Just Great Software

DeployMaster

Version 7.3.0 — 4 January 2024

Published by Just Great Software Co. Ltd.

Table of Contents

Part 1

Introduction to DeployMaster

1. Welcome to DeployMaster Help

Thank you for choosing DeployMaster! You will soon find out that DeployMaster will allow you to easily create an installation package for your software, which will be nice and convenient for your users and will help your software to make a good first impression. It gives advanced users the control they want, but does not impose that responsibility upon novice users. Before reading on, you may want to be aware of the conventions used in this help file. If you have any comments or questions about a particular topic in the help file, select Help|Forum in the menu in DeployMaster to discuss the issue in the DeployMaster user forum.

DeployMaster can create installers that are compatible with Windows 98, ME, NT4, 2000, XP, Vista, 7, and 8 as well as Windows Server NT4, 2000, 2003, 2008, 2008 R2, and 2012. The installers can install 32-bit software on all 32-bit and 64-bit versions of Windows, and install 64-bit software on all 64-bit versions of Windows. You can target all these platforms with a single setup. The setup package DeployMaster builds for you will automatically handle the differences between these operating systems, such as different system folders. To build setups with DeployMaster, you need Windows 2000, XP, Vista, 7, or 8, 32-bit or 64-bit.

Conventions Used in the DeployMaster Documentation

- Whenever there was a choice between masculine and feminine pronouns, one gender was chosen at random. DeployMaster has been designed to be equally useful for both men and women. We hope that anybody needing an installation tool, men and women, will find DeployMaster to be a very attractive choice.
- "You" refers to the person who will be using DeployMaster to distribute her software or her company's software. Since you are reading this text, which is the documentation for DeployMaster Builder, you are probably such a person.
- "The user" is the person who will (attempt to) deploy that software, packaged by you with DeployMaster, on his system. So he is either your customer or working for your customer. Making him happy will benefit your cash flow.
- The phrase "installation/setup/deployment program" is the software that the user will interact with. This is the package that you will distribute to the users. The generic name for this file is `setup.exe`.
- "DeployMaster" means either the DeployMaster Builder program that you will use, or the DeployMaster setup program the user will use, or both.

2. Why Novice Users Like DeployMaster

Users that are relatively new about computers or only put up with them because they have no other choice, will appreciate it if you deliver your software through DeployMaster

Most other installation tools bother everyone with questions like which folder they want to install to, which program group to create, etc. "What do I care?", thinks Joe User, "I just want to start using this software. Can't my computer figure out what it needs to do?" Joe User does not care about files or folders. He just wants to get his job done. Installing software becomes a test on how fast he can repeatedly click on a button labeled "Next".

While it is acceptable to bother people with questions when they are applying for a gun permit, software should be able to figure out all by itself where it is reasonable to install itself. Joe User only wants to see an icon pop up on his desktop that he can click on to run your software. People who buy a computer expect it to help them instead of the other way around. With DeployMaster, Joe User only has to click twice after launching the installation program. He'll click the default "Immediate Instalation" button, indicating he has no interest in helping the computer keeping its act together. If you supplied a license agreement, he'll need another click on the "I accept" button. DeployMaster will do the rest.

After Joe User has gained some more experience with computers, he may wish to take an intermediate route. He still does not want to be asked any technical questions, but he does want to select which parts of the application he wants installed, and which not. He can do this by clicking the "Custom Installation" button. This will show up a flat list of components. Clicking on a component will show a detailed description of what purpose it serves, so he can make an educated choice. DeployMaster will automatically keep track of requirements between components, without displaying error messages that humiliate Joe or allowing him to break the integrity of the installation. Components that are required by other components, but only serve a purpose when those components are installed, are not presented to Mr. User as they make no sense to him. If it is not an option, it is not presented.

It gets even better. The first time around, Joe User will probably just select the fast and painless install, not changing the component selection. Later on, after having spent some time with your software, he feels a need to install an optional component that's not installed by default. Or, if your package installs everything by default, he may want to uninstall a piece of your software he does not use to free up some disk space (he will need some when you release your next killer app). Traditional installation tools only offer the nuke'n'resurrect option: uninstall everything and then reinstall those pieces you need, probably with several reboots in-between. Not so with DeployMaster! Running the installation program a second time, without first uninstalling, will present the same Custom Installation button and it will then update the installation. It will remove what is no longer desired and add what has been newly selected.

In sharp contrast with Joe User's need, John Geek wants full control of what happens to his computer. Learn why advanced users like DeployMaster.

3. Why Advanced Users Like DeployMaster

Advanced computer users often really dislike installation software. The main reason is that the software messes with the system behind the user's back. The user is dissatisfied because he loses control and garbage is piling up in the \Windows and \Windows\System folders.

The first thing John Geek will do when installing a DeployMaster-delivered application, is click the "Advanced Installation" button.

First he can select the folders the software should be installed in, either by typing them in directly or by selecting it from a tree view, accessible through the ellipsis (...) buttons. DeployMaster will not be so rude as to ask if the folder should be created, if it does not yet exist. John Geek knows what he wants: a new folder for a new application.

Next John Geek will see a list of file associations the setup program is told to register by the developer (you) and he gets the chance to disable any or all of them. Chances are that he already has associated one or more file types with his favorite software, and does not want to change those. (At least not until he finds out that your software is much better.) And if he does allow file associations to be changed, he will be glad that upon uninstallation, the original associations will be automatically restored.

Then the component selection screen appears, but it is slightly different from the one Joe User sees. First, components that are not user-selectable are also listed. DeployMaster will still pay attention to the fact that some components require others, and enforce those selections, but John Geek will see (and understand) what is going on. Also, at the bottom a checkbox now appears. When checked, the component list will be expanded into a tree showing each and every file that will be installed. So if a particular piece of software thinks it can use his \Windows folder as a trash can, John Geek will be warned and probably hit the Abort button before any modifications have been made to his system. DeployMaster allows him to confirm that your application behaves properly and that there is no risk installing it.

When the installation is finished, DeployMaster saves a deployment log in the folder where the application was installed. Any installation tool with an uninstall facility does this. However, with DeployMaster, the installation log is a plain text file that John Geek can inspect, even a long time after the installation. And its format is documented.

4. Share Experiences and Get Help on The User Forums

When you click the Login button you will be asked for a name and email address. The name you enter is what others see when you post a message to the forum. It is polite to enter your real, full name. The forums are private, friendly and spam-free, so there's no need to hide behind a pseudonym. While you can use an anonymous handle, you'll find that people (other DeployMaster users) are more willing to help you if you let them know who you are. Support staff from Just Great Software will answer technical support questions anyhow.

The email address you enter is used to email you whenever others participate in one of your discussions. The email address is never displayed to anyone, and is never used for anything other than the automatic notifications. DeployMaster's forum system does not have a function to respond privately to a message. If you don't want to receive automatic email notifications, there's no need to enter an email address.

If you select "never email replies", you'll never get any email. If you select "email replies to conversations you start", you'll get an email whenever somebody replies to a conversation that you started. If you select "email replies to conversations that you participate in", you'll get an email whenever somebody replies to a conversation that you started or replied to. The From address on the email notifications is forums@jgsoft.com. You can filter the messages based on this address in your email software.

DeployMaster's forum system uses the standard HTTP protocol which is also used for regular web browsing. If your computer is behind an HTTP proxy, click the Proxy button to configure the proxy connection.

If you prefer to be notified of new messages via an RSS feed instead of email, log in first. After DeployMaster has connected to the forums, you can click the Feeds button to select RSS feeds that you can add to your favorite feed reader.

Various Forums

Below the Login button, there is a list where you can select which particular forum you want to participate in. The "DeployMaster" forum is for discussing anything related to the DeployMaster software itself. This is the place for technical support questions, feature requests and other feedback regarding the functionality and use of DeployMaster.

