an HTML Executable project and compile this page into an executable (EXE) file.

And voila! You've created your own web browser with HTML Executable. This browser can feature skins, your company logo, trademarks, and more. It's that simple!

6.4. How to Enable Webcam Access Without User Prompt

To automatically grant access to the Webcam in your application project without prompting the user, you need to modify the `OnPermissionRequested` event in your HEScript UserMain script. Follow these steps:

1. Locate the `OnPermissionRequested` Event

The `OnPermissionRequested` function handles permission requests, such as camera or microphone access. You can customize it to automatically allow permissions for specific resources.

2. Modify the Script

Open your main script (UserMain) and add the following implementation to the `OnPermissionRequested` function:

```
function OnPermissionRequested(URI: string; PermissionKind: Integer; UserInitiated:
Boolean; StateDefault: Integer): Integer;
begin
  // This function handles permission requests.
  // Parameters:
  // - URI: The URI or URL associated with the permission request.
  // - PermissionKind: The type of permission being requested:
  //   - 0: Unknown permission
  //   - 1: Microphone access
  //   - 2: Camera access
  //   - 3: Geolocation access
  //   - 4: Notifications
  //   - 5: Other sensors access
  //   - 6: Clipboard read access
  // - UserInitiated: Indicates whether the user initiated the request (true) or
not (false).
  // - StateDefault: The default state for the permission:
  //   - 0: Use default behavior
  //   - 1: Allow the permission
  //   - 2: Deny the permission

  // Automatically allow camera access
  if PermissionKind = 2 then
    Result := 1 // Allow the permission
  else
    Result := StateDefault; // Use the default behavior for other permissions
end;
```

3. Explanation of the Code

PermissionKind = 2: This identifies a request for **Camera access**.
Result := 1: This line forces the function to automatically allow access to the Webcam.
StateDefault: For other permissions (e.g., microphone or geolocation), the default behavior is applied.

4. Save and Compile

Save the script changes and recompile your project to ensure the new behavior is applied.

5. Testing

Run your application and verify that Webcam access is granted automatically without prompting the user.

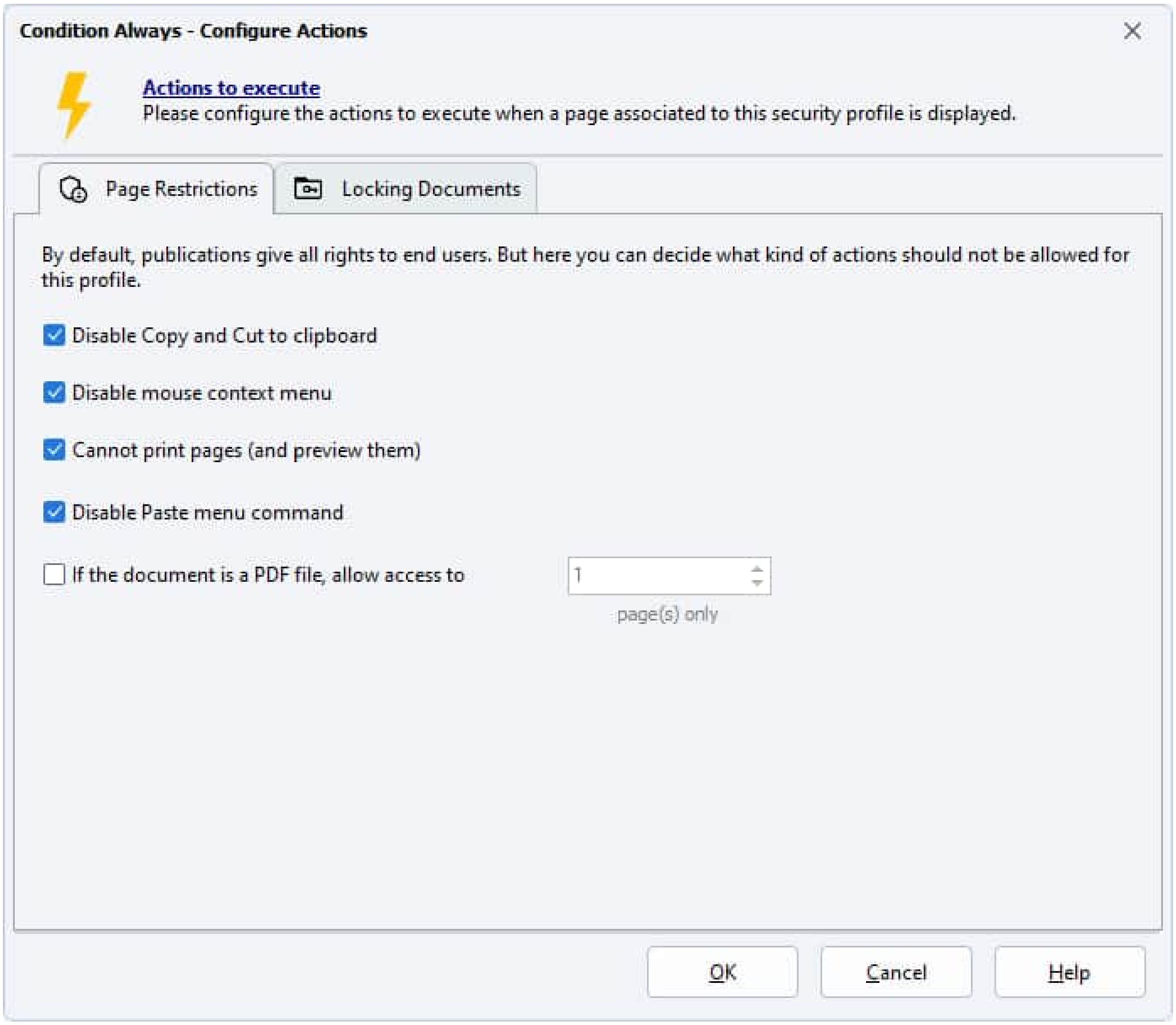
Notes:

