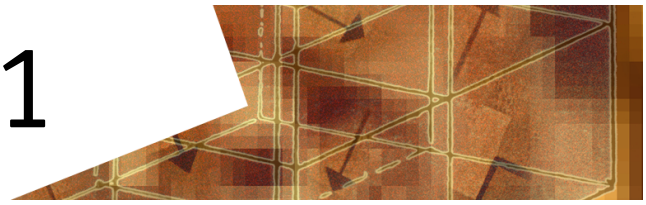

MenuBox™
CLOANTO®

MenuBox

by Cloanto Corporation

Table of Contents

Part 1



1 Introducing MenuBox

Welcome to MenuBox

Thank you for choosing MenuBox for your AutoRun projects and for your professional menu window design needs. It is our mission to deliver not only the most practical, reliable and easy to use tool in its field, but also the finest documentation and support. Should you ever have questions, comments or special requirements, please do not hesitate to let us know. We always appreciate your feedback.

Your Cloanto Team

Getting Started

Please refer to the following sections of this documentation for instructions on creating your first projects:

- Overview
- The MenuBox Wizard
- The Sign Project Tool
- Registering MenuBox
- Quality Checklist

Technical Reference

The following sections cover in more detail the operation and configuration of MenuBox, making it possible to manually build sophisticated projects, or to further customize projects created with the MenuBox Wizard:

- Command Line Options
- Configuration File Options
- Windowless Mode
- Text Window Mode
- HTML Window Mode
- The MenuBox Extended DOM
- Paths and Current Directory
- AutoRun CDs and DVDs
- Redistributable Files

Online Articles

Additional technical articles are available online. Here are some examples:

- JavaScript Error Trapping
- Using HIDEFOCUS to Remove Dotted Borders
- Using X-UA-Compatible to Set the Compatibility Mode
- Disabling Click and Other System Sounds
- Support for High-DPI Displays
- Additional Web Resources

Additional Examples and Information

To view and download additional examples created with MenuBox, or if you need more specific information about technical or licensing issues, please refer to the MenuBox Homepage (cloanto.com/menubox).

1.1 New Features

MenuBox includes the following new features.

Version 6.0.3.0

- New native 64-bit executable for Windows PE

Version 6.0.2.0

- Improved handover to default browser for external link actions

Version 6.0.1.0

- Improved application icon high-DPI support
- New context menu options in HTML window mode

Version 6.0.0.0

- High-DPI display support
- New DPIAware and DisableNavigationSound options in HTML window mode
- New ReadFile and WriteFile methods in MenuBox Extended DOM
- Finer version checking via MenuBoxBrowser key
- "Intuitive AutoRun" now uses autorun.txt with added text instructions instead of autorun.inf (which may be blocked by antivirus applications)

Version 5.2.2.0

- Fixed possible process hang on exit
- Modified behavior when Esc key is pressed in HTML window mode (it can now be processed by the script code)

Version 5.2.1.0

- Improved diagnostic messages when a [MenuBox] section is added directly to an autorun.inf file and some components cannot be found

Version 5.2.0.0

- Updated initial window when MenuBox.exe is launched in stand-alone mode (no companion files)
- Addressed "Intuitive AutoRun" configuration issue in MenuBox Wizard

Version 5.1.0.0

- Passed the official "Compatible with Windows 7" logo test.
- New browser windows opened from MenuBox now use the default system browser, rather than Internet Explorer, on Windows XP SP 2 and higher.

Version 5.0.1.0

- Modified behavior when Enter key is pressed in HTML window mode and HTML content does not have any selectable items: in this case, MenuBox does not close the window any more.

Version 5.0.0.0

- New MenuBox Wizard features: support for web applications (wrap Web 2.0 apps, etc.), "Intuitive AutoRun" (stores MenuBox as autorun.exe at root of medium, hiding all other files), and various other refinements, including new dialogs for improved handling of relative paths.
- New [MenuBox] section may now be placed in "autorun.inf" itself to reference a different directory while minimizing files at the root of the medium
- New OpenOnlyOnce and RememberPosition options in Text window mode and HTML window mode

- Changed behavior of Execute method to not return error if user cancels UAC confirmation request
- Added Exit option to Execute method of MenuBox Extended DOM
- Terminology in MenuBox Wizard, documentation, examples and website made more consistent
- Project built with new compiler (Visual Studio 2008 with Legacy Extender) and new installer (WiX toolset)
- New website

Version 4.0.3.0

- Added "Optimize for video" option in MenuBox Wizard (disables effects which might slow down video playback inside MenuBox window). Excluded non-file URLs from path processing introduced in versions 3.1.2.0 and 4.0.1.0.

Version 4.0.2.0

- Improved detection of default applications that do not have the Open action associated to a document type specified via the DocumentType key of the [ApplicationCheck] section

Version 4.0.1.0

- Software updated and independently tested to comply with Certified for Windows Vista Logo requirements
- New Sign Project tool (project signature feature) replaces and extends plain text license key in configuration file
- Added Architecture key in [ApplicationCheck] section
- Added Architecture property to the MenuBox Extended DOM
- Added /x86 and /x64 command line options
- Added Filex86 and Filex64 keys in [ApplicationCheck] and [Windowless] sections

Version 3.2.1.0

- Fixed a recently-introduced bug in the processing of environment variables in paths (the issue only occurred if the second character in a path was a backslash character, and would cause the following character to be filtered)

Version 3.2.0.0

- New Resizable and Show options in HTML window mode

Version 3.1.2.0

- Modified language detection (used for localized configuration files, and in the Language property of the MenuBox Extended DOM) to reflect current user interface language preferences, rather than the Windows default locale on multilingual systems
- Modified processing of environment variables in file system paths and in ExpandPath method of the MenuBox Extended DOM to normalize multiple consecutive backslash characters that may be present after merging path elements

Version 3.1.1.0

- Added ExpandPath method and Medium property to the MenuBox Extended DOM

Version 3.1.0.0

- Added support for nonvolatile variables (persistent settings) via new GetNV and SetNV methods in MenuBox Extended DOM
- New GetRegistry method in MenuBox Extended DOM to read registry values and to check for the existence of registry keys
- Added support for environment variables in file system paths
- Added support for volume labels (names) to FindDrive method

- New Message method in MenuBox Extended DOM

Version 3.0.4.0

- New Minimize, Maximize, Restore, Move and Size methods in MenuBox Extended DOM
- New NoExecute option in HTML window mode

Version 3.0.2.0

- New Text Selection option in HTML window mode

Version 3.0.0.0

- MenuBox Wizard to quickly build projects without having to manually edit the menubox.ini or AutoRun files
- Support for non-Latin character sets
- New FindDrive and PlaySound methods in MenuBox Extended DOM

Version 2.3.2.0

- Added Title property to the MenuBox Extended DOM

Version 2.3.1.0

- Integrated binaries with Windows Error Reporting (WER) system

Version 2.3.0.0

- Added Exists method and Language property to the MenuBox Extended DOM
- Added optional Scripting argument to MenuBoxBrowser key in [ApplicationCheck] section

Version 2.2.2.0

- Improved version detection of ActiveX objects

Version 2.2.1.0

- Minor change in handling of scrollbars on Internet Explorer 3 systems
- Modified one error message text

Version 2.2.0.0

- New ScrollBars option in HTML window mode

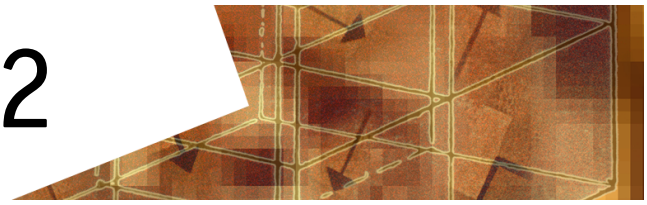
Version 2.1.1.0

- Updated examples
- Improved dynamic handling of possible dependencies on wininet.dll, olepro32.dll and oleaut32.dll on Windows 95 Retail Edition without service packs or Internet Explorer (a specific error message is now displayed if for example internet access is required by the MenuBox window content)
- Fixed a bug introduced in a 2003 recompile of version 2.0 (the original 2002 version is not affected)

Version 2.0.0.0

- New HTML window mode
- Text window mode functionality includes new effects such as fade-in and fade-out

Part 2



2 Getting Started

This section covers:

- Overview
- The MenuBox Wizard
- The Sign Project Tool
- Registering MenuBox
- Quality Checklist

2.1 Overview

MenuBox can be used in three different modes:

- Windowless (e.g. to open a PDF, Word, PowerPoint, etc. document using the CD or DVD AutoRun functionality)
- Text window (simple menus combined with graphics, mouseover information, sound, fade-in/fade-out effects, etc.)
- HTML window (HTML content in custom browser container, with options for borderless or full screen rendering, kiosk applications, etc.)

The three modes can be combined, so that if for example a certain document viewer could not be detected and/or installed, or if MenuBox is running on a very old version of Windows which does not have at least Internet Explorer 3, then the text window (which does not require any external viewers or Windows components) is opened.

The MenuBox Wizard includes preset templates to easily build projects in all three operating modes.

MenuBox Executable and Options

The redistributable MenuBox software functionality is provided by a single executable file ("menubox.exe") combined with one or more of:

- Command line option(s) (optional, with the default configuration file in the current directory being opened if no command line options are given)
- Configuration file(s) (always required when a window is opened or conditional code is used; default file names are "menubox.ini" and/or language-localized names, e.g. "menubox-en.ini", "menubox-de.ini", "menubox-fr.ini", "menubox-it.ini", etc.)
- HTML document(s) (only required if MenuBox is used in browser mode)
- Optional AutoRun files

When the MenuBox Wizard is used, the above files are created and configured automatically.

If MenuBox is used as a simple launcher (no window, no conditional code), then the command line options are all that is required (e.g. "MenuBox ReadMe.txt"). All command line options also have a configuration file equivalent, so that it is possible to keep the command line short and simple, and work on the configuration file. A configuration file (usually named "menubox.ini") is required when MenuBox is used to open a window (both for text menus and for HTML content) and when conditional code is used (for example to install viewer software or to fall back to a text window if the installation of a required viewer fails, or if browser mode is not possible due to lack of Internet Explorer 3 or higher).

HTML Documents: Windowless vs. HTML Window Mode

Unlike other types of documents, which can only be viewed by launching the appropriate viewer (which MenuBox can take care of automatically), HTML documents can be displayed in two different ways:

- From the command line (e.g. "MenuBox MyDocument.html") or using an equivalent configuration file entry, in which case the default browser and settings are used to open the document
- By indicating the desired document name and window size and settings in the MenuBox

configuration file, in which case MenuBox itself becomes a browser container (requires Internet Explorer 3 or higher, which can be installed by adding the appropriate conditions and setup instructions in the configuration file)

Related Topics

- For an overview of introductory and reference sections, see Welcome.
- For more information on registering the software, see Registering MenuBox.
- For more information about paths and directories, see Paths and Current Directory.
- For more information about commands and files distributed on AutoRun-enabled media, see AutoRun CDs and DVDs.
- For more information about redistributable files, see Redistributable Files.

2.2 The MenuBox Wizard

The MenuBox Wizard makes it possible to built MenuBox projects by using preset templates, without requiring any knowledge of the technical details involving the various MenuBox and AutoRun configuration files. Projects created with the MenuBox Wizard can then be further edited, e.g. by manually modifying the MenuBox configuration file output by the wizard.

To launch the MenuBox Wizard open **MenuBox Wizard** in the MenuBox program group of the Start menu.

Project Files

The MenuBox Wizard can be used to work on an existing set of project files (e.g. a set of HTML documents, or a complex CD AutoRun project) or to put together the files as the various details are collected by the wizard. In either case, the destination (output) folder entered in the wizard can be either the final destination of the project (e.g. a set of CD master files), if available, or a project-specific folder used only for the MenuBox portion (e.g. in "My Documents").

Files with Dependencies

If the files used by the final MenuBox project have not yet been organized into a single redistributable set, the MenuBox Wizard can copy the various items which you indicate, and copy them to the output directory as appropriate. This is usually a straightforward process, as it involves copying atomic files (e.g. a CD icon file, a viewer installation executable, etc.) However, in some cases, files may have dependencies, and MenuBox Wizard has no way of knowing that it has to copy more than one required file.

Cases in which files referenced by MenuBox may depend on other files include:

- HTML projects, where only the default page is entered into MenuBox Wizard (which does not "know" about linked files, including images)
- Executable files with linked libraries (DLLs) and other resources, in which case the main executable file name is entered into MenuBox, but other required components are not

In these cases it is important to manually make sure that the entire set of files is available in the final project directory.

Project Signature

Files created with the MenuBox Wizard include a signature in the project data. If the MenuBox configuration file (named "menubox.ini" by default) is modified manually (without using the MenuBox Wizard), the signature must be updated with the Sign Project tool.

Related Topics

- For more information about signing a project, see The Sign Project Tool.

2.3 The Sign Project Tool

The Sign Project tool creates a checksum of the MenuBox configuration file (INI file) and writes it to the Signature key of the [Project] section of the file, encoding it together with a hash of the software license information into a single non-editable binary field.