The "regular expressions" forum is for discussing regular expressions in general. Here you can talk about creating regular expressions for particular tasks, and exchange ideas on how to implement regular expressions with whatever application or programming language you work with.

Searching The Forums

Before starting a new conversation, please check first if there's already a conversation going on about your topic. If you find one, start with reading all the messages in that conversation. If you have any further comments or questions on that conversation, reply to the existing conversation instead of starting a new one. That way, the thread of the conversation stays together, and others can instantly see what you're talking about. It doesn't matter whether the conversation is many years old. If you reply to it, it becomes an active conversation that moves to the top automatically.

In the top right corner of the Forum panel, there is a box on the toolbar that you can use to search for messages. When you enter something into that box, only conversations that include at least one message containing the word or phrase you entered are shown.

The filtering happens in real time as you type in your word or phrase. It includes all conversation subjects, all message summaries, and all author names. Is also includes the message bodies of conversations that have been downloaded. DeployMaster automatically downloads the 20 most recent conversations when you connect to the forum. Other conversations are downloaded only if you click on them to view them. Downloaded messages are cached between DeployMaster sessions.

This means that the real time filtering does not search through the body text of older messages that you've never viewed. To search through those messages, click the Search Forum button that sits immediately to the right of the search box. DeployMaster then shows all conversations with messages containing the search term in their summaries, author names, or body texts, including messages that haven't been downloaded yet.

You can enter only one search term, which is searched for literally. If you enter "find me", only conversations containing the two words "find me" next to each other and in that order are shown. You cannot use boolean operators like "or" or "and". Since the filtering is instant, you can quickly try various keywords.

If your search term can be found in the subject of a conversation, then all messages in that conversation are always shown. If the search term cannot be found in the subject of a conversation, but it can be found in the summary or body text of a message in that conversation, then the Show Complete Conversations button in the far top right corner determines which messages are shown. When this button is up, only the messages in which the search term was found are shown for that conversation. When this button is down, all messages are shown for that conversation.

If you have previously participated in the forums, you can use the Show My Conversations button to show only conversations that you have participated in and conversations that have a message that you gave a +1. If you did not enter a search term, this shows all messages in all those conversations. If you did enter a search term, this reduces the search results to conversations that you participated in.

Conversations and Messages

The left hand half of the Forum pane shows two lists. The one at the top shows conversations. The bottom one shows the messages in the selected conversation. You can change the order of the conversations and messages by clicking on the column headers in the lists. A conversation talks about one specific topic. In other forums, a conversation is sometimes called a thread.

If you want to talk about a topic that doesn't have a conversation yet, click the New button to start a new conversation. A new entry appears in the list of conversations with an edit box. Type in a brief subject for your conversation (up to 100 characters) and press Enter. Please write a clear subject such as "scraping an HTML table in Perl" rather than "need help with HTML" or just "help". A clear subject significantly increases the odds that somebody who knows the answer will actually click on your conversation, read your question and reply. A generic scream for help only gives the impression you're too lazy to type in a clear subject, and most forum users don't like helping lazy people.

After typing in your subject and pressing Enter, the keyboard focus moves to the empty box where you can enter the body text of your message. Please try to be as clear and descriptive as you can. The more information you provide, the more likely you'll get a timely and accurate answer. If your question is about a particular regular expression, don’t forget to attach your regular expression or test data. Use the forum’s attachment system rather than copying and pasting stuff into your message text.

If you want to reply to an existing conversation, select the conversation and click the Reply button. It doesn’t matter which message in the conversation you selected. Replies are always to the whole conversation rather than to a particular message in a conversation. DeployMaster doesn’t thread messages like newsgroup software tends to do. This prevents conversations from veering off-topic. If you want to respond to somebody and bring up a different subject, you can start a new conversation, and mention the new conversation in a short reply to the old one.

When starting a reply, a new entry appears in the list of messages. Type in a summary of your reply (up to 100 characters) and press Enter. Then you can type in the full text of your reply, just like when you start a new conversation. However, doing so is optional. If your reply is very brief, simply leave the message body blank. When you send a reply without any body text, the forum system uses the summary as the body text, and automatically prepends [nt] to your summary. The [nt] is an abbreviation for “no text”, meaning the summary is all there is. If you see [nt] on a reply, you don’t need to click on it to see the rest of the message. This way you can quickly respond with “Thank you” or “You’re welcome” and other brief courtesy messages that are often sadly absent from online communication.

When you’re done with your message, click the Send button to publish it. There’s no need to hurry clicking the Send button. DeployMaster forever keeps all your messages in progress, even if you close and restart DeployMaster, or refresh the forums. Sometimes it’s a good idea to sleep on a reply if the discussion gets a little heated. You can have as many draft conversations and replies as you want. You can read other messages while composing your reply. If you’re replying to a long question, you can switch between the message with the question and your reply while you’re writing.

Dates and Times

The Started column indicates how long ago each conversation was started. The Last Reply column indicates how long ago the last reply was made, if any. For older conversations, it indicates how much later that reply came after the conversation was started. If you sort conversations by last reply, conversations without replies are sorted by their starting date. The sort order is always based on absolute dates.

The Date Posted column indicates how long ago each message was posted. For replies to older conversations, this date indicates how much later that message was posted after the conversation was started. The Date Edited column indicates how long ago the message was edited, if at all. For older messages, it indicates how much later it was edited after it was posted. If you sort messages by the date they were edited, messages that weren’t edited are sorted by their posting date. The sort order is always based on absolute dates.

Directly Attach DeployMaster Setup Scripts and Other Files

One of the greatest benefits of DeployMaster’s built-in forums is that you can directly attach the DeployMaster setup script or any other files you’re working with. Simply click the Attach button and select the item you want to attach. If you select to attach your setup script, all the settings in your setup, including any unsaved changes, are attached to your message.

To attach a screen shot, press the Print Screen button on the keyboard to capture your whole desktop. Or, press Alt+Print Screen to just capture the active window (e.g. DeployMaster's window). Then switch to the Forum panel, click the Attach button, and select Clipboard. You can also attach text you copied to the clipboard this way.

It's best to add your attachments while you're still composing your message. The attachments appear with the message, but won't be uploaded until you click the Send button to post your message. If you add an attachment to a message you've written previously, it is uploaded immediately. If you send a message and later notice you forgot an attachment then you can attach it directly. You shouldn't click the Edit button unless you want to edit the body text of the message.

You cannot attach anything to messages written by others. Write your own reply, and attach your data to that.

To check out an attachment uploaded by somebody else, click the Use or Save button. The Use button loads the attachment directly into DeployMaster. If you load a DeployMaster setup script, it will replace all the settings in DeployMaster as would happen when you open a `.deploy` file. If you click the Save button, DeployMaster prompts for a location to save the attachment. DeployMaster does not automatically open attachments you save.

DeployMaster automatically compresses attachments in memory before uploading them. So if you want to attach an external file, there's no need to compress it using a zip program first. If you compress the file manually, everybody who wants to open it has to decompress it manually. If you let DeployMaster compress it automatically, decompression is also automatic.

Saying Thanks or Me Too and Starring Conversations

The +1 button lets you say "thanks" or "me too" for the message that you are presently reading. This is a quick way to show your appreciation or agreement without having to post a reply. Everybody can see how many people gave a +1 to a particular message. But nobody can see who gave those +1. You can only see whether one of those +1 came from you or not. The +1 column in the list of messages shows a number to indicate the total number of people (possibly including you) that gave a +1 to that message. The +1 column shows a + before the number if one of those +1 came from you.

The list of conversations also has a +1 column. The number in this column indicates the total number of different people that gave a +1 to one or more messages in the conversation. The number of +1 for the conversation will be less than the sum of the +1 of all messages in the conversation if one person gave a +1 to multiple messages in the conversation. The +1 column for conversations shows a + before the number if you gave a +1 to any of the messages in that conversation.