Be cautious when automatically allowing permissions, as it can raise privacy concerns.
Adjust the logic to suit your application's security requirements, especially when handling multiple permissions.
Windows must be configured to allow the app to use the webcam.

6.5. How do I disable the Copy to Clipboard menu in ebooks

If you want to prevent end users from copying text from your ebooks to the clipboard, HTML Executable provides a versatile function known as security profiles. Here's how you can use it:

- Navigate to the **Security => Security Profiles** tab.
- Select a profile (for instance, the Default one) or create a new one.
- Choose a condition (for instance, Always) and click **Configure**.
- Enable the "**Disable Copy and Cut to clipboard**" option:


Condition Always - Configure Actions

Actions to execute
Please configure the actions to execute when a page associated to this security profile is displayed.

By default, publications give all rights to end users. But here you can decide what kind of actions should not be allowed for this profile.

- [✓] Disable Copy and Cut to clipboard
- [✓] Disable mouse context menu
- [✓] Cannot print pages (and preview them)
- [✓] Disable Paste menu command
- [ ] If the document is a PDF file, allow access to 1 page(s) only

The "Copy to clipboard" function can be restricted on any page or only the pages you decide. To learn more about how to restrict user rights on ebooks and publications using security profiles, check out our guide on security profiles.

6.6. How to use ebooks as help files for my application?

HTML Executable allows you to convert your HTML help files into standalone applications. This guide will walk you through the process of using ebooks as help files for your application.

Converting CHM to EXE

HTML help files (CHM) can be converted to HTML Executable projects with the free CHM to Exe add-on. CHM To Exe (or Chm2exe) is a free program that allows you to convert Microsoft HTML Help files (.chm) into stand-alone applications (.exe). With this program, you can open CHM files, explore them, extract all source files, or create related HTML Executable projects.

Integrating Ebooks with Your Applications

HTML Executable lets you compile help files or documentation in HTML format for your applications. You can use the HTML Executable APIs to open help publications, tell them which help topic should be displayed, close them when your application is closed, etc. This allows you to integrate publications and ebooks with your own applications.

Ebooks and publications made with HTML Executable have extensive API and command-line support:

- Each page compiled in an ebook receives a unique map ID. HTML Executable generates a list of all of these map IDs called a map file. You can configure HTML Executable to export the map file in common programming languages like Pascal, VB, C++ that can be directly integrated into your application's project.
- You can create **context-sensitive help files** by always displaying the requested page immediately even if the publication is already launched thanks to API or command-line options.
- Command line is especially useful with the "Only one instance of the publication" option (available in Security -> Global Protection). For instance, if you launch your publication again with different command line arguments, the second and subsequent invocation of the EXE doesn't start a new instance, but uses the one that's already running and only changes the page displayed in it according to the command line.

By leveraging these capabilities, HTML Executable positions itself as a comprehensive solution for creating standalone help files, significantly surpassing the functionalities of older systems like Microsoft HTML Help.

Explore the advantages of HTML Executable as an HTML Help alternative.

Finally, the HTML Executable API SDK is available to registered users on request if you want to get deeper integration.

6.7. How do I create an ebook in kiosk mode?

Publications and ebooks made with HTML Executable easily support the famous **kiosk mode** function. When a browser runs in kiosk mode, the window is maximized and there is no title bar. The F11 key is handled and supported natively.

But you may also want to always simulate the kiosk mode for your ebook applications.