A signed project can be redistributed as a registered application, without exposing the license key in plain text, and limiting the possibility of third-party modifications to your files.

The Sign Project tool must be run each time an INI file is modified (unless it is modified by the MenuBox Wizard, which automatically updates the signature). If the signature is not correct, the project will run with a "Free Version" message.

When the sign Project tool is run, it also checks whether the MenuBox redistributable application file in the project directory is up to date. If not, the tool can optionally update this executable file with the newer version taken from the current MenuBox installation.

To launch the tool open **Sign Project** in the MenuBox program group of the Start menu.

Command Line

The Sign Project tool can also be invoked from the command line:

menuboxw.exe [/sign:file] [/?]

If the /sign option is not followed by the file name, a dialog is opened to specify the file manually.

2.4 Registering MenuBox

Limitations of the Unregistered Version

The command line options, which allow MenuBox to run in windowless mode with no conditional code, can be used with no limitations without requiring any software license details to be set.

The more advanced features made available through the configuration file may be used without purchasing the software or entering the license serial information, however in this case a small advertising window briefly appears each time MenuBox starts or exits.

The registered version of MenuBox offers the additional advantage of creating a unique publisher entity and namespace. The GetNV, SetNV, OpenOnlyOnce and RememberPosition features operate within this scope, which belongs exclusively to a given registered user and to its MenuBox applications.

After registration, projects can be signed with the Sign Project tool, which makes sure that the project can be redistributed as a registered version without the "Free Version" message, and limits the possibilities of tampering with the project itself.

Entering the License Key

To disable the "Free Version" message in the redistributable packages the license serial number must be entered during installation or in the Registration dialog which is accessible from the Start menu (in the MenuBox program group). The licensing information is automatically added to the redistributable files created by the MenuBox Wizard.

In case of manual editing (no MenuBox Wizard) of the MenuBox configuration file (named "menubox.ini" by default) which is distributed with MenuBox, the licensing information can be added or updated manually via the Sign Project tool, which writes this information to the Signature key of the [Project] section of the file.

Web Links

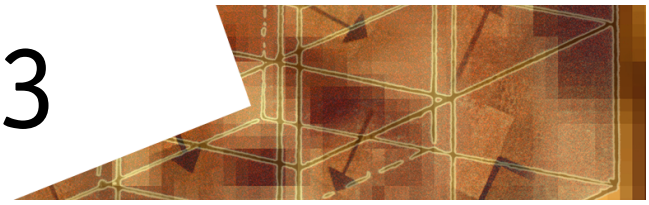
- Click here to register the software now.

2.5 Quality Checklist

	Step	Reference
1	Redistribution implies far greater risks and responsibilities than personal use. Even if you normally don't virus-check your files, we recommend that you do so before publishing your project. Make sure that your antivirus software and virus definition data are up-to-date.	
2	Test your project. Ideally, you should run the project on a different system and/or drive letter. Make sure that all links work, including links in documents referenced by MenuBox (HTML, PowerPoint, etc.)	Paths and Current Directory
3	Check for script errors. For HTML projects, test the content in IE with script debugging enabled (not disabled, as it is by default).	
4	If your MenuBox configuration files use conditional code, make sure that all conditions are tested and that the associated sections are executed.	Configuration File Options
5	If your project is designed to run as an AutoRun application, create and mount an ISO image, or burn a real CD or DVD and test the medium on as many computers as possible.	AutoRun CDs and DVDs
6	Are you using MenuBox to open documents which require a specific viewer (e.g. PowerPoint Viewer, Acrobat Reader, Shockwave Player, etc.)? Use the [ApplicationCheck] section to verify that the required viewer/version actually exists on the target system, and automatically install it or at least display a message if necessary. Check out the website of the viewer's publisher (e.g. Microsoft, Adobe, Macromedia, etc.) to determine technical and legal requirements for redistribution.	Configuration File Options
7	Test your fonts. If you use fonts, did you make sure that these fonts are available on all systems, including Windows 95 (like HTML, the [LinkFont] and [DescriptionFont] sections in text window mode support multiple font names to provide for fallback choices)? Did you test the fallback font? If you are using MenuBox in HTML window mode, are your font sizes expressed in pixels ("px" units, recommended to preserve the intended pixel-exact sizes)? If you are not sure, try opening the HTML files with Internet Explorer, and toggle the View/ Text Size setting from Largest to Smallest: the page content should not change.	Text Window Mode, HTML Window Mode

	Step	Reference
8	Don't forget extreme conditions, if applicable. Your project may run on systems with different screen resolutions. Does your window fit in 800x600 pixels? And in 640x480 (although you may decide to not support this resolution)? If you are using a full-screen window, does it display well on larger resolutions, e.g. on a 5120x2880 pixel High-DPI setup?	Text Window Mode, HTML Window Mode
9	Did you miss a feature, or did you experience a problem? Please let us know!	Contact Form

Part 3



3 Reference

This section covers:

3.1 Command Line Options

The redistributable MenuBox component consists of a single executable file named "menubox.exe", which can be run in all contexts where executable files can be launched (from the command line, from an "autorun.inf" file, etc.) The software is digitally signed using Microsoft Authenticode technology, which makes it possible to verify both the origin and the integrity of the file.

Like all Windows executable files, MenuBox can be launched either as "menubox.exe" or as "MenuBox.exe", "menubox" or "MenuBox" (case insensitive, with or without the ".exe" suffix), however indicating the complete file name (with the extension) may result in slightly faster access to the file. "menubox.exe" may be renamed (e.g. to "setup.exe" or "autorun.exe") without affecting its functionality, file properties or digital signature.

If MenuBox is launched without command line options, then the software tries to open the default configuration file ("menubox.ini" or a matching localized version, e.g. "menubox-de.ini", etc.) in the directory containing the MenuBox executable file. If "menubox.exe" is renamed then the default base name of the configuration file is changed accordingly (e.g. if "menubox.exe" is renamed to "autorun.exe" then "menubox.ini" becomes "autorun.ini", etc., as long as no /m option is used to explicitly set the name of the configuration file). A different directory may also be specified in the "autorun.inf" file itself.

The format of the command line options is:

menubox.exe [file] [/x86:file] [/x64:file] [/p:parameters] [/d:current-dir] [/v:verb] [/s:show] [/a] [/w] [/m:ini-file] [/?]

The following options are available.

Option	Description
file	Document or executable file. If omitted, MenuBox processes the default configuration file or the file referenced by /m.
/x86:file	Identical to file, but only executes the file on an x86 operating system. If present, file is ignored. May be combined with /x64 to open or execute different files on different systems.
/x64:file	Identical to file, but only executes the file on an x64 operating system. If present, file is ignored. May be combined with /x86 to open or execute different files on different systems.

Option	Description
/p:parameters	Command line parameters to be passed to the executable file, e.g. /p:"param1 param2 param3".
/d:current-dir	Changes the current drive and directory for the launched application. Supports absolute and relative paths. When launched from autorun.inf paths are relative to the root of the medium by default.
/v:verb	Action to be performed, e.g. "edit", "explore", "print", "properties", etc. The default action for most document types is "open".
/s:show	Show options, e.g. "minimized", "maximized", etc. The default is "normal".
/a	Add absolute path information to the file argument.
/w	Wait for the launched application to exit.
/m:ini-file	Process a MenuBox configuration file, which supports conditional code, menu windows and other advanced features. Defaults to "menubox.ini". Optional additional file ending with "-xx.ini" (where "xx" is language code as per ISO 639-1) is given priority if language code matches current user locale.
/?	Display help information.

The documentation of the corresponding keys as they are used in the [Windowless] section of the configuration file includes a more detailed description of each option.

Related Topics

- For more information about paths and directories, see Paths and Current Directory.
- For more information about using commands in autorun.inf, see AutoRun CDs and DVDs.
- For practical examples, see Web Resources.

3.2 Configuration File Options

Overview

The configuration file options of MenuBox include and extend the functionality provided by the command line options. A single INI file (named "menubox.ini" by default) is usually sufficient to cover most needs, including windowless mode, text window mode, HTML window mode, conditional code, application setup and fallback options. Multiple files are used to support language-localized configurations, and can also be employed for more complex conditional branches and/or to create multiple levels of text menus (just like multiple levels of HTML menus can be created using different HTML pages).

Localization (Internationalization)

Multiple INI files, each having a name ending with "-xx.ini" (where "xx" is a two-letter language code as per ISO 639-1), can be placed in the same directory as the default INI file ("menubox.ini", i.e. the same base name as the MenuBox executable file, unless a different name is specified using the /m command line option). When MenuBox starts, it first checks to see if a configuration file with a name matching the current system language exists (e.g. "menubox-en.ini" for English, "menubox-de.ini" for German, "menubox-es.ini" for Spanish, "menubox-fr.ini"

for French, "menubox-it.ini" for Italian, etc.), and then, if no match is found, it opens the file with no "-xx" language code (e.g. "menubox.ini").

There should always be a fallback file with no language code (e.g. use "menubox.ini", and not "menubox-en.ini", for English, if you want MenuBox to use the English language configuration in case no other configuration file matches the user's language). There is no need to use language-coded files if only one language is supported (e.g. use "menubox.ini", and not "menubox-en.ini", for English, if English is the only language which is supported by your project).

If the MenuBox executable file is renamed, for example from "menubox.exe" to "autorun.exe", "setup.exe", etc., then the default base name of the .ini file (if not set explicitly with the /m command line option) automatically changes accordingly: "autorun-xx.ini"/"autorun.ini", "setup-xx.ini"/"setup.ini", etc.

INI File Structure

A MenuBox INI file is an 8-bit text file (ISO 8859-1 character set) divided into sections, each containing one or more keys. Each key contains one or more values. The file can be created and modified with a simple editor like Notepad or WordPad (Save as type: Text Document).

Example:

```
[SectionName]
keyname=value
;comment
keyname=value, value, value ;comment
```

Section names are enclosed in square brackets, and must begin at the beginning of a line. Section and key names are case-insensitive, and cannot contain spacing characters. The key name is followed by an equal sign ("=", decimal code 61), optionally surrounded by spacing characters, which are ignored.

Multiple values for a key are separated by a comma followed by at least one spacing character.

Numerical values may be entered in decimal, hexadecimal (digits prefixed by "0x") or octal format (digits prefixed by a "0" character). In Boolean keys "True" and "False", and "Yes" and "No" are equivalent to 1 and 0.

When the MenuBox parser encounters an unrecognized section name, the entire section (with all its keys) is skipped. Within a known section, only unrecognized keys are skipped.

Both Space (decimal code 32) and Horizontal Tab (HT, decimal code 9) are acceptable spacing characters.

Lines are terminated by a CR (decimal code 13) and/or LF (decimal code 10) character.

Comments are introduced by a semicolon character (";", decimal code 59). Comments must begin at the beginning of a line or after a spacing character. Comments terminate at the end of the line.

The Cloanto website contains additional technical information about the INI file format as it is used in MenuBox.

Overview of Sections

In the following documentation of the MenuBox configuration file, section descriptions are grouped into four parts.

Shared settings (used with all three operation modes):

- [Project] (one per file)
- [ApplicationCheck] (multiple sections allowed)

Windowless mode:

- [Windowless] (one per file)

Text window mode:

- [TextWindow] (one per file)

- [LinkArea] (one per file)
- [LinkFont] (one per file)
- [DescriptionArea] (one per file)
- [DescriptionFont] (one per file)
- [Link] (multiple sections allowed)

HTML window mode:

- [HTMLWindow] (one per file)

[Project]

This section contains the keys used to enter the software license information and to optionally assign a unique ID to each MenuBox project.

Keys:

Signature = *String*

If you are using a registered version of MenuBox you can sign your projects to enable full functionality and disable the "Free Version" message which is otherwise displayed when MenuBox starts and exits. Click here to register the software now.

The Signature key contains licensing information combined with a checksum of the INI file content itself. This key cannot be added or edited manually. It is inserted automatically by the MenuBox wizard, and can also be inserted or updated manually by using the Sign Project tool.

If the INI file is modified, but the signature is not updated, MenuBox will work as if it were not registered ("Free Version" message).

UniqueID = *String*

The registered version of MenuBox features a publisher-unique namespace in which the GetNV, SetNV, OpenOnlyOnce and RememberPosition features operate. This avoids conflicts with settings by other publishers. GetNV and SetNV are shared (global) within this scope, which belongs exclusively to a given registered publisher and to its MenuBox applications.

The UniqueID key may be used to further define a project-specific (local) scope for the OpenOnlyOnce and RememberPosition features. The Signature key and the UniqueID key are combined so that each publisher is guaranteed to have an independent range of unique identifiers. UniqueID is not used by the GetNV and SetNV functions. UniqueID is ignored if no Signature key is indicated (which is always the case for unregistered versions of the software).

For additional per-publisher nonvolatile storage options, see the GetNV and SetNV methods of the MenuBox Extended DOM, and the OpenOnlyOnce and RememberPosition keys in the [TextWindow] and [HTMLWindow] sections.

For example:

```
[Project]
Signature = "12,34,56,78,9A,BC,DE,F0,12,34,56,78,9A"
UniqueID = "MyProduct - Version 1.0"
```

Please note that the Signature key shown above is for example purposes only. You can enter a UniqueID key, and let the Sign Project tool add the Signature key at a later stage, when you are finished editing.