You can click the +1 column header to sort conversations or messages by their +1. Conversations or messages that you gave a +1 are placed above conversations or messages to which you did not give a +1. So you can also use the +1 feature to star or bookmark messages as sorting by +1 puts yours at the top. The conversations that you gave +1 are sorted among themselves by decreasing number of total +1. Below all those, the remaining conversations are sorted by their total +1.

Another way to save a conversation for later is to click the Reply button without clicking the Send button. Conversations with unsent replies are always sorted at the top. They are never hidden when filtering the forum. Nobody but you can see your unsent reply. Replies don't touch the server until you click the Send button. Unsent replies persist when you close and restart EditPad.

Taking Back Your Words

If you regret anything you wrote, simply delete it. There are three Delete buttons. The one above the list of conversations deletes the whole conversation. You can only delete a conversation if nobody else participated in it. The Delete button above the edit box for the message body deletes that message, if you wrote it. It is labeled Cancel if you have not yet sent your message. The Delete button above the list of attachments deletes the selected attachment, if it belongs to a message that you wrote.

If somebody already downloaded your message before you got around to deleting it, it won't vanish instantly. The message will disappear from their view of the forums the next time they log onto the forums or click Refresh. If you see messages disappear when you refresh your own view of the forums, that means the author of the message deleted it. If you replied to a conversation and the original question disappears, leaving your reply as the only message, you should delete your reply too. Otherwise, your reply will look silly all by itself. When you delete the last reply to a conversation, the conversation itself is also deleted, whether you started it or not.

Changing Your Opinion

If you think you could better phrase something you wrote earlier, select the message and then click the Edit button above the message text. You can then edit the subject and/or body text of the message. Click the Send button to publish the edited message. It will replace the original. If you change your mind about editing the message, click the Cancel button. Make sure to click it only once! When editing a message, the Delete button changes its caption to Cancel and when clicked reverts the message to what it was before you started editing it. If you click Delete a second time (i.e. while the message is no longer being edited), you'll delete the message from the forum.

If other people have already downloaded your message, their view of the message will magically change when they click Refresh or log in again. Since things may get confusing if people respond to your original message before they see the edited message, it's best to restrict your edits to minor errors like spelling mistakes. If you change your opinion, click the Reply button to add a new message to the same conversation.

Updating Your View

When you click the Login button, DeployMaster automatically downloads all new conversations and message summaries. Message bodies are downloaded one conversation at a time as you click on the conversations. Attachments are downloaded individually when you click the Use or Save button.

DeployMaster keeps a cache of conversations and messages that persists when you close DeployMaster. Attachments are cached while DeployMaster is running, and discarded when you close DeployMaster. By caching conversations and messages, DeployMaster improves the responsiveness of the forum while reducing the stress on the forum server.

If you keep DeployMaster running for a long time, DeployMaster does not automatically check for new conversations and messages. To do so, click the Refresh button.

Whenever you click Login or Refresh, all conversations and messages are marked as "read". They won't have any special indicator in the list of conversations or messages. If the refresh downloads new conversation and message summaries, those are marked as "unread". This is indicated with the same "people in the cloud" icon as shown next to the Login button.

Forum RSS Feeds

When you're connected to the user forum, you can click the Feeds button to select RSS feeds that you can add to your favorite feed reader. This way, you can follow DeployMaster's discussion forums as part of your regular reading, without having to start DeployMaster. To participate in the discussions, simply click on a link in the RSS feed. All links in DeployMaster's RSS feeds will start DeployMaster and present the forum login screen. After you log in, wait a few moments for DeployMaster to download the latest conversations. DeployMaster will automatically select the conversation or message that the link points to. If DeployMaster was already running and you were already logged onto the forums, the conversation or message that the link points to is selected immediately.

You can choose which conversations should be included in the RSS feed:

- All conversations in all groups: show all conversations in all the discussion groups that you can access in DeployMaster.
- All conversations in the selected group: show the list of conversations that DeployMaster is presently showing on the Forum panel.
- All conversations you participated in: show all conversations that you started or replied to in all the discussion groups that you can access in DeployMaster.
- All conversations you started: show all conversations that you started in all the discussion groups that you can access in DeployMaster.
- Only the selected conversation: show only the conversation that you are presently reading on the Forum panel in DeployMaster.

In addition, you can choose how the conversations that you want in your RSS feed should be arranged into items or entries in the feed:

- One item per group, with a list of conversations: Entries link to groups as a whole. The entry titles show the names of the groups. The text of each entry shows a list of conversation subjects and dates. You can click the subjects to participate in the conversation in DeployMaster. Choose this option if you prefer to read discussions in DeployMaster itself (with instant access to attachments), and you only want your RSS reader to tell you if there's anything new to be read.
- One item per conversation, without messages: Entries link to conversations. The entry titles show the subjects of the conversations. The entry text shows the date the conversation was started and last replied to. If your feed has conversations from multiple groups, those will be mixed among each other.
- One item per conversation, with a list of messages: Entries link to conversations. The entry titles show the subjects of the conversations. The entry text shows the list of replies with their summaries, author names, and dates. You can click a message summary to read the message in DeployMaster. If your feed has conversations from multiple groups, those will be mixed among each other.
- One item per conversation, with a list of messages: Entries link to conversations. The entry titles show the subjects of the conversations. The entry text shows the list of replies, each with their full text. If your feed has conversations from multiple groups, those will be mixed among each other. This is the best option if you want to read full discussions in your RSS reader.
- One item per message with its full text: Entries link to messages (responses to conversations). The entry titles show the message summary. The entry text shows the full text of the reply, and the

conversation subject that you can click on to open the conversation in DeployMaster. If your feed lists multiple conversations, replies to different conversations are mixed among each other. Choose this option if you want to read full discussions in your RSS reader, but your RSS reader does not mark old entries as unread when they are updated in the RSS feed.