To simulate a kiosk mode, go to **Application Settings**

In the **Main Window** page, **turn the following options on:**

- Display a maximized window (in state of the main window).
- Maximize on Full Screen (hides task bar)

In the **Skin Properties** page, **turn the following options on:**

- Create a window with no title bar
- Hide Caption Buttons

If you want to remove the menu bar too, go to **Visual Controls**, choose **Menu Bar** and uncheck the **Visible** property.

⚠ **Warning:** you should inform end users that they can still close your program by pressing **ALT+F4**.

ⓘ There is no way to disable the kiosk mode: if the publication starts in kiosk mode, it will always remain in that state.

6.8. How can I instruct HTMLEXE to open hyperlinks using Edge, FireFox or Chrome, and not the browser that comes with HTMLEXE?

You can open hyperlinks in an external browser thanks to the **_heexternal** target parameter.

```
<a target="_heexternal" href="index.htm">My link</a>
```

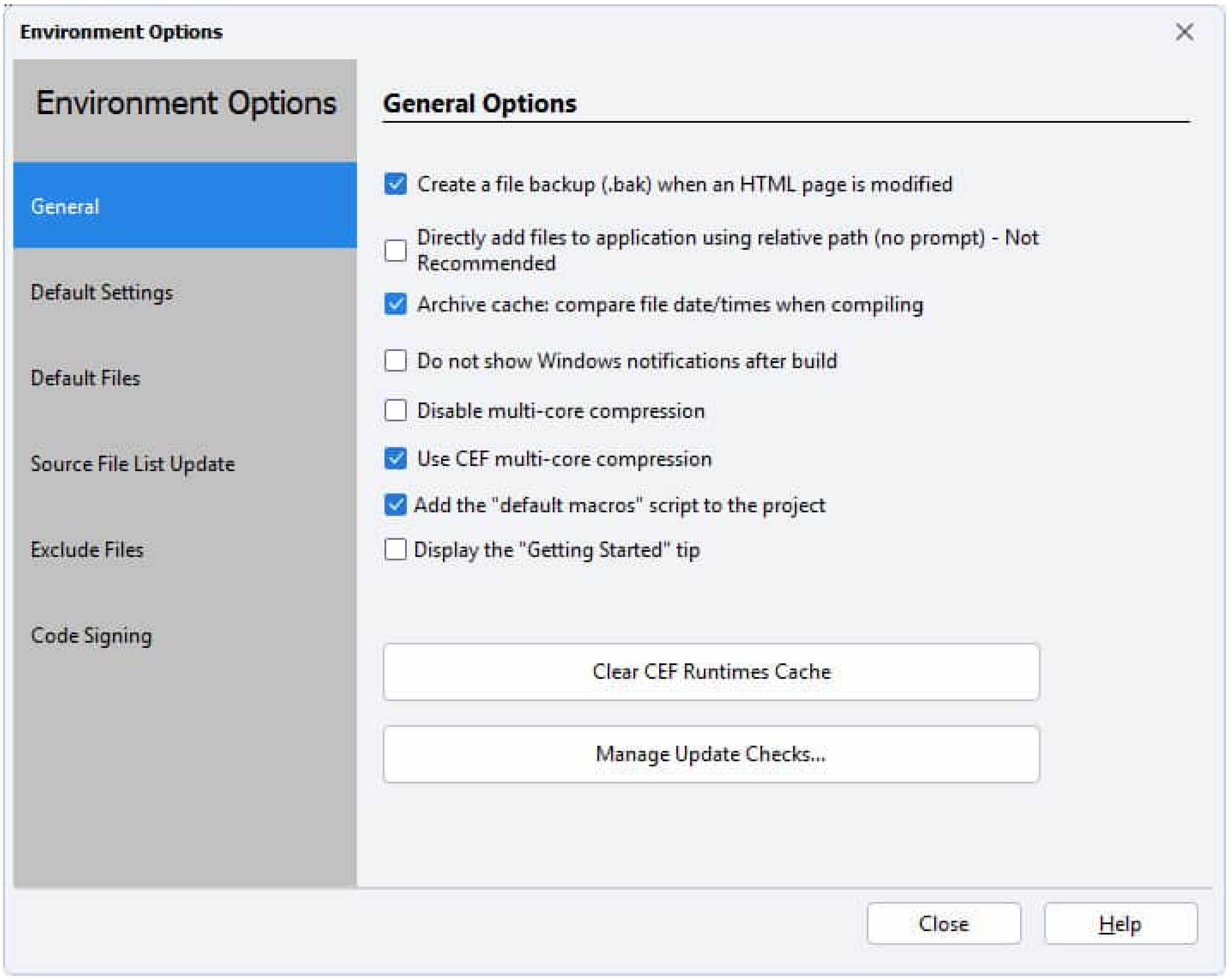
More special targets exist: see the Special Targets for Links, New Windows topic.