[ApplicationCheck]

This section not only controls the conditional execution of code (e.g. to install a viewer or to display an error message), but also determines which of the windowless, text window or HTML window sections should be switched off (ignored) based on certain conditions. An optional confirmation or information message can be displayed to the user before the conditional launch of an executable, or before exiting MenuBox, in case no other fallback option (e.g. a text menu) is available.

The existence of an application can be verified searching by document extension (e.g. ".html") or looking for the globally unique identifier (GUID) or the programmatic identifier (ProgID) of a program or COM object. Once an application has been found based on these criteria, it can additionally be checked to be at least a given version and/or to match a known file name. The Architecture key can be used to invoke different code on x86 and x64 systems.

This section can execute code (e.g. launch the setup procedure of a document viewer) if a certain condition (e.g. the presence of a viewer for a certain document type) is not met. It can optionally check again to see if the condition is met after the conditional code has been executed. Based on these results, it is possible to disable one or more of the [Windowless], [TextWindow] and [HTMLWindow] sections in the same INI file. When two or more of these sections are active (because they have not been disabled by conditional code in [ApplicationCheck]), they are processed in the following order:

[Windowless] -> [TextWindow] -> [HTMLWindow]

Multiple [ApplicationCheck] sections may appear in the same configuration file. A [Windowless], [TextWindow] or [HTMLWindow] section only needs to be disabled by a single [ApplicationCheck] section in order to be permanently disabled (sections are disabled using an OR rule, not following an AND logic of [ApplicationCheck] success/failure results).

Keys:

AbsolutePath = *Boolean*

If this key is set to True, MenuBox inserts absolute path information at the beginning of the string referenced by the Parameters key.

This key is described in more detail in the [Windowless] section.

ApplicationFile = *String*

This key can be combined with DocumentType or ClassID to make sure that the executable file (EXE, DLL, OCX, etc.) associated with a certain document extension or ClassID has the desired file name (e.g. "iexplore.exe", "netscape.exe", "acrord32.exe", "acroread.exe", "readerx86.exe", "readerx64.exe", etc.) You can indicate multiple acceptable names by listing several comma-separated strings.

We recommend to use this feature with care, as future versions of a known application may have unpredictable file names, unless the publisher has guaranteed backward compatibility (e.g. because of widespread use in a command line context). If you know the ClassID of the specific application you are looking for, that is usually a safer way of identifying it, especially if it is officially documented by the publisher of the software.

ApplicationVersion = *String*

This key can be combined with DocumentType or ClassID to make sure that the executable file (EXE, DLL, OCX, etc.) associated with a document type or COM object has at least the specified version. The string may contain only digits and dot characters (e.g. "4", "1.2.3.4", "5.00.2195", etc.)

Architecture = *String*

This key is used to enable the section only on a desired (x86 or x64) system. The string must be either "x86" or "x64" and matches the host operating system accordingly. If the string does not match the system, the entire section is ignored. If the key is not present, the section is enabled on all systems.

ClassID = *String*

This key can indicate either the GUID of a COM object you want to check for in the "{########-####-####-####-############}" format (where "#" are hexadecimal digits, e.g. "{D27CDB6E-AE6D-11cf-96B8-444553540000}" for the Macromedia Flash ActiveX control), or the OLE short name in the "servername.typename" format (e.g. "ShockwaveFlash.ShockwaveFlash").

If you are in doubt between using DocumentType and ClassID, and you did not find official documentation concerning the identifier of the application you are looking for, you should consider using the DocumentType key to avoid the risk that the software publisher changes

the ClassID in a different product version.

The DocumentType, ClassID and MenuBoxBrowser keys are mutually exclusive (you can use only one for each [ApplicationCheck] section).

Directory = *Path*

This string indicates the directory which will become the current directory before processing the File key. By default, the current directory is the directory containing the configuration file.

This key is described in more detail in the [Windowless] section.

DocumentType = *String*

This key indicates a document file extension (including the initial dot, e.g. ".pdf", ".doc", ".pps", ".html", etc.) for which you want to verify that a registered application exists. For example, use ".doc" to verify that a program capable of opening Word documents exists (WordPad, which can read different Word formats, is an optional component of Windows, and you may want to install the free Word viewer if neither WordPad nor Word nor the Word viewer are already installed).

The DocumentType, ClassID and MenuBoxBrowser keys are mutually exclusive (you can use only one for each [ApplicationCheck] section).

File = *File Name*

If the conditional keys resulted in a failure to find a matching application, and if the user did not give a negative answer to the optional MessageAsk message, then the file referenced by this key, if present, is executed.

The key is described in more detail in the [Windowless] section.

Filex64 = *File Name*

Identical to File, but only opens or executes the file on an x64 (64-bit) operating system.

The key is described in more detail in the [Windowless] section.

Filex86 = *File Name*

Identical to File, but only opens or executes the file on an x86 (32-bit) operating system.

The key is described in more detail in the [Windowless] section.

MenuBoxBrowser = *Basic* | *Advanced, Scripting*

Use this key with the "Basic" or "Advanced" argument to verify that the system on which MenuBox is running is capable of rendering HTML contents to support the MenuBox HTML window mode. "Basic" (or "IE3") requires Windows 98 or higher, or Windows 95 or Windows NT 4.0 with at least Internet Explorer 3 (released in August 1996 and included with Windows 95 v. 950b, also known as OEM Service Release 2). "Advanced" requires Windows 98 or higher, or Windows 95 or Windows NT 4.0 with at least Internet Explorer 4 (released in September 1997). The MenuBox Extended DOM functionality requires the "Advanced" (or "IE4") level.

Alternatively to the "Basic" or "Advanced" argument, it is possible to explicitly indicate a required "browser version" (as installed on the system): "IE3" (equivalent to "Basic"), "IE4" (equivalent to "Advanced"), "IE5", "IE6", "IE7", "IE8", "IE9", "IE10", "IE11" or "IE12".

Windows 7, Windows Server 2008 and higher, as well as previous systems with Internet Explorer 8 (released in March 2009) installed (i.e. MenuBoxBrowser key set to "IE8") further support using X-UA-Compatible to set the Compatibility Mode.

Use the "Scripting" argument to verify that scripting is enabled for the security zone to which the document referenced by the URL key belongs. Local files, including files on CD and DVD media, belong to the My Computer security zone, where scripting is always enabled (if scripting were disabled for this zone by editing the registry, system functionality such as the ability to properly display help files would be compromised as well).

The DocumentType, ClassID and MenuBoxBrowser keys are mutually exclusive (you can use only one for each [ApplicationCheck] section).

MessageAsk = *String*

This key can be used to display a confirmation message before running the conditional code. For example: "Your system does not appear to be able to display PowerPoint Slide Show documents. Would you like to install the PowerPoint Viewer now?"

The message is only displayed if the conditions defined by the previous keys are not met.

MessageFail = *String*

This message is displayed if the conditions set for this [ApplicationCheck] section are not met. This key can be used with or without the File key, and with Retry enabled or disabled.

For example: "Your system does not appear to be able to display PDF documents. We recommend that you install Acrobat Reader and then open MyDocument.pdf on this CD."

Parameters = *String*

This string indicates a set of one or more parameters (e.g. a file name, or command line options) which is passed to the executable referenced by the File key.

This key is described in more detail in the [Windowless] section.

ProceedFailure = NoWindowless | NoTextWindow | NoHTMLWindow

This key, which accepts multiple comma-separated values, specifies which sections should be disabled (ignored) if the conditions set for this [ApplicationCheck] section are not met.

ProceedSuccess and ProceedFailure are often used together to disable different sections based on the outcome of the [ApplicationCheck] section. It is also possible to combine multiple [ApplicationCheck] sections, each with their ProceedSuccess/ProceedFailure key pair, for more sophisticated handling, e.g. to display Flash or HTML content depending on the availability (and/or successful installation) of the desired component, or otherwise fall back to a MenuBox text window.

ProceedSuccess = NoWindowless | NoTextWindow | NoHTMLWindow

A single configuration file may contain [Windowless], [TextWindow] and [HTMLWindow] sections. This key, which accepts multiple comma-separated values, specifies which sections should be disabled (ignored) if the conditions set for this [ApplicationCheck] section are met.

Show = Normal | Minimized | Maximized | *Custom String or Numerical ID*

This key indicates how the application referenced by the File key should be opened (minimized window, maximized window, etc.) The default behavior is always "Normal".

This key is described in more detail in the [Windowless] section.

Retry = *Boolean*

If this key is set to True, which it is by default, MenuBox waits until the execution of the application referenced by File has completed, and then repeats the verification of the conditions set for this [ApplicationCheck] section. The result of this final verification, instead of the result of the initial verification, is then used to process the MessageFail, ProceedSuccess and ProceedFailure keys. This behavior is useful to verify the successful installation of a document viewer or media player.

Verb = Open | Edit | Explore | Print | Properties | *Custom String*

This key indicates the action to be performed, e.g. "edit", "explore", "print", "properties", etc. The default action is application-specific, and for most application and document types it is "open".

This key is described in more detail in the [Windowless] section.

Related Topics

- For more detailed reference information about the INI file format, see Cloanto Implementation of INI File Format (web link).

- For more information about paths and directories, see Paths and Current Directory.
- For more information on two-letter language codes, see ISO 639-1 Language Codes.
- For an introduction to using templates to build suitable configuration files, see The MenuBox Wizard.
- For more information about signing a project, see The Sign Project Tool.
- For practical examples, see Web Resources.

3.3 Windowless Mode

Overview

In windowless mode MenuBox acts as a non-graphical document or application launcher. The functionality made available by the [Windowless] section of the configuration file, which is described below, is equivalent to that provided by the command line options.

The [Windowless] section also shares most keys which the [ApplicationCheck] section (where a file is launched based on certain conditions) and the [Link] section (where a file is launched based on user action), and with the HTML window mode's window.external.execute() method and the corresponding menubox_execute() script function, where the launching of files is controlled from within an HTML document. The only exception in this group is that the Wait key is not available in the [Link] section (where control over user interaction is always immediately returned to MenuBox) and in the [ApplicationCheck] section (where Wait behavior is implicit in the Retry action).

[Windowless]

A single, optional [Windowless] section may appear anywhere in the MenuBox configuration file. If the [TextWindow] and/or [HTMLWindow] section are also present, [Windowless] is always executed first, before a window is opened. If one or more [ApplicationCheck] sections are present, then the [Windowless] section is processed only if no [ApplicationCheck] section has disabled the [Windowless] section as a result of its ProceedSuccess or ProceedFailure keys.

Keys:

AbsolutePath = *Boolean*

If this key is set to True, MenuBox inserts absolute path information at the beginning of the string referenced by the Parameters key.

For example, if you create a DVD-ROM application where:

1. MenuBox and its configuration file are stored in the "MenuBox" directory on the DVD
2. The DVD is run from the D drive on the user's computer
3. The File key indicates "myprogram.exe"
4. The Parameters key is set to "mydocument.xyz"
5. The Directory key is set to "..\myprograms\"

MenuBox will first set the current directory to "myprograms", and then launch "myprogram.exe D:\myprograms\mydocument.xyz" (instead of "myprogram.exe mydocument.xyz", as would be the case without the AbsolutePath flag). Quote characters are recognized and/or added (if the path contains spaces) around the path and file name.

This key has been designed to support some specific applications where the document name is the first or only parameter, and in which functionality (for example the ability to reference links) is impaired unless the full (absolute) path is indicated with the document file name.

This key is optional.

Directory = *Path*

This string indicates the directory which will become the current directory before processing the File key. By default, the current directory is the directory containing the configuration file.

Be sure to indicate a directory path which does not include a drive letter (otherwise your code will not work if it is executed on a computer with different drive letters) and which is relative to the configuration file.

This key is optional.

File = *File Name*

This key indicates the name of the executable or document file to be opened. It is most frequently either used alone or in combination with a path set via the Directory key.

The string can be a file name, with or without a path (e.g. "setupviewer.exe", "viewers \ppview97.exe", "..\viewers\myreader.exe" etc.) If you do indicate a path, be sure to indicate a path which does not include a drive letter (otherwise your code will not work if it is executed on a computer with different drive letters) and which is relative to the configuration file (or the directory referenced by the Directory key, if used). You don't need to indicate a path if the file to be executed is in the same directory as the configuration file (or the directory referenced by the Directory key, if used).

If you reference an HTML file using only its name, e.g. "mydoc.html", some web browsers, including Internet Explorer, may try to open the HTTP(S) address "http://mydoc.html/" instead of the intended local file. To avoid this, be sure to always set the AbsolutePath key to True when using the File key to open local documents which are to be opened by internet-enabled viewers. This will make sure that the application knows that it has to deal with a local file rather than with an item on the internet.

In order not to introduce a difference between the current directory and the directory in which the file is stored, it is, especially for older software, most prudent to use the Directory key to specify a path, instead of indicating a path in the File key. This is because some older and/or poorly written programs can get "confused" if they need to use relative paths to open additional files which are not in the same directory in which they are running from. If the Directory key is used, then the file and path (if indicated) referenced by the File key are relative to the path of the Directory option.