Part 2

Creating Installers with DeployMaster

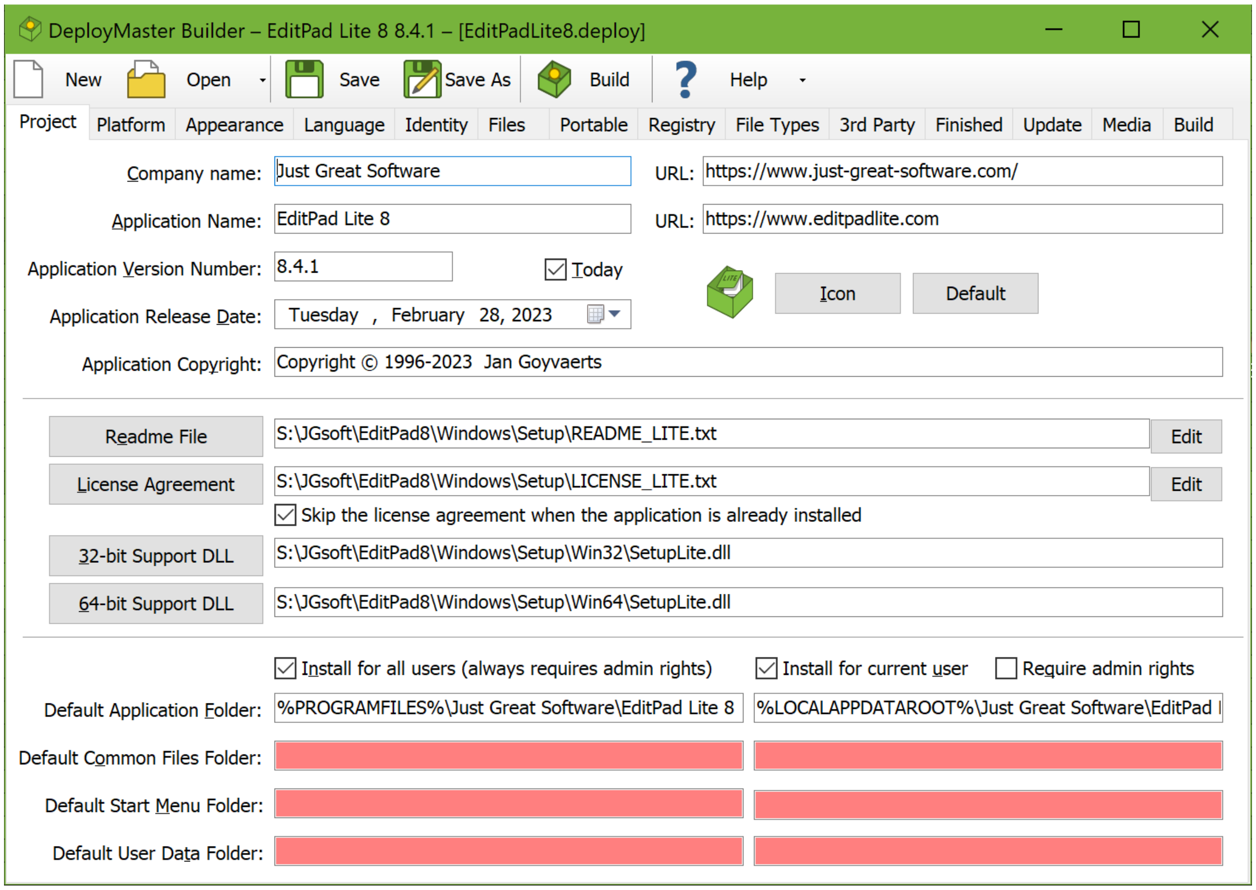
1. Deploying With DeployMaster

Fourteen steps are involved in putting your software into a deployment package using DeployMaster:

1. Supply general information about your software package on the Project page.
2. Select which versions of Windows your setup should support on the Platform page.
3. If you want to, you can create a nice background for your setup program on the Appearance page.
4. Localize the strings used by the setup program to the language of your software on the Language page.
5. You can have your setup program ask the user for identity information by selecting the options on the Identity page.
6. Select of which files your package consists, and arrange them into components on the Files page.
7. If your application can be run in a portable manner (i.e. without modifying the host computer) from a removable medium like a memory card or USB stick, configure DeployMaster's ability to install your application in a portable manner on the Portable page.
8. If any registry keys should be created during the installation, specify them on the Registry page.
9. Create any file associations that should be made, on the File Types page.
10. If your application relies on 3rd party libraries or applications, such as the Microsoft .NET framework, add those to your installer on the 3rd Party page.
11. Specify what should happen when the installation is complete on the Finished page.
12. When releasing a new version of an existing setup package, choose the update options on the Update page.
13. Determine the format, destination, and digital signatures of the generated installation package on the Media page.
14. Have DeployMaster generate that package on the Build page.

2. Project

On the Project page, you can specify general information about the application your setup package will install.



Company name: Just Great Software
URL: https://www.just-great-software.com/
Application Name: EditPad Lite 8
URL: https://www.editpadlite.com
Application Version Number: 8.4.1
☑ Today
Application Release Date: Tuesday , February 28, 2023
Application Copyright: Copyright © 1996-2023 Jan Goyvaerts
Readme File: S:\JGsoft\EditPad8\Windows\Setup\README_LITE.txt
License Agreement: S:\JGsoft\EditPad8\Windows\Setup\LICENSE_LITE.txt
☑ Skip the license agreement when the application is already installed
32-bit Support DLL: S:\JGsoft\EditPad8\Windows\Setup\Win32\SetupLite.dll
64-bit Support DLL: S:\JGsoft\EditPad8\Windows\Setup\Win64\SetupLite.dll
☑ Install for all users (always requires admin rights) ☑ Install for current user ☐ Require admin rights
Default Application Folder: %PROGRAMFILES%\Just Great Software\EditPad Lite 8 | %LOCALAPPDATAROOT%\Just Great Software\EditPad
Default Common Files Folder:
Default Start Menu Folder:
Default User Data Folder:

The information in the top part of the Project page will be shown to the user when the installation program is executed. Filling out these fields is the first thing you should do when creating a new DeployMaster package. When you have specified the name of your company and the name of the application, you will see that several other fields automatically receive a value (which you can, of course, change if you want to).

Company: The name of the company that produced the application or the brand name it is sold under. Type in your own name if this is not a company project. Leaving this field blank is permitted, but not recommended.

Company URL: URL of the web site of the company or brand specified earlier. If a URL is specified, the company name shown in the setup program will become clickable. If clicked, the user's default browser will be launched to visit the company web site.

Application name: The full name of the application the setup package will install. If you are not packaging an application but a collection of documents, enter a general title for these documents. Naming your project is **mandatory**. Each project must have a unique name. This name is what DeployMaster uses to determine whether the application that is about to be installed is already installed or not. If a previous install with the same application name already exists, DeployMaster will replace that installation with the new install. If you want two (major) versions of your product to co-exist side by side, you have to change the application name between releases, e.g. by indicating the (major) version number.

Application URL: Should point to the web page containing more information about this specific product. If specified, the application name will become clickable.

Application version number: Version number of the application, usually something in the form of 1.0.0. Leave blank if you do not use a meaningful version numbering scheme. The version number is displayed to the user, but is not used by the setup application itself. You can type in anything you like and that your users will understand. On the Appearance and Media pages, you can use the %VERSION% and %VERSION09% placeholders to display the version number or to include it in the file name of your installer. The section explaining the Media page has more details.

Application release date: It is **very important** to specify an accurate release date. It should increase whenever the version number increases, and be identical when the version number is identical. The setup program uses this information to find out—in case another copy of the application has already been installed—whether the user is trying to install the same, an older or a newer version. On the Appearance and Media pages, you can use the %DATEN%, %DATE09%, and %YYYYMMDD% placeholders to display the release date or to include it in the file name of your installer. The section explaining the Media page has more details.

If you tick the **Today** checkbox then DeployMaster automatically sets the release date to today's date whenever you build your installer. When today's date is in a different year than the previous release date and the copyright string contains the year number of the old release date then that number is replaced with the number of the present year.

Icon: Icon that identifies the installer. The generated setup.exe will be identified by this icon when it is listed in Windows Explorer or other file managers. You can only supply an .ico file. If you want to use the icon of another .exe file or one stored in a .dll, you'll need to use a resource manager or icon editor to extract the icon into an .ico file first. If you don't supply an icon, the installer will use DeployMaster's icon. Using DeployMaster's icon is perfectly fine. It identifiers your setup.exe as a user-friendly installer.

Support Files

Then you can specify which file should be used as the **Readme** document. This document should contain any information that might be useful for the user before he installs the application. It should also contain a quick overview of what the setup package will actually install, so the user can make a good decision whether he wants to go through with the installation or not. Click on the Readme button to locate the appropriate file with an open file common dialog.

If you have specified a readme file, the setup program will present a button to the user labeled "More Information". When clicked, the document will be extracted from the compressed setup archive into a temporary folder and displayed. It is recommended to specify a .txt or .rtf file. These files are displayed inside the installation program. If you specify another type of file, the file will be shown to the user by extracting it into a temporary folder and running the default program to open it. If you know your users' computers support them, you can use .html, .chm or .pdf files. Using a .txt or .rtf file has the advantage that you can be 100% sure the user will be able to properly read the readme file. The other file types, certainly PDF, allow you to create a much nicer readme document to further improve the first impression.

Also, DeployMaster offers to show the readme file before the installation process starts, and not when it is finished, at which point installation tips and system requirements are no longer of any use. You will probably also want to add the readme file to the application's main component on the Files page, so the user can read it again later.