6.9. Can HTML Executable display Microsoft Word (DOCX) documents?

Yes, HTML Executable 2025 and later include a built-in viewer for Microsoft Word .docx files. This allows your end users to view DOCX documents directly within your compiled application, even if they do not have Microsoft Word installed. The built-in viewer also supports security features such as preventing copying, printing, and integration with security profiles. You can enable it in **Application Settings => Word Viewer**. For more details, see Displaying DOCX Files and Features of the Word Viewer.

7. Environment Options

You can customize your HTML Executable environment through setting global options, default settings, and miscellaneous properties for your HTML Executable projects. Let's review them.



General Options

- [✓] Create a file backup (.bak) when an HTML page is modified
- [ ] Directly add files to application using relative path (no prompt) - Not Recommended
- [✓] Archive cache: compare file date/times when compiling
- [ ] Do not show Windows notifications after build
- [ ] Disable multi-core compression
- [✓] Use CEF multi-core compression
- [✓] Add the "default macros" script to the project
- [ ] Display the "Getting Started" tip

You can access the Environment Options dialog box by clicking **File** and selecting the "**Environment Options**" menu command.

General Options

- **Create a file backup when an HTML page is modified**: If you edit an HTML file using the internal HTML editor, HTML Executable can backup the original file before saving the changes back to the file. The backup filename will be the same except that the extension is changed to .BAK. Recommended.
- **Directly add files to publication using the relative path**: When new files are being added manually or using the Source File List Update, HTML Executable can prompt you to specify the publication virtual path for these files, or automatically find the best virtual path and use it. This option lets you disable the prompt and lets HTML Executable decide itself for the virtual paths. However, it is NOT recommended: it is better to keep an eye on how files can be accessed in the publication at runtime.
- **Archive cache: compare file date-times when compiling**: an additional option for the archive caching feature. HTML Executable only compresses files again when they were modified or the source file lists were updated; if this option is enabled, it stores the file date-times when it compresses them for the first

time. For next builds, it then compares the file date-times with the stored ones. If a pair is different, then it will compress all source files again. This option is highly recommended.

- **Do not show Windows notifications after build**: HTML Executable displays a system notification when a build has finished. Use this option if you want to avoid that behavior.
- **Disable multi-core compression**: use this option if you have troubles while compiling your ebook.
- **Use CEF multi-core compression**: for users with strong hardware computers, you can activate this option to speed up the CEF compression.
- **Clear CEF runtimes cache**: to speed compilation up, the Chromium Embedded Framework and PHP Runtime files are compressed once and stored in HTML Executable's cache for reuse. If source files were modified, you have to clear the cache.
- **Manage Update Checks**: lets you start the Web Update utility or configure automatic update checks.

Default Settings

These settings will be used each time you start a new project.

- Author Name/Company Name: Enter your full name or even your company name.
- Web Homepage: Enter the URL to your Web site.
- Default Copyright Sentence: Enter the default copyright information for your packages.
- Default Auto-Open Extensions: The list with all the file extensions that the application should open in their external viewer application by temporarily extracting the file and launching the associated application to view it. Only files not supported by the Chromium display engines should be listed, such as Microsoft Office files. Please note, the files are temporarily extracted to the hard drive and are therefore no longer secure.

Default Files

These files will be used each time you start a new project.

Note: The default language file must exist! If the default language file is not found, HTML Executable will try to use the one in its program folder. If this one does not exist too, an error will occur.

Source File List Update

Additional options for the Source File List Update operation

- **Automatically remove non-existing files**: When turned on, HTML Executable automatically removes missing files from source file lists (a file is missing if it is not found on the hard disk at the place it should be). Otherwise, you will be prompted. Note that this will also remove the properties associated with this file, but not the possible references (in the TOC for instance).
- **Monitor the source folder for changes**: If turned on, HTML Executable will use the Windows Shell extensions to monitor if some changes happen to the root folder and its subdirectories. If something has happened, the entire file lists will be rebuilt (and the File Manager refreshed). If turned off, file lists are updated only when a live folder update occurs.
- **Perform a file list update when a project is loaded**: The source folder will be scanned for changes and file lists will be refreshed as soon as the project has been loaded.
- **Perform a file list update when a project is compiled**: The source folder will be scanned for changes and file lists will be refreshed just before the compilation of the publication really starts.

Exclude Files

You can also exclude files based on their file extension, like executables, projects, backups, and other types of files. Just add the extension(s) you wish to the list.

Notes:

- You should exclude all files associated with HTML Executable: projects (HEPX), languages (HEL), backups (BAK), skins (SKN)...
- If you want to **exclude a single file** from the compilation, you can set this in the File Properties dialog box.

Code Signing Default Properties

This page has options to define default properties for code signing your publications.