Filex64 = *File Name*

Identical to File, but only opens or executes the file on an x64 (64-bit) operating system.

This key is optional. If the File key is also present, it is ignored. The key may be combined with Filex86 in the same section to open or execute different files on different systems.

Filex86 = *File Name*

Identical to File, but only opens or executes the file on an x86 (32-bit) operating system.

This key is optional. If the File key is also present, it is ignored. The key may be combined with Filex64 in the same section to open or execute different files on different systems.

Parameters = *String*

This string indicates an optional set of one or more parameters (e.g. a file name, or command line options) which is passed to the executable referenced by the File key.

This key is optional.

Show = Normal | Minimized | Maximized | *Custom String or Numerical ID*

This optional key indicates how the application referenced by the File key should be opened (minimized window, maximized window, etc.) The default behavior is always "Normal".

This key is processed by the application, not by MenuBox. If you are familiar with the specific application you are going to launch you can also use custom strings or their equivalent numerical ID. The full set of supported custom strings is: Hide, ShowNormal, Normal, ShowMinimized, ShowMaximized, Maximize, ShowNoActivate, Show, Minimize, ShowMinNoActive, ShowNA, Restore, ShowDefault, ForceMinimize.

This key is optional.

Verb = Open | Edit | Explore | Print | Play | Properties | *Custom String*

This optional key indicates the action to be performed, e.g. "edit", "explore", "print", "play", "properties", etc. The default action is application-specific, and for most application and document types it is "open".

The exact meaning of some generic verbs may vary. For example, explicitly using "open" when trying to run an executable file may, on some systems, open a hex editor displaying the content of the file, instead of causing the file to be run. In this case, not using the Verb key is the best way to ensure that an executable file referenced by the File key is actually run, rather than edited.

This key is optional.

Wait = *Boolean*

If this key is set to True MenuBox waits until the execution of the application referenced by File has completed before exiting. If the application was already open (instead of having been launched by MenuBox) then Wait has no effect.

This key is optional.

Related Topics

- For an introduction to using templates to build suitable configuration files, see The MenuBox Wizard.
- For practical examples, see Web Resources.

3.4 Text Window Mode

Overview

In text window mode MenuBox works as a stand-alone text menu application, compatible with all versions of Windows (Windows 95 and higher), without requiring any additional components to display its content. The text-based menu approach is powerful, yet lightweight and easy to master. A window can be normal, borderless or full screen, featuring a background and icon graphics, sounds, fade-in, fade-out and translucency effects (rendered on systems supporting this feature) and separate rectangular regions for menu texts and mouseover information texts. Each menu item can independently make use of the same sophisticated application and document handling options as those of the [Windowless] section.

The text window mode is configured via a [TextWindow] section, which describes window attributes, and additional sections to define the coordinates and fonts of the menu and information text regions. The actual menu items and their menu and information texts and actions are defined in one or more [Link] sections.

Image and Sound Objects

MenuBox uses different main file formats for the multimedia content used in text window mode:

- Windows bitmap format (BMP) for the window background (compatible with Windows 95, 98 and Me);
- PNG and JPEG formats for the window background (requires Windows 2000 or higher);
- Windows icon format (ICO), used in the window title, on the toolbar when the software is in use or minimized, as well as in the ALT+TAB selection window;
- Windows wave format (WAV), used when the window is opened or closed and for mouseover and mouse click effects.

BMP and PNG files may be palette-based or true color. True color images are automatically color-reduced to palette-mode if the display does not support true color, however we recommend that you try to color-reduce true color images to 256 colors (palette-based), and use the smaller 256-color bitmap file if your graphics software is able to produce a high-quality result.

ICO files should include at least the 16x16 format. Depending on the operating system version and icon size settings, the 32x32 and 48x48 formats are used mainly in the ALT+TAB selection window. The 24x24 format is used as the smallest icon instead of the 16x16 one on displays that have more than 96 DPI. The 16x16 icon is automatically resized if other formats are not available, however the result tends to be "blurry", so it is recommended to include additional native formats.

For maximum compatibility with older versions of Windows (including Windows 95), WAV sound files should be saved using either the Windows PCM (uncompressed pulse code modulation) codec or the Microsoft ADPCM (slightly lossy 4:1 compression) codec.

All coordinates are expressed in 96-DPI pixels from the top-left edge of the usable window area (i.e. excluding the title bar and window borders, if present), counting from position 0:0. By default, the usable size of the window automatically matches the size of the background image set via the BackgroundBitmap key. Objects are referenced relative to their top left corner.

All keys referencing image, icon and sound files may optionally include path information, relative to the directory containing the configuration file.

DPI Scaling

On high-DPI displays (more than 96 DPI), text window mode content is automatically scaled by MenuBox. All display units (width, height, positions) remain unchanged at 96 DPI.

If you are providing a high-resolution background image to be used for both high-DPI and lower density displays, do the following:

1. In the BackgroundBitmap key, reference an image having a pixel density that is a multiple of the canonical 96 DPI, and set the position value to Fill
2. Set the Height or Width key (but not both) to the original 96-DPI size value

For example, to display a window content that will have a size of 320x240 pixels on a "normal" display and a size of 640x480 pixels on a 192-DPI display, set a 640x480-pixel background image, while setting Width to 320 and Height to 240. The window will have the same visual size also on a hypothetical 384-DPI display, except that the image will be scaled up (looking a bit "blurred").

[TextWindow]

A single, optional [TextWindow] section may appear anywhere in the MenuBox configuration file. If the [Windowless] section is also present, [Windowless] is always executed first, before a window is opened. If one or more [ApplicationCheck] sections are present, then the [TextWindow] section is processed only if no [ApplicationCheck] section has disabled the [TextWindow] section as a result of its ProceedSuccess or ProceedFailure keys. Although it would in most cases not be practical to open both a text window and an HTML window, if both [TextWindow] and [HTMLWindow] exist in the same configuration file and remain enabled following all [ApplicationCheck] conditions, then the text window is opened first and the HTML window is opened after the first window has been closed.

Keys:

AlwaysOnTop = *Boolean*

This option ensures that the MenuBox window is always displayed on top of other windows. In full screen mode this effectively hides all other windows, even if opened by user action or by system events occurring after the launch of MenuBox.

This key is optional.

BackgroundColor = *Red Value, Green Value, Blue Value*

These three values represent the Red, Green and Blue components of the window background color, in a range from 0 to 255 (0, 0, 0 is black, 255, 255, 255 is white, 255, 0, 0 is red, etc.)

This key is optional.

BackgroundBitmap = *File Name,* Fill | Fit | Stretch | Center

This key indicates the file name of the background image to be used for the window, and its position within the window. If you need to retain compatibility with Windows 95, 98 or Me, use the BMP file format. Otherwise, feel free to use either BMP, PNG or JPEG.

The position value can be set to one of Fill, Fit, Stretch or Center (default, compatible with projects created for earlier versions of MenuBox). If necessary, Fill resizes the image so that it fills the entire window (some image parts may be cropped), while Fit resizes the image so that no parts are cropped (but some window areas may remain blank). Fill and Fit resize the image by preserving the original image ratio, while Stretch does not. Center never resizes the image.

If no Width and Height keys are used, the window size is automatically adjusted to the width and height of the image (assumed to be at 96 DPI). For optimal quality across different display densities, it is therefore recommended to set the Height and Width keys and to use an image having a pixel density that is a multiple of the canonical 96 DPI.

In Fill or Fit mode it is sufficient to indicate one bitmap dimension (Width or Height key), in which case the other is calculated at runtime to be the exact pixel size sufficient to display the full image.

This key is optional. If only the file name is set, the position value defaults to Center.

Borderless = *Boolean*

If this key is set to True, the window is opened without title bar or borders. Since such a window has no close button it is important to include an appropriate [Link] section to close the window (i.e. with the Exit flag enabled).

This key is optional.

FullScreen = *Boolean*

If this key is set to True, the window is opened in full screen (ignoring the Width and Height keys).

Due to the broad potential diversity of screen sizes and ratios, the use of full-screen text window mode is only advised when MenuBox is deployed on known hardware (e.g. internal use, custom kiosks, etc.)

This key is optional, and is usually combined with the Borderless key.

Height = *Number*

This number indicates the height of the window, excluding the title bar and borders. The value is in 96-DPI units.

This key is optional. In order to be compatible with different display densities, it is recommended to explicitly set one value (for Fill or Fit positioning) or both values (for Center or Stretch mode). If the value is not set, it is autocalculated from the Width (for Fill or Fit modes), or it defaults to the height of the image referenced by BackgroundBitmap.

Icon = *File Name*

This key indicates the file name of the icon, in Windows icon format (ICO), used in the window title, on the toolbar when the software is in use or minimized, as well as in the ALT +TAB selection window.

This key is optional. If no icon file is provided, the built-in default icon is used.

OnClickSound = *File Name*

This key indicates the file name of the sound file, in Windows wave format (WAV), to be played when a menu item is opened.

This key is optional.

OnLoadFade = *Milliseconds, From Value, To Value*

This key defines the duration, initial translucency and final translucency of the transition effect applied when the window is opened. In the translucency values 0 indicates complete transparency, and 100 complete opacity. If the final translucency value is smaller than 100 the window remains in a partially translucent state.

Not all versions of Windows support translucency. Where translucency is not supported MenuBox opens the window normally, without transition effect.

This key is optional. By default the window appears normally (no fade-in effect). If a single value is set in this key, the fade effect defaults from 0 to 100.

OnLoadSound = *File Name*

This key indicates the file name of the sound file, in Windows wave format (WAV), to be played when the window is opened.

This key is optional.

OnMouseoverSound = *File Name*

This key indicates the file name of the sound file, in Windows wave format (WAV), to be played when the mouse pointer passes over a menu item.

This key is optional.

OnUnloadFade = *Milliseconds, From Value, To Value*

This key defines the duration, initial translucency and final translucency of the transition effect applied when the window is closed. In the translucency values 0 indicates complete transparency, and 100 complete opacity. To avoid any undesired flickering, the initial translucency value should be the same as the final translucency of the OnLoadFade effect.

This effect can be disabled in by a [Link] section which closes the window, e.g. to open a new instance of MenuBox displaying a different menu level.

Not all versions of Windows support translucency. Where translucency is not supported MenuBox closes the window normally, without transition effect.

This key is optional. By default the window is closed normally (no fade-out effect). If a single value is set in this key, the fade effect defaults from 100 to 0.

OnUnloadSound = *File Name*

This key indicates the file name of the sound file, in Windows wave format (WAV), to be played when the window is closed.

This effect can be disabled in by a [Link] section which closes the window, e.g. to open a new instance of MenuBox displaying a different menu level.

This key is optional.

OpenOnlyOnce = *Boolean*

If this key is set to True, multiple instances of the same MenuBox project will open or reopen only one window. By default, MenuBox opens a new window each time it is invoked, even if it is executed with reference to the same MenuBox executable and configuration file.

This may result in multiple identical windows being opened at the same time, for example when the same AutoRun-enabled medium is repeatedly inserted and ejected. which may or may not be desirable. If you use the UniqueID key to assign a unique identifier (e.g. a sequence number, a product-unique string, etc.) to a configuration file, and you set the OpenOnlyOnce key, then multiple instances of the same MenuBox executable which refer to a configuration file with the same UniqueID will cause the first MenuBox window to be brought to the front, instead of new windows being opened.

This key is optional. In order to exclude conflicts with other publishers, the software must be registered and a project-specific UniqueID key must be set.

RememberPosition = *Boolean*

If this key is set to True, MenuBox will remember the position and size of the window when the same project is opened again. This information is stored in the user registry, within an exclusive namespace defined by the publisher registration and the UniqueID key.

This key is optional. In order to exclude conflicts with other publishers, the software must be registered and a project-specific UniqueID key must be set.

Title = *String*

This key indicates the window title text. The same text is also used on the toolbar when MenuBox is in use or minimized.

This key is optional.

Width = *Number*

This number indicates the width of the window, excluding borders. The value is in 96-DPI units.

This key is optional. In order to be compatible with different display densities, it is recommended to explicitly set one value (for Fill or Fit positioning) or both values (for Center or Stretch mode). If the value is not set, it is autocalculated from the Height (for Fill or Fit modes), or it defaults to the width of the image referenced by BackgroundBitmap.

[LinkArea]

This sections defines the rectangular region (text column) in which menu texts are displayed.

Keys:

Height = *Number*

This key indicates the height of the region. The value is in 96-DPI units.

Left = *Number*

This key indicates the position of the region, from the left margin of the window (excluding the window border). The value is in 96-DPI units.

Top = *Number*

This key indicates the position of the region, from the top margin of the window (excluding the window title). The value is in 96-DPI units.

Width = *Number*

This key indicates the width of the region. The value is in 96-DPI units.

[DescriptionArea]

This sections defines the rectangular region (text column) in which mouseover information texts are displayed. The [DescriptionArea] section is optional and does not need to be used if no information texts are used.

Keys:

DescriptionText = *String*

This key indicates the text which is displayed by default in the description area (when the mouse is not over the menu item).