The **license agreement**, on the other hand, must be a plain text or a .rtf file. This is because it will be shown inside the DeployMaster installer, so the user cannot bypass it. As not to bloat the installer, only plain text files and basic .rtf files are supported. Basic means whatever you can create in the WordPad applet that comes with Windows. The WordPad viewer (the RichEdit common control) is embedded in Windows and can be easily accessed by applications such as DeployMaster. The license agreement is shown when the user clicks the Install button. When the user clicks Yes to accept the agreement, the installation will continue. If the user clicks No, the installation will be aborted.

You will probably also want to add the license agreement file to the application's main component on the Files page, so the user can read it again later. Not specifying any license agreement is perfectly okay for DeployMaster, though your lawyer may have a different opinion.

If you add a license agreement, you can tick "skip the license agreement when the application is already installed" to show the license agreement only the first time your is installed. The license agreement won't be shown when the same application is installed again (regardless of whether it is the same or a different version of that application). If you don't tick this checkbox, then the license agreement is shown before every installation.

Providing a **Support DLL** is optional. You can use a support DLL to add functionality to your installer that DeployMaster does not provide. The sample Support DLL source code includes comments that explain what you can do with a Support DLL.

Default Installation Folders

DeployMaster can generate installers that **install for all users**, **install for the current user**, or give the user the choice of either type of installation. Current user installs can be done with or without admin rights. Installing for all users is the most common way to install software. But that requires administrator rights. Offering the option to install for the current user without admin rights allows the software to be installed by people who don't have admin rights on their company PC.

Which of the two options you should choose or whether you can choose both depends on the installation requirements that your application has. Installing into protected folders like c:\Program Files or creating registry keys under HKEY_LOCAL_MACHINE requires admin rights. Placing data files into a user-specific folder such as "My Documents" or creating registry keys under HKEY_CURRENT_USER makes the installation specific to a user. If you want to support both types of installations, then your application will need to be adaptable to different installation folders (see below) and different registry branches (as explained for the Registry page).

If you do enable both types of installations, the user will be able to make their choice via the Advanced Installation button the first time they install your application. If they instead click the Immediate Installation or the Custom Installation button, then the installer chooses automatically. If the user has admin rights or is using an account that can be elevated to admin rights then the installation is done for all users. If your installer requires admin rights for a per-user install then the automatic choice is also to install for all users. So a per-user install without admin rights is only done automatically if the user does not have admin rights and their account does not support elevation. Basically, the installer will take full advantage of admin rights when they are available or when they are always required by your installer. It falls back to a user-specific install only to avoid a black screen security prompt for which the user may not have the password when their own Windows account cannot be used to pass this prompt.

Your installer can have two sets of default installation folders. The left hand set is used for installations for all users. The right hand set is used for installations for the current user. The user can change the actual installation folders via the Advanced Installation button. If you don't put files in any of these folders on the Files page then you should leave the default for that folder blank so that the user won't be prompted to select an unused folder during an advanced installation.

See the standard folder names topic for more information about the placeholders used in the default installation folders. You can add values on the Registry page if your application needs to determine the actual installation folders.

Default Application Folder is the folder that will be represented by %APPFOLDER% in the Files tab. This is the main installation folder where you place your application's executable and related non-data files. This is normally a subfolder of %PROGRAMFILES% for all-user installs and a subfolder of %LOCALAPPDATAROOT% for user-specific installs.

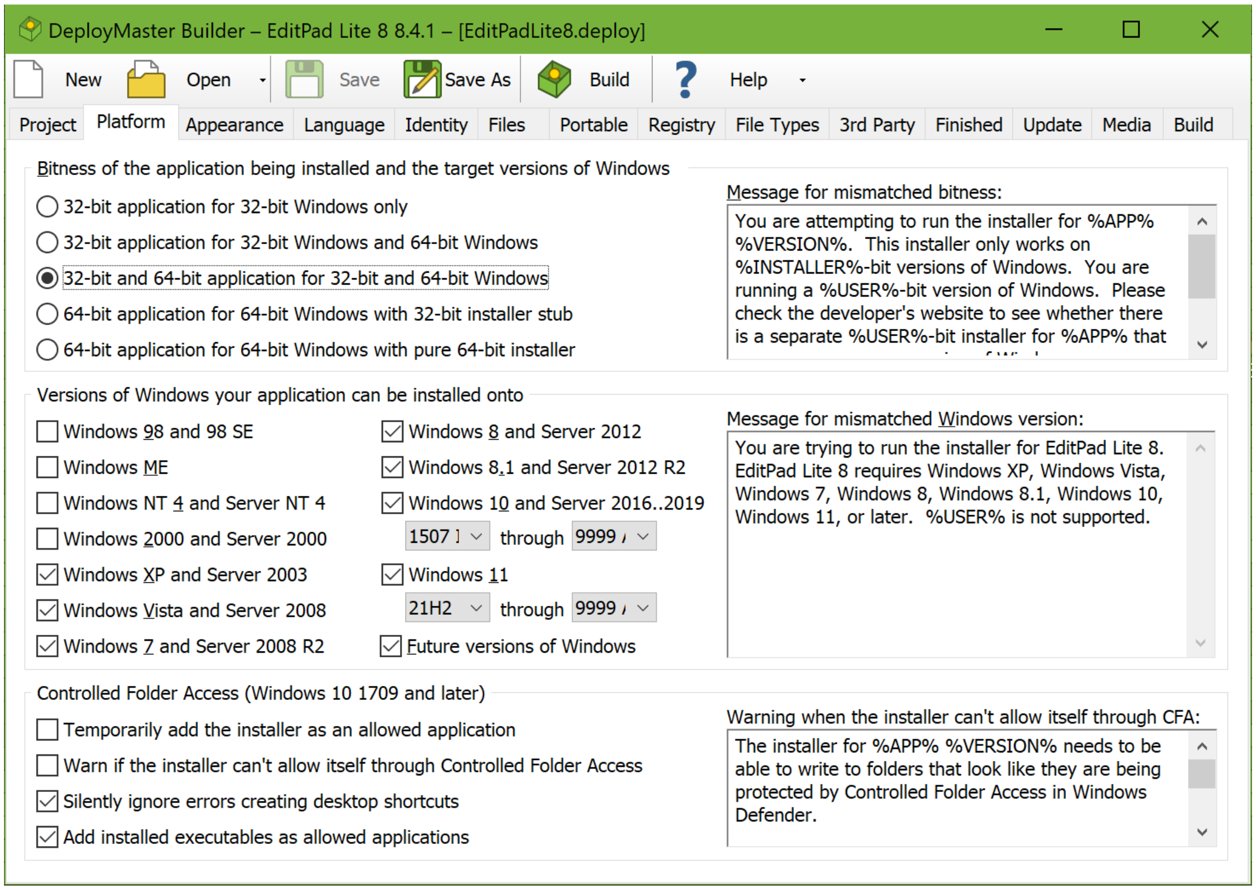
Default Common Files Folder is the default for %APPCOMMONFOLDER%. This folder can be used to install DLLs and other files that are shared between your applications. Sharing such files can save disk space but can also lead to versioning conflicts. This is normally a subfolder of %COMMONFILES% for all-user installs and a subfolder of %LOCALAPPDATAROOT% for user-specific installs.

Default Start Menu Folder is the default for %APPMENU%. This is where you put the shortcuts that you want to appear in the Start menu. It should be set to %PROGRAMSMENU% or a subfolder thereof for both all-user and user-specific installs.

Default User Data Folder is the default for %USERDATA%. This is where you put data files and settings files that the user may want to modify while using your application. This should be a subfolder of %COMMONAPPDATAROOT% or %COMMONDOCUMENTS% for all-user installs. Make it a subfolder of %APPDATAROOT% or %MYDOCUMENTS% for user-specific installs. The AppData folder is more appropriate for files like configuration files that the user doesn't deal with directly. The documents folder is more appropriate for files that store the user's work or results.