The code signing utility (GSignCode or SignTool) used by HTML Executable requires an Internet connection in order to timestamp the publication's signature. It will use the URLs defined here to contact the timestamp servers.

- **Default Authenticode SHA1 Time-Stamp Authority (TSA) URL:** The URL for an Authenticode-compatible timestamp server.
- **Default RFC 3161 Time-Stamp Authority (TSA) URL:** The URL for an RFC-3161-compatible timestamp server. This is generally preferred for modern signing.

You can configure their URLs or use the default ones provided by HTML Executable by clicking the **Default** buttons near the fields.

SignTool Configuration

- **SignTool.exe Path:** If you plan to use Microsoft's SignTool.exe for signing (either directly via "SignTool Commands" or for Azure Trusted Signing), specify the full path to SignTool.exe here. Click the browse button (folder icon) or the locate button (magnifying glass icon) to find it. SignTool is part of the Windows SDK.

- **SignTool Default Command:** You can define a default command line for SignTool. This can be useful if you frequently use the same parameters. Placeholders like **{$OUTPUTFILES}** can be used.

Azure Trusted Signing Configuration

- **Azure Trust Signing dlib Path:** If you use Azure Trusted Signing, specify the full path to the `Azure.CodeSigning.Dlib.dll` file. This DLL is part of the Trusted Signing Client Tools. Click the browse button (folder icon) or the locate button (magnifying glass icon) to find it. It's typically located in `C:\Users\[YourUserName]\AppData\Local\Microsoft\MicrosoftTrustedSigningClientTools\`.

8. Modern Alternative to Microsoft HTML Help (CHM)

For years, Microsoft HTML Help (CHM) was the standard for creating compiled help files and offline documentation on Windows. However, with CHM technology no longer being actively developed by Microsoft and facing limitations on modern systems, many developers and content creators are seeking robust alternatives. HTML Executable emerges as a **powerful and flexible solution**, offering not only the core functionalities of CHM but also a wealth of modern features for creating secure, customizable, and interactive offline documentation and ebooks.

Why Consider HTML Executable as an HTML Help Alternative?

- **Modern Rendering Engine:** Unlike CHM which relies on an outdated version of Internet Explorer's rendering engine, HTML Executable uses modern engines based on Chromium (CEF or WebView2). This ensures excellent support for HTML5, CSS3, JavaScript, and rich media, allowing for more dynamic and visually appealing help content.

- **Enhanced Security:** HTML Executable provides superior content protection. Source files are securely embedded, and you can leverage advanced security features like password protection, security profiles to restrict access or actions on specific pages, anti-screenshot measures, and even trial/licensing mechanisms if you plan to sell your documentation.

- **Full UI Customization:** Go beyond the standard CHM viewer look. With HTML Executable, you can fully customize the application window, use skins, design custom toolbars and ribbons, modify menus, and much more, creating a branded and user-friendly experience.

- **Multimedia and Document Support:** Easily embed and play audio and video content. Display PDF documents with the built-in PDF viewer and even Microsoft Word DOCX files with the built-in DOCX viewer, all without requiring external software on the user's machine.

- **Cross-Platform Compatibility (Windows):** Compiled applications run on all recent versions of Windows, from XP (with limitations) up to Windows 11, without dependencies like .NET framework for core functionality.

- **Active Development and Support:** HTML Executable is actively maintained and updated with new features and compatibility improvements, backed by dedicated support.

Familiar Help Features, Enhanced

If you're accustomed to CHM functionalities, you'll find their powerful equivalents in HTML Executable:

- **Table of Contents (TOC):** Create a hierarchical TOC for easy navigation. You can even import existing .HHC files or customize entries with icons and HTML. The TOC can also define a Browse Sequence for linear reading.

- **Search Engine:** A powerful, built-in full-text search engine indexes your HTML, PDF, and DOCX content, providing fast and relevant results with keyword highlighting.

- **Context-Sensitive Help (Map IDs and API):** HTML Executable generates Map IDs for each page, similar to CHM's map files. These can be used with command-line arguments or the HEScript API (and a dedicated SDK for registered users) to launch your help application and display a specific topic directly from your main software. This is how HTML Executable's own F1 help works!

- **Customizable User Interface:** Design the user interface with custom toolbars (Navigation Bar), menus (Menu Bar, Menu Button, Context Menu), and more. You are not limited to a fixed viewer layout.

- **Scripting for Advanced Customization:** Use HEScript to extend functionality, respond to user actions, interact with the system, and create highly dynamic help systems far beyond CHM's capabilities.

Transitioning from CHM to HTML Executable

If you have existing CHM projects, HTML Executable offers a smooth transition path:

- ✓ **CHM to Exe Add-on:** Utilize the free CHM To Exe add-on to decompile your CHM files, extract source HTML files, and even generate a basic HTML Executable project to get you started.
- ✓ **Import HHC:** As mentioned, directly import your .hhc Table of Contents file.