This key is optional. If this key is omitted, the description area only contains text during menu mouseover activity.

Height = *Number*

This key indicates the height of the region. The value is in 96-DPI units.

Left = *Number*

This key indicates the position of the region, from the left margin of the window (excluding the window border). The value is in 96-DPI units.

Top = *Number*

This key indicates the position of the region, from the top margin of the window (excluding the window title). The value is in 96-DPI units.

Width = *Number*

This key indicates the width of the region. The value is in 96-DPI units.

[LinkFont] and [DescriptionFont]

These two sections respectively define the fonts to be used to display the menu item texts and the mouseover information texts. The [DescriptionFont] section is optional and does not need to be used if no mouseover information texts are used.

Keys:

Alignment = Left | Center | Right , Top | Middle | Bottom

This key indicates the horizontal and vertical alignment of the text within the rectangular box defined by the [LinkArea] and [DescriptionArea] sections.

This key is optional. The default value is Left, Top.

CharacterSet = *Number*

For non-Unicode strings, this key determines the character set used to map the string bytes to the actual characters in the font.

This key is optional. The default value is 0, meaning that the "ANSI" character set (Windows 1252, a variant of ISO-8859-1) is used.

DefaultColor = *Red Value, Green Value, Blue Value*

These three values represent the Red, Green and Blue components of the text color, in a range from 0 to 255 (0, 0, 0 is black, 255, 255, 255 is white, 255, 255, 0 is yellow, etc.)

Height = *Number*

This key indicates the text size (em height, in 96-DPI pixel units).

Italic = *Boolean*

If this key is set to True, the text is displayed in italics.

This key is optional. The default value is False.

MouseoverColor = *Red Value, Green Value, Blue Value*

These three values represent the Red, Green and Blue components of the color of the selected text (i.e. during mouseover), in a range from 0 to 255 (0, 0, 0 is black, 255, 255, 255 is white, 0, 255, 255 is cyan, etc.)

This key is optional. This key is only available in the [LinkFont] section (it is not used in the [DescriptionFont] section).

MouseoverUnderline = *Boolean*

If this key is set to True, the text is displayed underlined when selected (i.e. during mouseover).

This key is optional. The default value is False. This key is only available in the [LinkFont] section (it is not used in the [DescriptionFont] section).

Name = *Font Name, Font Name, Font Name, etc.*

This key indicates one or more font names, in order of preference (e.g. "ITC Officina Sans Book", "Verdana", "Arial"). If a font is not found on the system on which MenuBox is running, the software tries to open the next font in the list. The last font name should preferably refer to a font that is present on all versions of Windows, such as "Arial", "Courier New", "Times New Roman", "Symbol", "Wingdings", "Marlett", "Small Fonts", "MS Serif" and "MS Sans Serif". If no exact match is found, MenuBox opens a font using the default system default fallback logic (which for example on newer versions of Windows includes procedures which consider similarities in font names).

Underline = *Boolean*

If this key is set to True, the text is displayed underlined.

This key is optional. The default value is False.

Weight = *Number*

This key indicates the "weight" of the font. The following table lists the effect of various weigh values on the font style.

Value	Style
100	Thin
200	Extra Light
300	Light
400	Normal
500	Medium
600	Semi Bold
700	Bold
800	Extra Bold
900	Heavy

This key is optional. The default value is 400 (Normal font weight).

Width = *Number*

This key is combined with the Height key to indicate the font X:Y ratio. Characters may be "stretched" by changing the value of Width.

This key is optional. The default value is 0, meaning that the default ratio for the font should be used.

[Link]

This section defines the contents and behavior of individual menu items. Multiple [Link] sections may appear in the same [TextWindow] context.

Keys:

AbsolutePath = *Boolean*

If this key is set to True, MenuBox inserts absolute path information at the beginning of the string referenced by the Parameters key.

This key is described in more detail in the [Windowless] section.

Description = *String*

This key indicates the text which is displayed in the description area when the mouse is moved over the menu item (e.g. "Opens an Explorer window showing the contents of the CD").

This key is optional.

Directory = *Path*

This string indicates the directory which will become the current directory before processing the File key. By default, the current directory is the directory containing the configuration file.

This key is described in more detail in the [Windowless] section.

Exit = *Boolean*

If this key is set to True, MenuBox closes the window when the menu item is opened. In practice, this functionality can be used both to close the MenuBox window and to close the current menu level while a new level is being loaded by a different instance of MenuBox referencing a different configuration file.

This key is optional. The default value is False.

File = *File Name*

This key indicates the name of the executable or document file to be opened. It is most frequently either used alone or in combination with a path set via the Directory key.

The key is described in more detail in the [Windowless] section.

Parameters = *String*

This string indicates a set of one or more parameters (e.g. a file name, or command line options) which is passed to the executable referenced by the File key.

This key is described in more detail in the [Windowless] section.

Show = Normal | Minimized | Maximized | *Custom String or Numerical ID*

This key indicates how the application referenced by the File key should be opened (minimized window, maximized window, etc.) The default behavior is always "Normal".

This key is described in more detail in the [Windowless] section.

Text = *String*

This key indicates the menu text (e.g. "Browse CD"), which is displayed in the menu area.

UnLoadEffects = *Boolean*

This key determines whether the effects referenced by the OnUnloadSound and OnUnloadFade keys of the [TextWindow] section are enabled. If set to False, MenuBox exits without sound or visual transition effects, which may be useful when switching between menu levels.

This key is optional, and can only be used in combination with Exit. The default value is True.

Verb = Open | Edit | Explore | Print | Properties | *Custom String*

This key indicates the action to be performed, e.g. "edit", "explore", "print", "properties", etc. The default action is application-specific, and for most application and document types it is "open".

This key is described in more detail in the [Windowless] section.

Related Topics

- For more information on Windows character set codes, see Windows Character Set Codes.
- For an introduction to using templates to build suitable configuration files, see The MenuBox Wizard.
- For practical examples, see Web Resources.

3.5 HTML Window Mode

Overview

In HTML window mode MenuBox functions as a browser container capable of rendering HTML and other web content. This functionality requires Windows 98 or higher, or Windows 95 or Windows NT 4.0 with at least Internet Explorer 3 (released in August 1996 and included with Windows 95 v. 950b, also known as OEM Service Release 2) installed. MenuBox can render the content either in a window or in full screen, and includes special kiosk mode features such as automatic content reset after a period of inactivity and the disabling of mouse text selection and context menus (e.g. View Source, Properties, etc.)

The MenuBox HTML window mode does not require Internet Explorer to be the default browser on the system on which MenuBox is running. A condition can be created using the MenuBoxBrowser key in the [ApplicationCheck] section so that if the computer does not have the required browser functionality (Basic or Advanced) a fallback action is initiated (e.g. installing Internet Explorer software from the same CD, opening MenuBox in text window mode, displaying a message, etc.)

DPI Scaling

On high-DPI displays (more than 96 DPI), HTML window mode content can either be automatically scaled by MenuBox, or be custom-rendered by DPI-aware HTML content, as selected by the DPIAware key. The window Width and Height arguments are always expressed at the canonical 96-DPI units.

[HTMLWindow]

A single, optional [HTMLWindow] section may appear anywhere in the MenuBox configuration file. If the [Windowless] section is also present, [Windowless] is always executed first, before a window is opened. If one or more [ApplicationCheck] sections are present, then the [HTMLWindow] section is processed only if no [ApplicationCheck] section has disabled the [HTMLWindow] section as a result of its ProceedSuccess or ProceedFailure keys. Although it would in most cases not be practical to open both a text window and an HTML window, if both [TextWindow] and [HTMLWindow] exist in the same configuration file and remain enabled following all [ApplicationCheck] conditions, then the text window is opened first and the HTML

window is opened after the first window has been closed.

Keys:

AlwaysOnTop = *Boolean*

This option ensures that the MenuBox window is always displayed on top of other windows. In full screen mode this effectively hides all other windows, even if opened by user action or by system events occurring after the launch of MenuBox.

This key is optional.

Borderless = *Boolean*

If this key is set to True, the window is opened without title bar or borders. Since such a window has no close button it is important to include an appropriate menubox_close() or window.external.close() link to the page content in order to close the window.

This key is optional.

ContextMenuDefault = *Boolean*

If this key is set to True, the default browser menu is enabled. The default value is False (no browser context menu).

This key is optional.

ContextMenuText = *Boolean*

If this key is set to True, selection of text in the HTML browser window is possible, and text-related context menus are enabled. The default value is False (no text selection and no context menus).

This key is optional.

DisableNavigationSound = *Boolean*

If this key is set to True, system navigation sounds are always suppressed. This disables optional effects such as hyperlink clicks. The default value is False (sounds are enabled or disabled depending on the system settings).

This key is optional.

DPIAware = *Boolean*

If this key is set to True, no automatic scaling is performed on the HTML content. DPI-aware content can query the screen.deviceXDPI and screen.deviceYDPI DOM properties to obtain information about the current display. Unlike a web browser, the zoom value is always 100%.

If the key is set to False (default), then the content is assumed to be at 96-DPI, and automatically scaled accordingly on higher-density displays.

This key is optional.

FullScreen = *Boolean*

If this key is set to True, the window is opened in full screen (ignoring the Width and Height keys).

This key does not make the window resizable. To open a resizable maximized window with borders, use the Resizable and Show options instead, in which case the Width and Height keys can be set to define the "normal" window size. With the FullScreen option, the "normal" window size is the size of the full screen.

This key is optional, and is usually combined with the Borderless key.

Height = *Number*

This number indicates the height of the window, excluding the title bar and borders, but including an optional horizontal scroll bar. Regardless of the DPIAware key setting, the

value is always expressed in 96-DPI pixels.

This key is not required if the FullScreen key is enabled.

Icon = *File Name*

This key indicates the file name of the icon, in Windows icon format (ICO), used in the window title, on the toolbar when the software is in use or minimized, as well as in the ALT +TAB selection window.

This key is optional. If no icon file is provided, the built-in default icon is used.

KioskReset = *Seconds*

This key indicates after how many seconds of keyboard and mouse inactivity MenuBox will reset its content and again open the "homepage" referenced by the URL key.

This key is optional, and recommended for environments where MenuBox is run in kiosk mode (e.g. in full screen with a touch screen instead of a mouse, no keyboard, etc.), so that new visitors see the intended introductory page rather than the last page visited by the previous user. The default value of 0 means that MenuBox never resets its content.

NoExecute = *Boolean*

If MenuBox is used in a kiosk-like environment with general access to untrusted internet sites, it is recommended to disable the Execute functionality provided by the MenuBox Extended DOM. The NoExecute key makes it possible to deny use of the Execute functionality.

This key is optional.

NoFiles = *Boolean*

If MenuBox is used in a kiosk-like environment with general access to untrusted internet sites, it may become desirable to disable all files access functionality provided by the MenuBox Extended DOM. Although file access is limited to the publisher-specific sandbox, malicious use may result in wasteful file writes. The NoFiles key makes it possible to deny use of the ReadFile and WriteFile functionality.

This key is optional.

NoRegistry = *Boolean*

If MenuBox is used in a kiosk-like environment with general access to untrusted internet sites, it may become desirable to disable all registry access functionality provided by the MenuBox Extended DOM. The NoRegistry key makes it possible to deny use of the GetNV, SetNV and GetRegistry functionality.

This key is optional.

OnLoadFade = *Milliseconds, From Value, To Value*

This key defines the duration, initial translucency and final translucency of the transition effect applied when the window is opened. In the translucency values 0 indicates complete transparency, and 100 complete opacity. If the final translucency value is smaller than 100 the window remains in a partially translucent state.

Not all versions of Windows support translucency. Where translucency is not supported MenuBox opens the window normally, without transition effect. The introduction of transition effects causes the MenuBox window to open in layered mode. With some display cards and drivers, this may affect the quality of video playback inside the window.

This key is optional. By default the window appears normally (no fade-in effect). If a single value is set in this key, the fade effect defaults from 0 to 100.

OnUnloadFade = *Milliseconds, From Value, To Value*

This key defines the duration, initial translucency and final translucency of the transition effect applied when the window is closed. In the translucency values 0 indicates complete transparency, and 100 complete opacity. To avoid any undesired flickering, the initial translucency value should be the same as the final translucency of the OnLoadFade effect.

Not all versions of Windows support translucency. Where translucency is not supported MenuBox closes the window normally, without transition effect. The introduction of transition effects causes the MenuBox window to open in layered mode. With some display cards and drivers, this may affect the quality of video playback inside the window.

This key is optional. By default the window is closed normally (no fade-out effect). If a single value is set in this key, the fade effect defaults from 100 to 0.

OpenOnlyOnce = *Boolean*

If this key is set to True, multiple instances of the same MenuBox project will open or reopen only one window. By default, MenuBox opens a new window each time it is invoked, even if it is executed with reference to the same MenuBox executable and configuration file. This may result in multiple identical windows being opened at the same time, for example when the same AutoRun-enabled medium is repeatedly inserted and ejected. which may or may not be desirable. If you use the UniqueID key to assign a unique identifier (e.g. a sequence number, a product-unique string, etc.) to a configuration file, and you set the OpenOnlyOnce key, then multiple instances of the same MenuBox executable which refer to a configuration file with the same UniqueID will cause the first MenuBox window to be brought to the front, instead of new windows being opened.