3. Platform

On the Platform page, you can select which versions of Windows your installer should support.



Bitness of the application being installed and the target versions of Windows

- ( ) 32-bit application for 32-bit Windows only
- ( ) 32-bit application for 32-bit Windows and 64-bit Windows
- (●) 32-bit and 64-bit application for 32-bit and 64-bit Windows
- ( ) 64-bit application for 64-bit Windows with 32-bit installer stub
- ( ) 64-bit application for 64-bit Windows with pure 64-bit installer

Message for mismatched bitness:

You are attempting to run the installer for %APP% %VERSION%. This installer only works on %INSTALLER%-bit versions of Windows. You are running a %USER%-bit version of Windows. Please check the developer's website to see whether there is a separate %USER%-bit installer for %APP% that

Versions of Windows your application can be installed onto

- [ ] Windows 98 and 98 SE
- [ ] Windows ME
- [ ] Windows NT 4 and Server NT 4
- [ ] Windows 2000 and Server 2000
- [✓] Windows XP and Server 2003
- [✓] Windows Vista and Server 2008
- [✓] Windows 7 and Server 2008 R2
- [✓] Windows 8 and Server 2012
- [✓] Windows 8.1 and Server 2012 R2
- [✓] Windows 10 and Server 2016..2019
  - 1507 through 9999
- [✓] Windows 11
  - 21H2 through 9999
- [✓] Future versions of Windows

Message for mismatched Windows version:

You are trying to run the installer for EditPad Lite 8. EditPad Lite 8 requires Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows 11, or later. %USER% is not supported.

Controlled Folder Access (Windows 10 1709 and later)

- [ ] Temporarily add the installer as an allowed application
- [ ] Warn if the installer can't allow itself through Controlled Folder Access
- [✓] Silently ignore errors creating desktop shortcuts
- [✓] Add installed executables as allowed applications

Warning when the installer can't allow itself through CFA:

The installer for %APP% %VERSION% needs to be able to write to folders that look like they are being protected by Controlled Folder Access in Windows Defender.

First, you need to decide whether your installer will support 32-bit Windows, or 64-bit Windows, or both. If your software comes in 32-bit and 64-bit versions, you can package both into a single installer, or you can decide to have two separate installers for 32-bit and 64-bit. A single installer is easier for your users because they don't need to figure out whether they need to download the 32-bit or 64-bit version. Separate installers may result in smaller downloads if your software has lots of files that come in separate 32-bit and 64-bit versions, like .exe files and .dll files.

32-bit application for 32-bit Windows only: Choose this option if your application won't work on 64-bit Windows at all, or if you're creating separate installers for the 32-bit and 64-bit versions of Windows. DeployMaster will generate a pure 32-bit installer. The "message for mismatched bitness" will be shown when the user runs the installer on 64-bit Windows.

32-bit application for 32-bit Windows and 64-bit Windows: Choose this option if you only have a 32-bit version of your application and the 32-bit version of your application runs just fine on 64-bit Windows. DeployMaster will generate a pure 32-bit installer that will run just fine on all current 64-bit versions of Windows.

32-bit and 64-bit application for 32-bit and 64-bit Windows: Choose this option if you have 32-bit and 64-bit versions of your application, and you want to package both into a single installer. On the Files page you can indicate which files are for the 32-bit version, which files are for the 64-bit version, and which files are shared by the 32-bit and 64-bit versions of your application. DeployMaster will generate an installer with a 32-bit stub that extracts and runs a 32-bit installer on 32-bit Windows and a 64-bit installer on 64-bit Windows.

64-bit application for 64-bit Windows with 32-bit installer stub: Choose this option if your application requires 64-bit Windows, or if you're creating separate installers for the 32-bit and 64-bit versions of Windows. A 32-bit stub is used so that the "message for mismatched bitness" can be shown when the user runs the installer on 32-bit Windows. The actual installer launched by the stub will be 64-bit. The installer will run fine on all present 64-bit versions of Windows. This option is recommended for 64-bit versions of end-user software.

64-bit application for 64-bit Windows with pure 64-bit installer: Choose this option if your application requires 64-bit Windows and you want your installer to support future versions of 64-bit Windows that may not be capable of running 32-bit software at all, or on servers where the administrator has disabled the WOW64 subsystem for running 32-bit applications. A system error message will be shown when the user runs the installer on 32-bit Windows. This error message will be cryptic, saying the setup.exe is not a valid executable, or simply "access denied", depending on which version of Windows the user is running. This option is recommended for 64-bit server software that people won't attempt to install on 32-bit machines.

Some of the platform options include a **message for mismatched bitness** in your installer. This message is shown in a message box when the user runs a 32-bit-only installer on 64-bit Windows, or a 64-bit-only installer on 32-bit Windows. You should replace the default message with a message that gives the user specific instructions how to obtain the correct installer for your software, or to tell the user that their version of Windows is not supported at all. You can use several placeholders in this message. %APP% represents the application name you've specified on the Project page. %INSTALLER% is replaced with 32 for 32-bit installers and with 64 for 64-bit installers. %USER% is replaced with 32-bit if the user has a 32-bit version of Windows, and with 64 if the user has a 64-bit version of Windows.

DeployMaster allows you to build a single installer that correctly installs your software on all versions of Windows since 1998. But your application may not support all those versions of Windows. Deselect the **versions of Windows** that your installer should not support. When the user runs your installer on a version of Windows that you've chosen not to support, it will show a message box with the **message for mismatched Windows version**. You can use several placeholders in this message. %APP% represents the application name you've specified on the Project page. %INSTALLER% is replaced with a slash-delimited list of the Windows versions your installer supports, such as "Windows 7/8/8.1/10/11". %USER% is replaced with the version of Windows your user is running, such as "Windows 11 23H2 (2023 Update)".

You can restrict support for **Windows 10** and **Windows 11** to a specific range of Windows 10 and 11 updates. Select the earliest and latest versions of Windows 10 and 11 that you want to support in the drop-down lists. For Windows 10 select 1507 and 9999 to support any version of Windows 10. For Windows 11 select 21H2 and 9999 to support any version of Windows 11.

The option for **future versions of Windows** allows your installer to run on Windows 12 or any version of Windows numbered greater than 11, if such a version of Windows is ever released. How well your installer and application will work on such a version of Windows will of course depends on how well Microsoft maintains backward compatibility with Windows 11.

Controlled Folder Access

Windows 10 version 1709 (Fall Creators Update) added a new feature to Windows Defender called **Controlled Folder Access**. It is disabled by default. It can be enabled via the Virus & Threat Protection settings in Windows Defender. Since Windows 10 version 1803 (April 2018 Update) this setting is labeled ransomware protection. This setting also exists in Windows 11. If enabled this prevents applications from writing to folders that are normally used to store personal files. That includes the Desktop and Documents folders. Additional folders can be added to the protection list. The standard folders cannot be removed from it. Even applications such as installers that run with Administrator privileges are blocked by Controlled Folder Access. If an application is to write to the Desktop and Documents folders, then it has to be added to the list of applications allowed through Controlled Folder Access.

When you enter a company name and application name on the Project page, DeployMaster automatically chooses default installation folders. None of these eight folders are protected by Controlled Folder Access (unless explicitly added by the user). When you add a new component to the Files page, DeployMaster adds a number of standard folders. When using the automatically chosen default installation folders, the only standard folder that is protected by Controlled Folder Access is the %DESKTOP% folder. So if you use only these default folders, then the only aspect of your installer that may be blocked by Controlled Folder Access is creating desktop shortcuts.