Conclusion

HTML Executable provides a modern, feature-rich, and secure platform for creating offline documentation and help systems. Its advanced customization options, robust security, and support for modern web technologies make it an ideal successor to Microsoft HTML Help for your Windows applications. Whether you're creating new help content or migrating existing CHM files, **HTML Executable offers the tools you need to deliver a superior user experience.**

Getting Started with HTML Executable

9. EPub to APP, import EPUB

Converting EPUB Ebooks into Secure Apps with HTML Executable

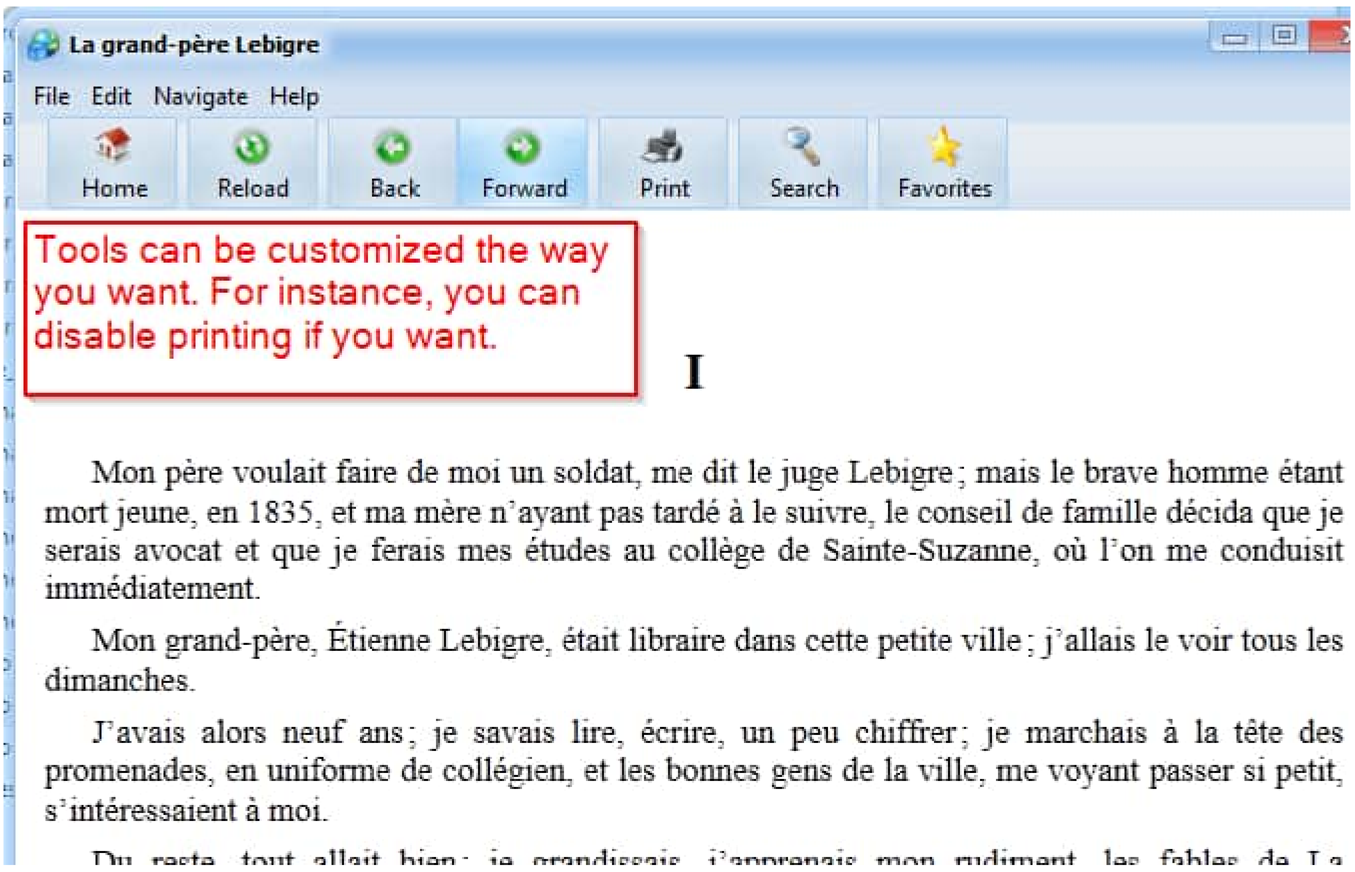
If you're looking to convert ebooks from EPUB format into secure ebooks for Windows, HTML Executable and EPub to APP have got you covered. The "EPub to APP" add-on allows you to choose an ebook in EPUB format and generates an entire HTML Executable project for you in just a few clicks. Then, you can compile your EPUB source files into standalone, secure, and customizable Windows applications using HTML Executable. Let's get started!