This key is optional. In order to exclude conflicts with other publishers, the software must be registered and a project-specific UniqueID key must be set.

RememberPosition = *Boolean*

If this key is set to True, MenuBox will remember the position and size of the window when the same project is opened again. This information is stored in the user registry, within an exclusive namespace defined by the publisher registration and the UniqueID key.

This key is optional. In order to exclude conflicts with other publishers, the software must be registered and a project-specific UniqueID key must be set.

Resizable = *Boolean*

If this option is set, the window can be resized by the user.

A resizable window can also be opened maximized by setting the Show option accordingly. A non-resizable window can only be opened "maximized" by setting the FullScreen option (instead of setting the initial Width and Height values).

This key is optional, and is usually combined with the ScrollBars key. By default the window is not resizable.

ScrollBars = *Boolean*

If this key is set to True, the window is opened with horizontal and/or vertical scroll bars (as may be required by the window content). If the window content fits entirely within the window, no scroll bars are added. The default value is False (no scroll bars). If you need more detailed control over frames and parts of frames, set ScrollBars to True in MenuBox, and control the scroll bars in HTML, e.g. use the overflow (CSS) and scroll (Microsoft) properties for the BODY element, and the scrolling attribute for FRAME and IFRAME elements.

This key is optional.

Show = Normal | Minimized | Maximized

This key indicates how the window should be opened. The Maximized option is ignored if the window does not also have the Resizeable option enabled.

This key is optional. The default behavior is to open a normal window.

Title = *String*

This key indicates the window title text. The same text is also used on the toolbar when MenuBox is in use or minimized.

This key is optional.

URL = *URL*

This is the first HTML file which MenuBox processes when it is launched. It may be a simple file name (e.g. "index.html"), a path and file name (e.g. "..\docs\index.html"), or a network address (e.g. "https://example.com/index.html"). MenuBox automatically canonicalizes the URL string, so that for example the "file://" prefix for file URLs is optional. Both slash ("/") and backslash ("\") characters may be used as file system path separators. The URL specified by this key is also reopened following a "Homepage" event (e.g. triggered by the corresponding key on certain "internet keyboards"), and as set with the KioskReset key.

Width = *Number*

This number indicates the width of the window, excluding borders, but including an optional vertical scroll bar. Regardless of the DPIAware key setting, the value is always expressed in 96-DPI pixels.

This key is not required if the FullScreen key is enabled.

Differences between File System and Web Server

MenuBox HTML window mode is often used to browse through CD-based content. In this scenario the HTML documents and the other objects are read from a local file system (CD, DVD, hard disk, etc.) rather than from a web server, requiring special attention to aspects which include:

- The default document ("index.html", "default.htm", etc.) of a directory, which may be omitted when referencing the default page of a directory on a web server, must be explicitly referenced when the same link points to a location on a file system (e.g. use "holidaypics/index.html" instead of "holidaypics/")
- Server-based code (e.g. CGI, PHP and ASP) is not processed when running from a file system (i.e. without a web server to execute the code)
- Relative paths and links (e.g. "images\mypicture.jpg" or "images/mypicture.jpg" instead of "D:\images\mypicture.jpg" or "https://example.com/images/mypicture.jpg"), rather than absolute paths or URLs, should be used to link local content (i.e. files stored on the same CD, DVD, computer, etc.)
- Absolute URLs, rather than relative paths and links, should be used to link from the file system to pages stored on the internet (e.g. "https://example.com/mywebcam/" instead of "../mywebcam/")

You do normally not need to worry about:

- Client-based code (e.g. ActiveX, Java and JavaScript), which works fine when run from the file system, as long as the appropriate viewers/players/plug-ins are available (they can be installed from the file system just like they can be installed from the internet)
- Slash ("/") characters in paths, which are automatically mapped to Windows backslash ("\") characters when necessary
- File and directory names (short "8.3" file names should be a thing of the past even on CDs)
- Download times and size of images, videos and other multimedia items (everything is almost instantly available from the local file system, allowing for richer content compared to most online projects)

Additional Design Considerations

MenuBox HTML windows are often designed to look more like application windows than web pages. In these cases, you may also want to consider the following:

- Font sizes should be expressed in pixels (rather than points, percent values, etc.) to guarantee accurate positioning and sizes on a variety of systems and user settings. You can use the following formula to convert between font size units: px = pt+33%. Using Cascading Style Sheets (CSS) to specify font sizes in pixels provides a good method for creating reliably uniform font sizes.
- When linking to other pages, especially web content, decide whether such items should be displayed inside the same MenuBox window or inside a new web browser window (handled by whatever the default web browser is) instead. In the latter case, be sure to reference links using the target="_blank" option (e.g. Target Frame... New Window option in some authoring tools).

Related Topics

- For more information on the MenuBox Extended Document Object Model, see The MenuBox Extended DOM.
- For more information about paths and directories, see Paths and Current Directory.
- For an introduction to using templates to build suitable configuration files, see The MenuBox Wizard.
- For practical examples, see Web Resources.

3.6 The MenuBox Extended DOM

Overview

In addition to providing full browser functionality (referred to as "Basic" browser functionality), MenuBox extends the standard browser Document Object Model (DOM) by adding properties and methods which make it possible to open documents, launch applications and quit MenuBox with the same ease and administrative privileges (i.e. without browser-typical confirmation and warning messages) as any other trusted application. More in general, these features allow you to use HTML code to build a project that looks and feels to the user more like a full Windows application than an HTML page. This functionality (which is referred to as "Advanced") requires Windows 98 or higher, or Windows 95 or Windows NT 4.0 with at least Internet Explorer 4 (released in September 1997) installed. Code for object detection and fallback under Internet Explorer 3 is also provided.

Extended Document Object Model Reference

In order to overcome the limitations of normal browser containers, MenuBox extends the Dynamic HTML Document Object Model (DOM) so that scripts can access the same document opening and code execution functionality already known from the MenuBox command line, windowless and text window modes. Such scripts refer to the host by specifying the External object that is available from the Window object. For example, a reference to "window.external.menuboxversion" will call MenuBox to resolve the name "menuboxversion" and return the program version number. All standard script within the HTML document will be executed normally, without being affected by the additional functionality provided by MenuBox.

Methods:

Sub **Close()**

Closes the MenuBox window.

Sub **Execute(***File As String, Parameters As String, Directory As String, Verb As String, Show As String, AbsolutePath As Boolean, Wait As Boolean, Exit As Boolean***)**

Opens a document or executes a file. Paths used by this function are relative to the directory containing the HTML file referenced by the URL key (the first HTML document which is opened), rather than the directory containing the current HTML document, if different. Within an HTML scripting environment, remember to use the applicable escape sequences for any special characters which may be contained in the function arguments (e.g. "\" should become "\\", and double quotes inside HTML double-quoted parameters should be transformed to single quotes, or vice versa).

The documentation of the corresponding keys as they are used in the [Windowless] section of the configuration file includes a more detailed description of each option. Setting the Exit argument to true closes the MenuBox window after completion, and is equivalent to invoking Close().

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoExecute key.

Function **Exists(***File As String***)** As Boolean

Indicates whether a file exists. No warning or error messages are displayed to the user if

the file does not exist, or if the medium (if an absolute path is specified) is not available. Paths used by this function are relative to the directory containing the HTML file referenced by the URL key (the first HTML document which is opened), rather than the directory containing the current HTML document, if different.

Within an HTML scripting environment, remember to use the applicable escape sequences for any special characters which may be contained in the File argument (e.g. "\" should become "\\").

Function **ExpandPath(***Path As String***)** As String

Returns the full absolute path, given a relative path or an environment variable. The resulting path is guaranteed to terminate with a backslash character. The Path string is returned unmodified if it cannot be expanded to a valid path.

Within an HTML scripting environment, remember to use the applicable escape sequences for any special characters which may be contained in the Path argument (e.g. "\" should become "\\").

Function **FindDrive(***Path As String, Message As String***)** As String

Looks for a drive containing the specified path and file (e.g. "MenuBox\MyCDIdentifier.txt"), prompting the user with a custom message (e.g. "Please insert MyCD in any drive.") to insert the medium if necessary. If the path string ends with :\" (e.g. "MyCD:\") then MenuBox looks for a volume with the specified label (name).

The function either returns the path (drive letter inclusive of trailing backslash, e.g. "D:\") of the first drive containing a matching medium, or a null string, if the medium was not found. If no Message parameter is provided, the function fails without prompt if the medium is not mounted.

For performance reasons, FindDrive does not attempt to scan floppy drives.

Function **GetNV(***VariableName As String***)** As String

Returns the value of a MenuBox nonvolatile variable. A null string is returned if the value is not set.

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoRegistry key.

Function **GetRegistry(***RegistryKey As String, RegistryValue As String***)** As String

Returns the specified registry value, or indicates whether a key exists. A null string is returned if the key or value is not found. The RegistryValue parameter is optional. If it is not provided or left empty, MenuBox returns "1" to indicate that the specificed key exists, or a null string to indicate that it was not found.

When a binary value is queried, MenuBox returns a string of space-separated hex values (e.g. "00 01 02 fc fd fe ff").

When a multi-string value is queried, MenuBox returns an array object (having a lower bound of 1). The following JavaScript example illustrates how to access such an object.

```
var ret = menubox_getregistry('HKEY_CURRENT_USER\\SOFTWARE\\Example',
'Test');
if (typeof(ret) == "string")
{
    menubox_message("MenuBox Example",ret);
}
else
{
    for (var i = ret.lbound(); i <= ret.ubound(); i++)
        menubox_message("MenuBox Example",ret.getItem(i));
}
```

On x64 systems the Registry Redirector isolates 32-bit and 64-bit applications by providing separate logical views of certain registry keys. In this case, MenuBox looks up first the x64 key, and then, if no x64 key was found, the corresponding x86 key. To override this behavior the RegistryKey parameter can be prefixed with "x86\" or "x64\", in which case MenuBox only searches the specified branch.

Within an HTML scripting environment, remember to use the applicable escape sequences for any special characters which may be contained in the RegistryKey argument (e.g. "\" should become "\\", as in "X86\\HKEY_LOCAL_MACHINE\\SOFTWARE\\Publisher\ \Application\\1.0").

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoRegistry key.

Sub **Maximize()**

Maximizes the window.

Function **Message(***Title As String, Message As String, Type As Long***)** As Long

Displays a message dialog with the desired title and message content. The "\n" escape sequence can be used to force new lines in the text.

By default, a simple system message dialog with an "OK" button is displayed. The optional Type argument can be set to change the type of dialog, using the same numeric values as in the Windows system MessageBox() function (0 = "OK", 1 = "OK" and "Cancel", 2 = "OK", "Retry" and "Ignore", 3 = "Yes", "No" and "Cancel", 4 = "Yes" and "No", 5 = "Retry" and "Cancel", etc.), which can be combined (ORed) with additional flags to set specific warning or error symbols, Help options, etc.

The return value is the same as documented for the Windows system MessageBox() function (1 = OK, 2 = Cancel, 3 = Abort, 4 = Retry, 5 = Ignore, 6 = Yes, 7 = No, 8 = Close, etc.)

If no Title string is provided, MenuBox displays the message in a small "always on top" window, suitable for example for displaying initialization progress information. This type of window has to be closed explicitly, by invoking the Message() function with an empty Message string.

Sub **Minimize()**

Minimizes the window.

Sub **Move(***XPos As Long, YPos As Long, Width As Long, Height As Long***)**

Moves the window to a given screen position (starting from the 0:0 coordinate corresponding to the top left of the primary display). The Width and Height arguments are optional, and allow to set the position and size of the window in a single step.

Sub **PlaySound(***File As String, Wait As Boolean***)**

Plays a sound file in Windows wave format (WAV). For maximum compatibility with older versions of Windows (including Windows 95), WAV sound files should be saved using either the Windows PCM (uncompressed pulse code modulation) codec or the Microsoft ADPCM (slightly lossy 4:1 compression) codec. This method does normally not interrupt any previously-playing sound. In order to interrupt a sound the method must be invoked with an empty file name, after which the method can be invoked again to play a new sound. If the Wait argument is true, the function returns only after the sound file has been played to its end. The default value is False (the function returns immediately).

Function **ReadFile(***FileName As String***)** As String

Returns the content of a MenuBox data file, as written by WriteFile. A null string is returned if the file does not exist.

Only the file name, not the directory name, can be set. The directory is always the "sandboxed" publisher-specific directory under a per-user directory reserved for MenuBox data storage.

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoFiles key.

Sub **Restore()**

Restores the window (which may be in a minimized or maximized state) to its default open size.

Sub **SetNV(***VariableName As String, Value As String***)**

Defines a MenuBox nonvolatile variable within the publisher's namespace, if not already defined, and sets its value. The variable is stored in the system registry for the current user and is persistent across reboots. The variable can be read with the GetNV method.

The maximum length of a nonvolatile variable name is 254 characters. Quote, slash and backslash characters are not allowed (if used, """, "/", "\" and ":" are converted to underscore characters). The maximum size of the value is 2048 characters.