If the user has a 3rd party anti-virus or anti-malware solution on their PC, then Windows Defender is disabled on their PC. If you turn on any of the options involving Controlled Folder Access, then your installer first checks whether Windows Defender is operational. If it is not, then your installer ignores all these options.

Tick **"temporarily add the installer as an allowed application"** if you want your installer to automatically add itself as an application allowed through Controlled Folder Access, if Controlled Folder Access is enabled. It will do this after the user has clicked a button to start the installation and passed the elevation prompt to give the installer Administrator privileges. Windows Defender takes a couple of seconds to process the settings change. The installer needs to restart itself for the new privileges to take effect. It does this automatically. During all this the installer shows a progress meter making the user aware of this process. Some users may not appreciate installers that automatically change security settings. You'll need to decide whether your installer does anything that may be blocked by Controlled Folder Access that is important enough to have it allow itself through.

The installer does not get Administrator privileges when installing for the current user without admin rights and when creating a portable installation. Then it cannot add itself or any other applications to the applications allowed through Controlled Folder Access. It will silently ignore the options to add itself and any installed executables as allowed applications.

If your installer places essential files in folders that may be protected by Controlled Folder Access, then you should turn on the option to **warn if the installer can't allow itself through Controlled Folder Access**. Your installer then checks the Windows Defender settings in the registry to see whether it is being blocked by Controlled Folder Access. If it is then your installer shows the **warning when the installer can't allow itself through CFA**. You should use the placeholder %FILE% in this warning to indicate the executable file that needs to be allowed through Controlled Folder Access. This file will be the actual installer in the user's temporary files folder. It will not be the file that the user downloaded. That file is only a stub that extracts and runs the actual installer. It is the actual installer that needs to be allowed through Controlled Folder Access.

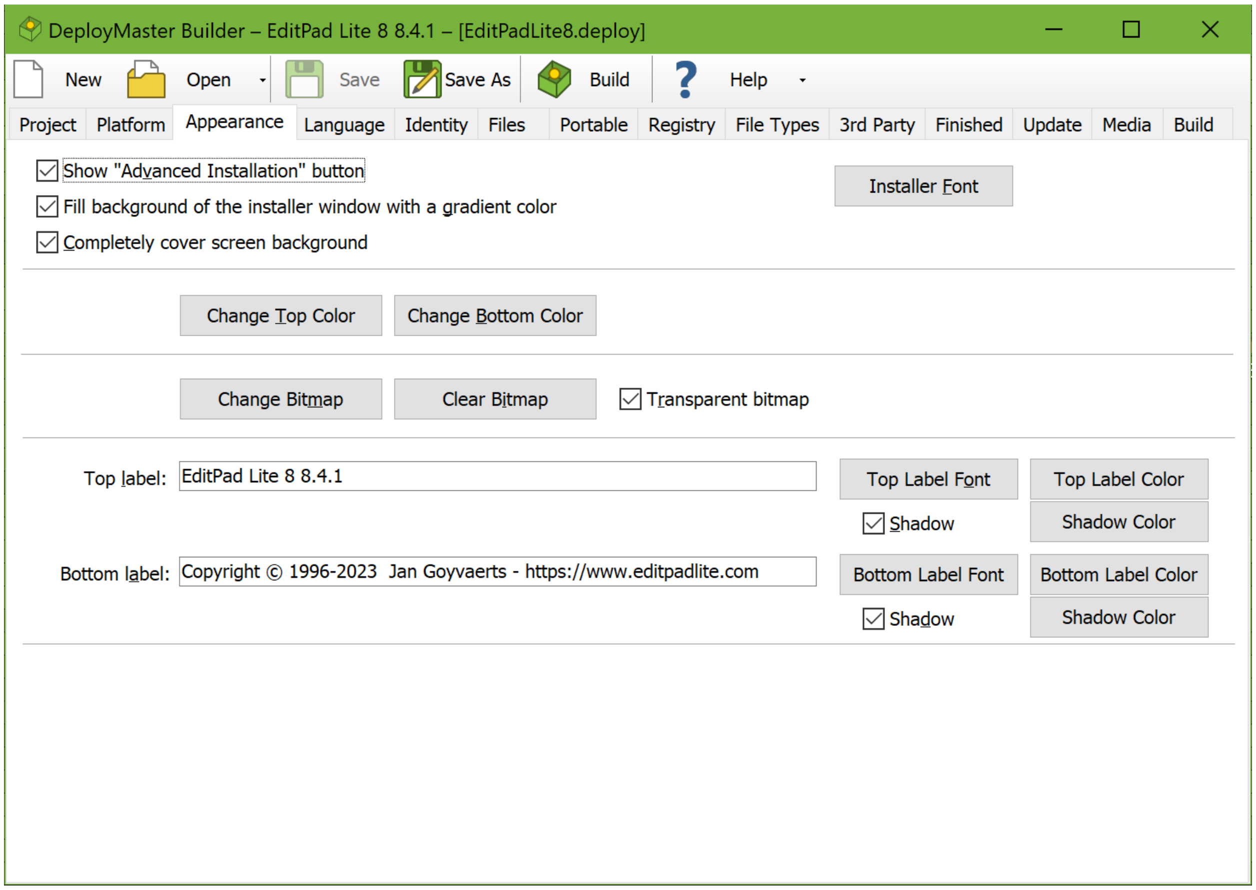
If your installer adds shortcuts to itself in %APPMENU% and %DESKTOP% then you may choose to **silently ignore errors creating desktop shortcuts**. Controlled Folder Access blocks the creation of desktop shortcuts. It does not block the creation of Start Menu shortcuts. If adding a desktop shortcut is the only thing your installer does that Controlled Folder Access doesn't like, then you may want to turn on this option and turn off the previous two options. There's no need to slow down your installer or bother the user with warnings and errors merely to create a desktop shortcut. The user can manually add a desktop shortcut later by dragging your application's icon from the Start Menu onto the Desktop.

If your installed application needs to be able to save files in folders that hold personal data files, then you should turn on the option to **add installed executables as allowed applications**. All .exe files installed by your installer are then allowed through Controlled Folder Access. If you turned on the option to run some of them on the Files page, then they will have already been allowed through before they are run. The .exe files are added even if Controlled Folder Access is disabled. This way, they will be allowed through if the user enables Controlled Folder Access after your application was installed.

Again, this will only work when your installer runs with Administrator privileges. If the ability to save files in personal data folders is critical to your application, you may want to have it verify this ability on its own. When using the Win32 API, calling CreateFile() to create a file that does not yet exist in a folder that does exist fails and GetLastError() returns ERROR_FILE_NOT_FOUND if that folder is protected by Controlled Folder Access and your application is not allowed through.

4. Appearance

Change the looks of your installation package on the Appearance page.



☑ Show "Advanced Installation" button
☑ Fill background of the installer window with a gradient color
☑ Completely cover screen background
☑ Transparent bitmap
Top label: EditPad Lite 8 8.4.1
☑ Shadow
Bottom label: Copyright © 1996-2023 Jan Goyvaerts - https://www.editpadlite.com
☑ Shadow

Show Advanced Installation button should always be on, except if you are sure that your setup packages will only be used by novices, who would be needlessly confused if they selected this option. If you disable this option, the end user will not be able to choose between installing for all users or for the current user or pick different installation folders or select which file associations can be created.

You can choose if you want to **fill the background of the installer window with a gradient color** or not. The gradient fill provides a nice visual effect.

Click the **Installer Font** button to select the font that should be used by the installation program. The font is not included in the setup program. You should select a font that is already present on the user's systems. Fonts that are part of Windows include Tahoma, Arial, and Courier New. If you plan to create a setup program in a language other than English, make sure to select the proper script when selecting the font. You can find the script option in the lower right corner of the font selection screen.