How to download EPub to APP and install it

EPub to APP is a **free add-on for HTML Executable**.

You can download EPub to APP from its website and install it yourself. Or use the Web Update utility shipped with HTML Executable to install the add-on.

Page navigation and formats of the initial EPUB are preserved.


La grand-père Lebigre
File
Edit
Navigate
Help
Home
Reload
Back
Forward
Print
Search
Favorites
Tools can be customized the way you want. For instance, you can disable printing if you want.

I

Mon père voulait faire de moi un soldat, me dit le juge Lebigre ; mais le brave homme étant mort jeune, en 1835, et ma mère n'ayant pas tardé à le suivre, le conseil de famille décida que je serais avocat et que je ferais mes études au collège de Sainte-Suzanne, où l'on me conduisit immédiatement.

Mon grand-père, Étienne Lebigre, était libraire dans cette petite ville ; j'allais le voir tous les dimanches.

J'avais alors neuf ans ; je savais lire, écrire, un peu chiffrer ; je marchais à la tête des promenades, en uniforme de collégien, et les bonnes gens de la ville, me voyant passer si petit, s'intéressaient à moi.

Du reste, tout allait bien ; je grandissais, j'apprenais mon rudiment, les fables de La

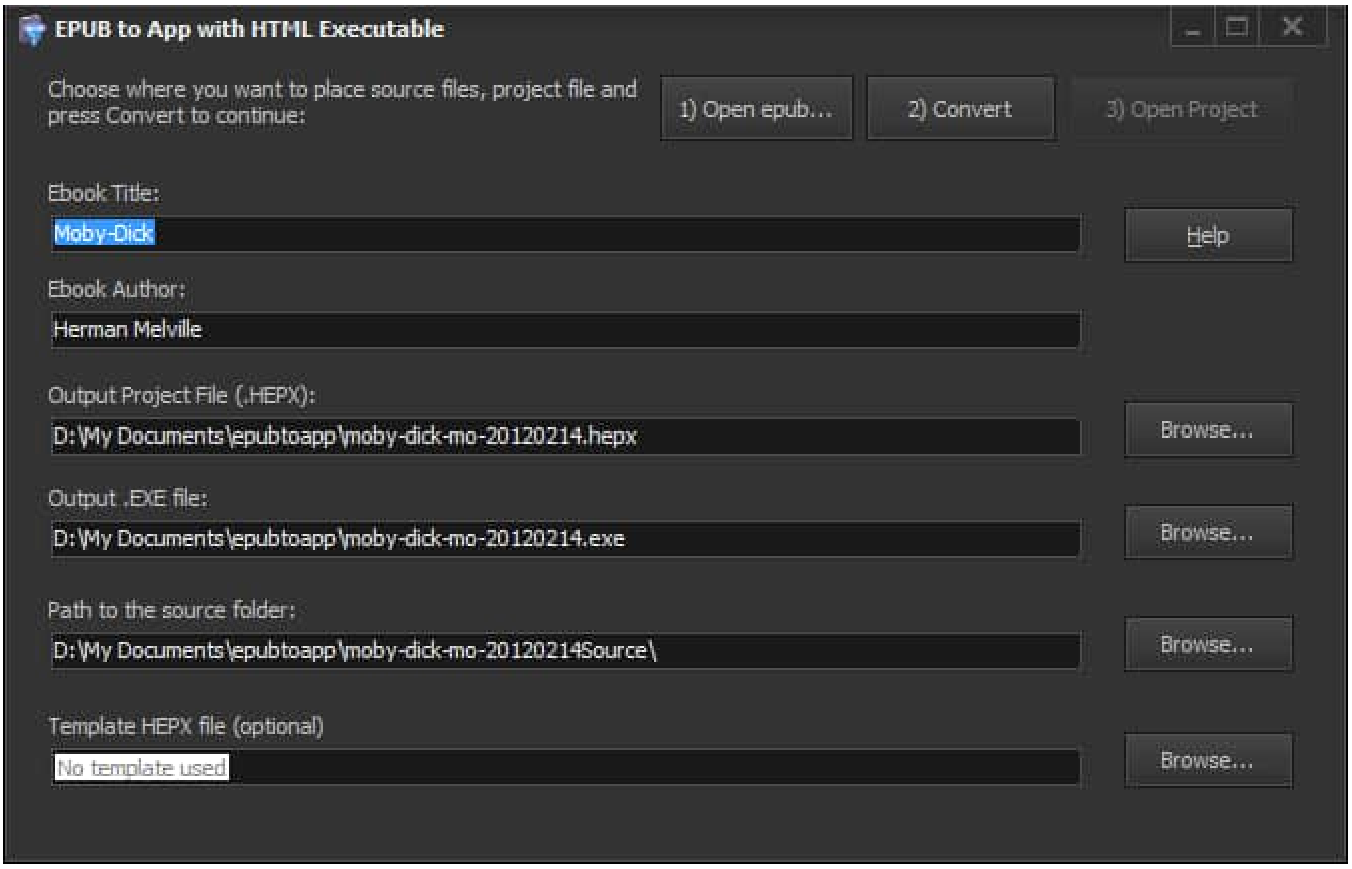
Some possibilities:

- Protect your EPUB source files against decompilation and copying once they are compiled in an application file.
- Make user-attractive and customized ebooks thanks to the features of HTML Executable: skin, menus, toolbars, windows... may be changed the way you like.
- Turn your EPUB ebooks into trial ebooks so you can sell them.
- Disable print, select, copy, print screen key commands.

- Add your own menus and toolbar buttons to your ebook navigator.
- Distribute your EPUB ebook files safely: digital signatures with Authenticode (certificates) are supported.

How to Convert EPUB to APP

- Click "**1) Open epub**" and choose the .EPUB file you want to import. EPub to APP reads properties of the selected .EPUB file and displays the title and the author of the ebook.
- Choose the HTML Executable project file that you want to create (.HEPX file) and the **source folder**: this is the folder where the content from the .EPUB file will be unpacked to.



Choose where you want to place source files, project file and press Convert to continue:

Ebook Title: Moby-Dick

Ebook Author: Herman Melville

Output Project File (.HEPX): D:\My Documents\epubtoapp\moby-dick-mo-20120214.hepx

Output .EXE file: D:\My Documents\epubtoapp\moby-dick-mo-20120214.exe

Path to the source folder: D:\My Documents\epubtoapp\moby-dick-mo-20120214Source\

Template HEPX file (optional): No template used

✓ Use the Browse button to select another location and EPub to APP will prompt you whether you want to automatically update other fields.

Click "**2) Convert**": EPub to APP creates HTML pages extracted from the EPUB ebook file, and it generates the associated HTML Executable project.
Click "**3) Open Project**": HTML Executable is launched in order to open the project. You can compile your ebook and run it.

Additional Features

- **Template HEPX File**: You can use an existing HTML Executable project file as a template for your converted EPUB file. This template file (HEPX extension) must have been created with HTML Executable 4.5 at least. Existing settings such as toolbar options, splash screen, security profiles... will be kept. Basic settings (like window title) and source files will be replaced.
- **Hints**: If you do not know HTML, ePUB ebooks can be written with editors like [Sigil] and easily transformed into secure ebooks with HTML Executable. Only DRM-free EPUB ebooks (according to the [IDPF specification]) can be converted by EPub to APP. EPUB 2.0 and 3.0 are accepted by the converter. Navigation order (of EPUB ebooks) is preserved and converted in a Browse Sequence.

10. Using Web Update

Stay Up-to-Date with HTML Executable's Web Update Utility

HTML Executable offers a utility called Web Update to help you ensure that your version of HTML Executable always has the latest updates. Not only does it keep your software current, but it can also provide you with additional resources, making your work easier and more efficient.

What Does Web Update Do?

The Web Update tool scans your current HTML Executable installation and provides a curated selection of updates that are relevant to the HTML Executable version installed on your system.

The utility's abilities aren't limited to basic updates, though. It can also fetch extra files not included in the original distribution, like additional tutorials, different CEF flavors, skin editor, documentation, or resources files. This way, you can personalize your HTML Executable experience by choosing the additional packages you want to download and install on your computer.

Please be assured that during this entire process, no private or personal data is transmitted to the server.

How to Use Web Update?

Using the Web Update utility is a simple and straightforward process:

- Run HTML Executable.
- Click on the "Tools" menu.
- Select "Launch Web Update".

By following these steps, you can access and benefit from all the latest updates and features that HTML Executable has to offer.

11. Contact Information

If you need the latest version of HTML Executable and add-ons, or if you have questions, ideas, suggestions, or need help, here's how you can reach us:

Visit our **website:** HTML Executable for the latest version of HTML Executable and add-ons.

For questions, ideas, suggestions, or if you just need help, do not hesitate to visit our forum.

Follow us on X/Twitter: @gdgsoft.

If you have questions, bugs to report, or feedback regarding HTML Executable, you can reach us by E-Mail at Contact Us. Please indicate your full name, a valid E-Mail address, the version of HTML Executable (and your computer configuration if this is a bug report). Registered users should also provide their **user ID to obtain priority for technical support**.

For more information, visit G.D.G. Software.