If the value is set to a null string, then the nonvolatile MenuBox variable, if it existed, is deleted. Once the last variable set by a publisher is deleted, the entire registry key containing it is also deleted with it, restoring the user's registry to a "clean" state, i.e. as it was before the first variable was set.

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoRegistry key.

Sub **Size(***Width As Long, Height As Long***)**

Sets the window size.

Sub **WriteFile(***FileName As String, Data As String***)**

Writes a file within a publisher-specific directory under a per-user directory reserved for MenuBox data storage. This directory is "sandboxed", and cannot be set in the file name. The file can be read with the ReadFile method.

The maximum length of the file name is 254 characters. Quote, slash and backslash characters are not allowed (if used, """, "/", "\" and ":" are converted to underscore characters). The maximum file size is 2 GB.

If the data is set to a null string, then the file, if it existed, is deleted. Once the last file set by a publisher is deleted, the entire publisher data directory containing it is also deleted with it, restoring the user's file system to a "clean" state, i.e. as it was before the first file was written to.

At the Windows file level, the string is written as a UTF-16 text stream. To store binary data, use an encoding like Base64.

This function can be disabled (e.g. for security considerations when opening pages in unknown and untrusted internet sites) by setting the NoFiles key.

Properties:

Property **Architecture** As String

Returns the architecture of the host operating system, i.e. "x86" or "x64".

Property **Language** As String

Returns a two-letter language code string as per ISO 639-1 (e.g. "en" for English, "de" for German, "es" for Spanish, "fr" for French, "it" for Italian, etc.) indicating the current user locale. If the current user locale is unknown, "en" is returned.

Property **Medium** As Long

Returns a single integer value indicating the type of medium from which MenuBox is running. Possible values are 0 (medium type cannot be determined), 1 (error determining medium type), 2 (removable media, e.g. floppy disk or removable hard disk), 3 (non-removable disk), 4 (network drive), 5 (CD or DVD) and 6 (RAM disk).

Property **MenuBoxVersion** As Long

Returns a single integer value consisting of the major program version number multiplied by 100, plus the minor version number (which is always in the range from 0 to 99). For example, version "2.0" (the first version implementing the MenuBox Extended DOM) is returned as 200. Version 12.34 would return a value of 1234, etc.

Property **Title** As String

Gets or sets the title of the MenuBox browser container window.

Playing Safe: Object Detection and Fallback

The functionality provided by the MenuBox Extended DOM is only available on Windows 98 or higher, or if Internet Explorer 4 or higher was installed on older systems. Invoking functions such as window.external.close() from Windows 95 or Windows NT 4.0 with Internet Explorer 3 will result in a script error (earlier versions of Internet Explorer were neither widespread nor did they support scripting). This can be avoided either by using object detection code as shown below, or by using the MenuBoxBrowser key in the [ApplicationCheck] section to verify that Advanced functionality is supported.

MenuBox includes a redistributable JavaScript (also referred to as ECMAScript and JScript) file named menubox.js, which defines a set of intermediate JavaScript functions which invoke the corresponding MenuBox functions only if the MenuBox Extended DOM is available. The menubox.js JavaScript file should be referenced by means of a SCRIPT element inside the HEAD element of an HTML document.

```
<head>
...
<script language="JavaScript" type="text/javascript" src="menubox.js"></script>
</head>
```

The intermediate JavaScript functions, such as menubox_close(), can safely be used instead of window.external.close(), regardless of operating system and browser version. In the worst case, i.e. if the MenuBox Extended DOM is unavailable, the script invokes similar functionality, which however usually requires user confirmation before a window is closed or a file is executed or saved (e.g. "The Web page you are viewing is trying to close the window. Do you want to close this window?" and "Would you like to open the file or save it to your computer?").

The JavaScript code has been tested not only in MenuBox (on systems with Internet Explorer 3 and higher), but also with different versions of Internet Explorer and other web browsers. Some versions of the Microsoft FrontPage Preview mode browser have a known problem with object detection code, and may display a script error message (which does not otherwise affect document editing or preview).

Security Considerations

The MenuBox Extended DOM allows HTML code to perform certain actions that are normally reserved to binary applications (such as, for example, non-HTML AutoRun tools). Whereas this does not introduce any new security implications when running in a local context (e.g. CD or DVD AutoRun), this is normally considered "dangerous" when running in an untrusted internet context (e.g. MenuBox used to operate a kiosk with general internet access).

The MenuBox Extended DOM inherits and is bound by the security privileges of the MenuBox application itself. More specifically, because the MenuBox redistributable runtime code is digitally signed with Microsoft Authenticode technology, MenuBox is bound by the policies which apply to signed applications.

When a user and/or an administrative policy allow the MenuBox executable to run (from a CD or DVD, or after software installation, etc.), that implies that MenuBox has been trusted and authorized to act as a menu-like front-end for opening certain known documents and programs. This is perfectly fine, and would apply in exactly the same way to any installed or AutoRun-launched application. Because normal web browsers were designed to protect the user from executing unknown and potentially malicious code in an unknown internet environment, rather than being deployed in a controlled distribution, much of the "trusted" functionality is normally not accessible to Dynamic HTML content, which is not even allowed to close the browser window without a warning message being displayed to the user (not to mention running an executable file). When operating in this context, the MenuBox Extended DOM provides a useful extension to the functionality which is normally accessible to HTML code, allowing the browser container to operate like a binary application, without introducing new security risks compared to other applications.

The GetRegistry method of the MenuBox Extended DOM gives read-only access to registry keys and values, with functionality being limited to checking whether a key exists and to read an existing value. In consideration of possible security concerns, MenuBox offers no functionality to write to an arbitrary registry location.

The GetNV and SetNV methods of the MenuBox Extended DOM provide read/write access to

nonvolatile variables. This information is stored in the current user's registry, inside a publisher-specific subkey (a publisher is a MenuBox licensee with a unique software license key) stored inside a MenuBox-specific key (effectively, a sandbox). In consideration of possible security concerns, the name of the variables is both normalized ("""", "/", "\" and ":" are converted to underscore characters) and truncated after 254 characters to help prevent potential buffer overrun, key traversal and other exploits. A maximum size of 2048 bytes is enforced for all values. As is the case with public Windows registry key, any application that can read or write to the system registry can access or modify these variables, which are simple and accessible by design.

The ReadFile and WriteFile methods of the MenuBox Extended DOM provide read/write access to private files stored inside a publisher-specific data directory (a publisher is a MenuBox licensee with a unique software license key) stored inside a MenuBox-specific per-user (roaming) data storage directory. The ReadFile and WriteFile methods do not allow access outside of this sandbox. In consideration of possible security concerns, the name of the file names is both normalized ("""", "/", "\" and ":" are converted to underscore characters) and truncated after 254 characters to help prevent potential buffer overrun, directory traversal and other exploits.

If MenuBox is used to operate a kiosk-like environment with general access to untrusted internet sites, then it would in theory be possible, for example, for a malicious person to access the kiosk and deliberately open an internet page containing malicious code, and execute that code via the MenuBox Execute function. On newer versions of Windows (Windows XP SP1 and higher) this attempted execution of code originating from the internet would in turn trigger the display of a warning message. When operating in such an untrusted context it is in general recommended to disable such functionality by setting the MenuBox NoExecute key. It would in theory also be possible to use the SetNV function to write a large number of nonvolatile variables, with the potential to fill up the registry. This can be prevented by setting the NoRegistry key, which at the same time also blocks all registry access, including the read-only access provided by the GetRegistry method. Similarly, a wasteful use of file writes can be prevented by setting the NoFiles key.

Related Topics

- For more information about paths and directories, see Paths and Current Directory.
- For an introduction to using templates to build suitable configuration files, see The MenuBox Wizard.
- For practical examples, see Web Resources.

3.7 Paths and Current Directory

Overview

After MenuBox has been launched from its own executable file ("menubox.exe", which may be renamed, if so desired) it needs to access one or more additional files (e.g. configuration files, documents to be displayed and executable files to be launched). Documents opened by MenuBox may in turn also contain links to other files. The following sections explain how to make sure that all files and other links references directly or indirectly by MenuBox will work flawlessly in a variety of configurations.

Relative vs. Absolute Paths

Paths would not be necessary if all files were located in the same directory, and if that directory were the current directory. When directories (or different drives, or computers, locally or on a network) are used to group content, paths indicate how to find files located in different locations. MenuBox supports different types of path formats in its command line and configuration file options, including:

Relative paths

e.g. "Directory\File", "Directory\Directory2\File", etc.

Absolute paths

e.g. "DriveLetter:\Directory\File, "DriveLetter:\Directory\Directory2\File", etc.

Paths containing environment variables

e.g. "%temp%\Directory\File, "%temp%\Directory\Directory2\File", etc.

UNC paths

e.g. "\\Server\ShareOrDrive\Directory\File", "\\Server\ShareOrDrive\Directory\Directory2File", etc.

Internet absolute paths

e.g. "https://Server/Directory/File", "ftp://Server/Directory/Directory2/File", etc.

Internet relative paths

e.g. "Directory/File", "Directory/Directory2/File", etc.

Paths within HTML scripts

e.g. "javascript:menubox_execute('Directory\\setup.exe')"

Absolute paths always begin with an absolute reference to a drive, device or server. While absolute paths may be useful when pointing to documents on the internet, we recommend to only use relative paths when referencing files on removable media such as CDs and DVDs. As different computer configurations use different drive letters (e.g. "D:\", "E:\", etc. for CD/DVD drives), absolute paths which include drive letters are guaranteed to fail as soon as the medium containing the application is used on a computer where the letters are assigned differently than on the test machine.

Relative paths are relative to the current directory, and may begin with one or more optional ".." parts, each meaning "up one level". For example, if MenuBox is running from inside the "AutoRun\Software" directory on a CD, and you need to reference an HTML file named "index.html" located in "Docs\HTML", you would reference that file as "..\..\Docs\HTML\index.html".

Keep in mind that space characters are not allowed within paths referenced in "autorun.inf".

A few rare and usually older applications do not support relative paths in their command line arguments. If you need to use such a program, for example a specific document viewer, either place the viewer and the document in the same directory as the configuration file ("menubox.ini"), or use the /d and/or /a command line options or the corresponding Directory and AbsolutePath configuration file keys to set the appropriate current directory and/or let MenuBox convert the relative path to an absolute path.

When using MenuBox to open Microsoft Office documents (e.g. PowerPoint, Excel, Access, etc.) which in turn link to other files which are also distributed with the linking document, be sure to enable the "Use relative path for hyperlink" option in the Office software before saving the document. For maximum compatibility with various document viewers it is recommended that you group all such linked files in a single directory, and that you set that directory to be the current directory with the appropriate MenuBox option (see below). You don't need to worry about this if the document is self-contained and does not contain references to other files.

If in doubt, remember to test all links of your final project, and then test again from a different drive (i.e. one where your project will be mounted under a different drive letter), preferably on a different computer (to also make sure that the links are not referencing some file which only exists on the hard disk where the project was created, e.g. you don't want to test a CD-ROM master on drives "D:\" and "E:\", only to later find out that the links were working only because they contained absolute references to something on your own "C:\" hard disk partition).

Within an HTML scripting environment, remember to use the applicable escape sequences for any special characters which may be contained in the File argument (e.g. "\" should become "\\").

Environment Variables

MenuBox automatically expands environment variables (e.g. "%name%") in paths to the full absolute path before further processing. This occurs at all levels (command line parameters, configuration file options, Windowless mode, and Extended DOM functionality). This allows a MenuBox application to locate and reference, for example, an installed application.

For maximum compatibility across different versions of Windows, and in consideration of the fact that paths stored in environment variables do not consistently end with or without a backslash character, multiple consecutive backslash characters that may be present in the final (expanded)

path are automatically converted to a single backslash character.

Current Directory

Relative paths and files referenced directly (without path), are always relative to the current directory. The following cases may apply depending on the application.

- The default current directory for command line mode is inherited from the context in which the MenuBox executable file is launched. For AutoRun applications this is the root of the medium. The /d command line option can be used to explicitly set the current directory.
- As soon as a configuration file (e.g. "menubox.ini") has been opened, the directory containing that file becomes the current directory. All files referenced from a configuration file are relative to this directory (unless a different directory is explicitly set via the Directory key).
- If MenuBox does not find a configuration file, it searches to see if an "autorun.inf" file exists, containing a [MenuBox] section with a Directory key. If that key is present, the current directory is set to the specified path. MenuBox then proceeds to open the configuration file in that new directory.
- Once an HTML file has been opened, all relative links inside the HTML document are relative to the location of the HTML file containing the link, regardless of the current directory (this behavior is implemented by web browsers, based on HTML specifications).
- Relative links contained in types of documents other than HTML (e.g. PowerPoint) are usually relative to the current directory, so care should be taken to set the appropriate directory to be the current directory (using the /d command line option or the Directory key of the configuration file) if necessary when opening such a document.
- If the MenuBox Execute() method is used from inside an HTML document (HTML window mode), that always operates relative to the directory containing the first HTML file (i.e. the file referenced by the URL key of the [HTMLWindow] section). This is because the MenuBox document object model is initialized when this first file is opened, and it does not reflect the directory of subsequent HTML files which may be opened during the browser session.

The default location of the configuration file, if not explicitly set using the /m command line option or in the "autorun.inf" file, is the directory containing the MenuBox executable file. This means that in most cases, as far as configuration files are concerned, the current directory will be the directory containing the MenuBox executable file (e.g. "menubox.exe").

If you understand how the current directory works, and if you group files properly with respect to each other in your distribution, you normally won't need to explicitly set or change directories. If however your needs are slightly more demanding, remember that the Directory and AbsolutePath keys, as well as the equivalent command line options, were designed to help you cover even the most exotic path and compatibility issues.

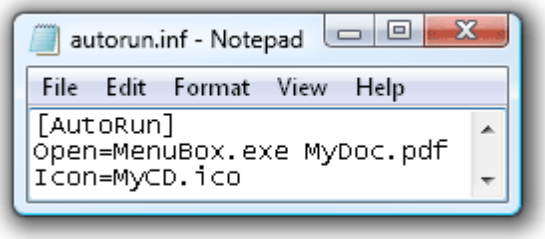
Related Topics

- For more information about using directories on AutoRun-enabled media, see AutoRun CDs and DVDs.

3.8 AutoRun CDs and DVDs

Overview

AutoRun is a feature of the Microsoft Windows operating system which makes it possible to automatically run a program (e.g. a menu window, a setup procedure, etc.) when a medium is inserted in the drive. Originally designed to launch CD-based applications, AutoRun also works on DVDs and other media. Support for AutoRun is a requirement of "Windows Logo" programs since Windows 95. MenuBox can not only be used as the application which opens a menu when the medium is inserted in the drive, but it also extends the original AutoRun specification (which covers the running of programs, not the opening of specific documents) by allowing non-program files such as HTML, PDF, PowerPoint, etc. to be automatically opened when the medium is inserted in the drive.



Autorun.inf

When a medium is inserted in an AutoRun-enabled drive, the system looks for a file named "autorun.inf", which has to be stored at the root of the medium (i.e. it cannot be inside a directory). This is a Windows feature unrelated to MenuBox, although a reference to the MenuBox directory can be placed inside this file. The file is structured following a simplified INI file format (the same format used by MenuBox for its own configuration files). Most AutoRun-related needs are fully covered by setting just one or two keys (Open and Icon) in the [AutoRun] section of the "autorun.inf" file, which can be edited with a simple program like Notepad.

The Open and Icon keys work on all versions of Windows (Windows 95 and higher). The leading path information and the trailing command line options in the Open key are optional. If you use the Icon key we recommend that, for best results, you make sure that the icon file contains images in at least the most popular formats (ready for use on different display resolutions and icon size settings), which are 16x16, 32x32 and 48x48 pixels. A larger additional 256x256 format was introduced in Windows Vista. If you also add the 256x256 format, be sure to test the result on your intended target operating system versions.

The AutoRun specification covers additional features (e.g. support for different CPU platforms, drive label and shortcut menus, handling of removable devices, installation of drivers, event handlers, multimedia content directories, etc.) which in part require a specific minimum version of Windows (e.g. Windows XP, Windows for Automotive, etc.), and which are covered by Microsoft's online documentation. Also, the default behavior of AutoRun (i.e. whether it should be enabled in general and/or on specific drives) can be modified via the Control Panel, programmatically, or via the registry. Links to additional information on these topics are included below.

[MenuBox]

Because "autorun.inf" was designed from the beginning to be open to future extensions, it is possible to add MenuBox-specific information to this file, which helps keeping the root of the medium free of unnecessary items. Rather than keeping the MenuBox executable and its INI file in the same directory, the two can be separated. This can further be combined with the fact that "MenuBox.exe" may be renamed to whatever is more intuitive from a usability point of view (e.g. "AutoRun.exe"). If the MenuBox executable does not find a matching INI file (e.g. "menubox.exe" looks for "menubox.ini", "autorun.exe" looks for "autorun.ini", etc.) in the same directory, it looks for a different directory reference in the "autorun.inf" file itself, or in "autorun.txt", in a key named Directory in the [MenuBox] section. If this information is found, MenuBox acts as if it were launched from that directory.

Some antivirus applications may block access to "autorun.inf". Also, some newer versions of Windows do not automatically run the AutoRun action on all types of media (e.g. on USB flash drives). In consideration of this it may be helpful to write a human-readable comment (lines introduced by ";") in a text file named "autorun.txt", followed by the [MenuBox] section with the Directory key.

Examples

In the following example "menubox.exe" (the MenuBox executable) and "menubox.ini" (the MenuBox configuration file) are stored at the root of the medium (i.e. together with "autorun.inf") and the behavior of MenuBox is defined in detail in the MenuBox configuration file (which is opened automatically when MenuBox is launched with no command line options).

```
[AutoRun]
Open=MenuBox.exe
```

The following example additionally assigns an icon to the medium and groups the MenuBox executable and configuration file, plus the icon file, in a subdirectory named "MenuBox" (this could be any other name, e.g. "AutoRun", etc.)

```
[AutoRun]
Open=MenuBox\MenuBox.exe
Icon=MenuBox\MyCD.ico
```

In the following example, "MenuBox.exe" is renamed "AutoRun.exe", and a reference to the MenuBox directory (containing the icon file and all MenuBox files except for the executable) is added to the "autorun.inf" file itself. This results in a cleaner and user-friendly root, where "AutoRun.exe" can be started manually by the user even if AutoRun is disabled.

```
[AutoRun]
Open=AutoRun.exe
Icon=MenuBox\MyCD.ico
[MenuBox]
Directory=MenuBox
```

The following example is a variation of the previous one, where "MenuBox.exe" is still renamed "AutoRun.exe", however the reference to the MenuBox directory is added to an "autorun.txt" file. This helps survive AutoRun being disabled or "autorun.inf" being blocked by an antivirus application, and provides further information to the user.

```
; To access the content of this medium select AutoRun.exe.
;
; This project was created with MenuBox (www.menubox.com).

[MenuBox]
Directory=MenuBox
```

The following example uses the MenuBox command line functionality, rather than a MenuBox configuration file, to open an HTML file using the default browser installed on the system. Command line mode is very easy to use, but it cannot be used to check if a viewer is installed and install it if necessary. (Note: although this example uses an external browser, HTML documents can also be opened inside MenuBox.)

```
[AutoRun]
Open=AutoRun\MenuBox.exe Docs\index.html
```

The following example uses the MenuBox command line functionality to open a PowerPoint presentation, making sure that the current directory is changed first, so that relative links from the first PowerPoint file to other files in the same directory will work properly (be sure to also enable the "Use relative path for hyperlink" option in PowerPoint). The capitalization has been modified with respect to the previous examples to show how section and key names are not case sensitive.

```
[autorun]
OPEN=menubox\menubox.exe /d:powerpoint presentation.pps
```

The above example can be applied in a similar way to other types of documents (e.g. PDF, Excel, Word, etc.), but in all cases it does not check whether the system is actually capable of displaying these documents. MenuBox includes powerful functionality to check whether an appropriate viewer/player is installed, and, if not, to install the required software (which can be included on your medium) before proceeding. This functionality cannot be expressed in a single command line, and is therefore available through the configuration file.

Related Topics

- For more information about paths and directories, see Paths and Current Directory.
- For an introduction to using templates to build AutoRun-enabled projects, see The MenuBox Wizard.
- For more detailed reference information about the AutoRun specification, see Web Resources.

3.9 Redistributable Files

Overview

When you acquire MenuBox you are licensed to redistribute the MenuBox executable file with your projects. Within this context you may also reuse the script code (i.e. the menubox.js file) and background graphics (as provided in the text window sample files). Don't forget to sign the project before redistributing it, which will enable the full software functionality without the "Free Version" banner. Please understand that you may not redistribute a MenuBox license key which was not registered either in your name (if you are the developer or data preparer of the project) or in the name of the publisher of the project.

Location of Files

If you installed MenuBox to the default hard disk location on an English language version of Windows you should find the redistributable files in "C:/Program Files/Cloanto/MenuBox/Redistributable".

When redistributing the MenuBox files we recommend that you create a dedicated directory (named as you prefer, e.g. "MenuBox", "AutoRun", etc.), so as to group all related files (which may include one or more configuration files, e.g. "menubox.ini", HTML files, e.g. "index.html", and additional files such as graphics and sound files). By doing so you will keep the file organization as simple as possible, hiding technical content which does not need to be manually accessed by your users, while also avoiding possible functional ambiguities related to linked items and the current directory. The only file which, if used, cannot be placed in a subdirectory, but must always be stored at the root of the distribution medium, is the AutoRun file ("autorun.inf").

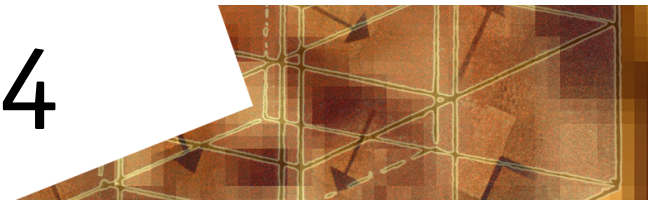
Native 64-Bit Executable

The default MenuBox.exe executable file was designed to work on both 32-bit and 64-bit versions of Windows. For custom deployments that require a specific environment such as the 64-bit version of Windows PE (which does not include the WOW64 subsystem), a pure 64-bit version of MenuBox is included in the set of redistributable files. If you need to use this version (which will not run on 32-bit versions of Windows), take the MenuBox64.exe file and rename it to suit your needs, replacing the default MenuBox.exe file.

Related Topics

- For more information about signing a project, see The Sign Project Tool.
- For information about renaming the executable and configuration files, see Command Line Options.
- For information about using multiple localized configuration files, see Configuration File Options.
- For more information about paths and directories, see Paths and Current Directory.
- For more information about AutoRun-enabled media, see AutoRun CDs and DVDs.
- For more information about registering the software, see Registering MenuBox.
- For an introduction to using templates to build sets of redistributable files, see The MenuBox Wizard.

Part 4



4 Additional Resources

This section covers:

- Web Resources
- ISO 639-1 Language Codes
- Windows Character Set Codes

4.1 Web Resources

You may find the following resources on the MenuBox website of interest:

- MenuBox Homepage
- Frequently Asked Questions
- Tutorials and Examples
- Web Links

4.2 ISO 639-1 Language Codes

MenuBox supports the following two-letter language codes as per ISO 639-1 standard. This specification always uses only two letters, e.g. Australian English, British English, Canadian English and US English are all referenced by "en" (rather than "en-au", etc.) The same codes are used in the file names of localized configurations and by the Language property of the MenuBox Extended DOM.

Language	ISO 639-1 Code
Afrikaans	af
Albanian	sq
Arabic	ar
Armenian	hy
Azeri	az
Basque	eu
Belarusian	be
Bulgarian	bg
Catalan	ca
Chinese	zh
Croatian	hr
Czech	cs
Danish	da
Dutch	nl
English	en
Estonian	et
Faroese	fo
Farsi	fa
Finnish	fi
French	fr
Galician	gl

Language	ISO 639-1 Code
Georgian	ka
German	de
Greek	el
Gujarati	gu
Hebrew	he
Hindi	hi
Hungarian	hu
Icelandic	is
Indonesian	id
Italian	it
Japanese	ja
Kannada	kn
Kazakh	kk
Korean	ko
Kyrgyz	ky
Latvian	lv
Lithuanian	lt
Macedonian	mk
Malay	ms
Marathi	mr
Mongolian	mn
Norwegian Bokmål	nb
Norwegian Nynorsk	nn
Polish	pl
Portuguese	pt
Punjabi	pa
Romanian	ro
Russian	ru
Sanskrit	sa
Serbian	sr
Slovak	sk
Slovenian	sl
Spanish	es
Swahili	sw
Swedish	sv
Tamil	ta
Tatar	tt
Telugu	te
Thai	th

Language	ISO 639-1 Code
Turkish	tr
Ukrainian	uk
Urdu	ur
Uzbek	uz
Vietnamese	vi

4.3 Windows Character Set Codes

MenuBox supports the following Windows character set codes for the display of text window mode texts. If no character set is explicitly set by means of the CharacterSet key, the "ANSI" character set (Windows 1252, a variant of ISO-8859-1) is used.

Character Set	Windows Code
ANSI (Western European)	0
Arabic	178
Baltic	186
Chinese (Big-5)	136
Default	1
Eastern European	238
GB-2312	134
Greek	161
Hangul	129
Hebrew	177
Johab	130
Mac	77
OEM	255
Russian	204
Shift JIS	128
Symbol	2
Thai	222
Turkish	162
Vietnamese	163

The Johab character set is included with the Korean language edition of Windows. The Arabic and Hebrew character sets are included with the Middle East language edition of Windows. The Thai character set is included with the Thai language edition of Windows. The OEM value specifies a character set that is operating-system dependent. On Windows 95, 98 and Me the Default value indicates that the name and size of a font fully describe the logical font. On Windows NT, 2000, XP and Server 2003 the Default value indicates that the character set is set to a value based on the current system locale. For example, when the system locale is English (United States), it is set as ANSI.