The **Completely cover screen background** checkbox determines whether your installation program will cover the whole screen or not. This gives your installation software the gradient-filled background that has become typical for installation software on the Windows platform. It does not serve any purpose, other that its visual effect. When you check this option, several options for defining the looks of the background appear:

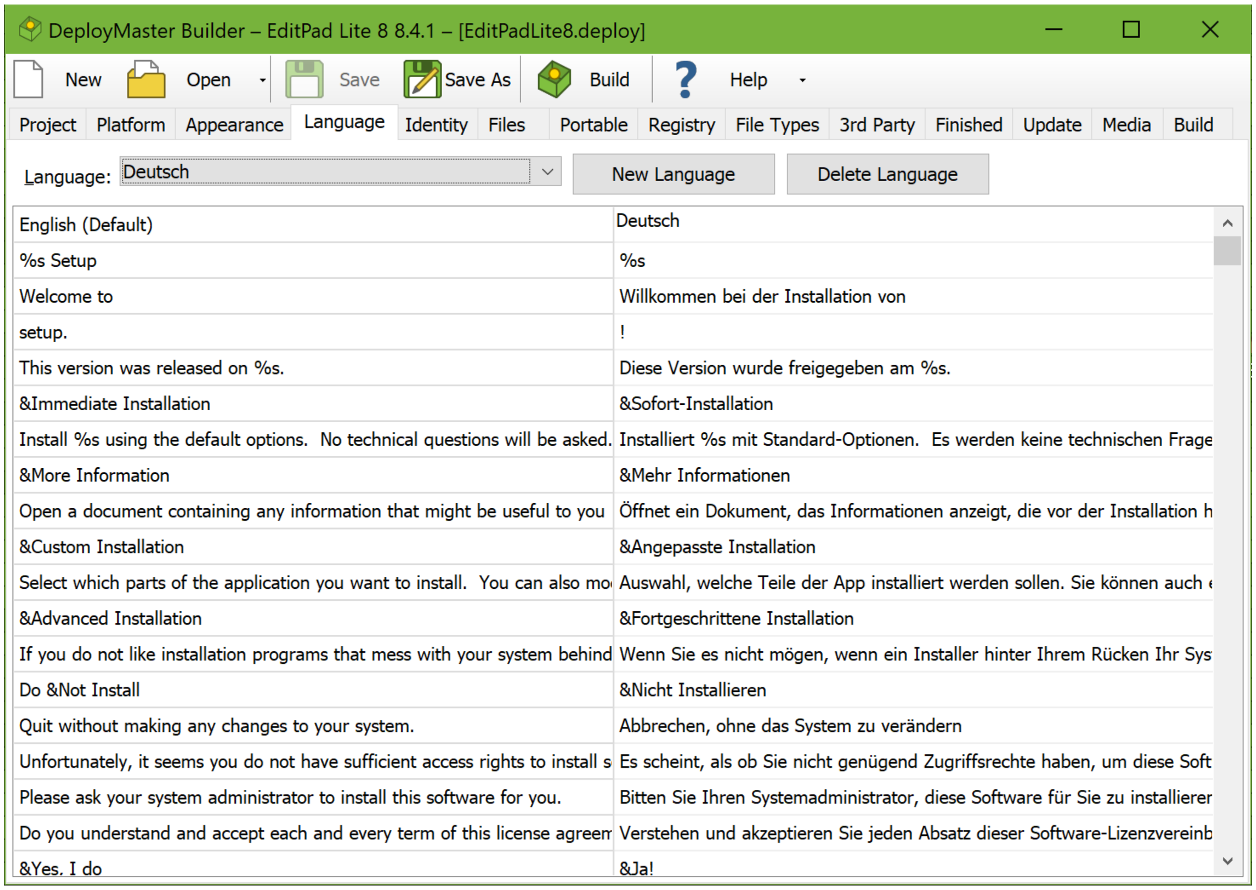
Change Top Color: Click this button to show up a color selection dialog box in which you can pick the color of the top of the background.
Change Bottom Color: Same story, but for the bottom color. The background is painted using a vertical, linear gradient fill.
Change Bitmap: Click this button if you want to display a bitmap in the top left corner of the background. This bitmap could typically be an application or company logo. Click **Clear bitmap** to get rid of it.
Transparent Bitmap: Mark this checkbox if you want the background bitmap to be transparent. The color of the lower-left pixel of the bitmap will be used as the background color. Wherever the bitmap's pixels have this color, the background will shine through. If the lower-left pixel does not have the appropriate color, use a paint program to add one row of pixels at the bottom of the image using the desired background color.

Top label: Text to be displayed at the top of the background. This is typically the name of the application the user is installing. If you want the label to span across more than one line, hit Ctrl+Enter to insert a line break while typing in the text.
Top label font: Shows a font selection dialog box for the top label.
Top label color: Since the standard Windows font selection dialog box allows you to choose between 16 basic colors only, you can click this button to select any color you would like for the top label.
Shadow: Mark this checkbox to paint a shadow behind the top label.
Shadow Color: Select a color for the top label's shadow.

The same options are available for the bottom label. This could typically be a company slogan, motto or copyright statement. It is probably also a good idea to mention your product or company URL in one of the labels.

5. Language

DeployMaster allows you to localize your installation software into any language you would like.



Language: Deutsch

English (Default)	Deutsch
%s Setup	%s
Welcome to	Willkommen bei der Installation von
setup.	!
This version was released on %s.	Diese Version wurde freigegeben am %s.
&Immediate Installation	&Sofort-Installation
Install %s using the default options. No technical questions will be asked.	Installiert %s mit Standard-Optionen. Es werden keine technischen Frage
&More Information	&Mehr Informationen
Open a document containing any information that might be useful to you	Öffnet ein Dokument, das Informationen anzeigt, die vor der Installation h
&Custom Installation	&Angepasste Installation
Select which parts of the application you want to install. You can also mo	Auswahl, welche Teile der App installiert werden sollen. Sie können auch
&Advanced Installation	&Fortgeschrittene Installation
If you do not like installation programs that mess with your system behind	Wenn Sie es nicht mögen, wenn ein Installer hinter Ihrem Rücken Ihr Sys
Do &Not Install	&Nicht Installieren
Quit without making any changes to your system.	Abbrechen, ohne das System zu verändern
Unfortunately, it seems you do not have sufficient access rights to install s	Es scheint, als ob Sie nicht genügend Zugriffsrechte haben, um diese Soft
Please ask your system administrator to install this software for you.	Bitten Sie Ihren Systemadministrator, diese Software für Sie zu installierer
Do you understand and accept each and every term of this license agreem	Verstehen und akzeptieren Sie jeden Absatz dieser Software-Lizenzvereinb
&Yes, I do	&Ja!

You can select any of the predefined languages from the drop-down list. If you would like to rephrase some of the text, simply edit it in the grid.

If your language is not present, click the New Language button. This will create a new language which will initially be identical to the default language (English). Translate all the strings in the grid. If you want your setup to be in English, but change some of the terminology, simply create a new language and edit those strings that you want to rephrase.

Any changes you make in the language grid will be automatically be saved in DeployMaster's settings. The grid's contents are not stored into your setup script, only the name of the language desired is. This means that if you correct a mistake in a translation you made, any setup scripts you previously created will automatically pick up the change.

All languages except for the default one are stored in separate .language files. These are stored in the user data folder that you can configure via the Advanced Installation button in DeployMaster's installer. If you want to change this folder, run DeployMaster's installer again and click Advanced Installation. You will not lose any settings or files if you do this without uninstalling DeployMaster.

If you have translated the strings into a new language, feel free to send the .language file to support@deploymaster.com. These files are plain text files. You can edit them with a text editor. Just make sure not to add or delete any lines in the file. Just translate the lines already present.

If you want your installer to use a language that does not use the same alphabet as Western European languages, you will probably need to select the proper font before you can make the translation. To select the font, click the Installer Font button on the Appearance page. Make sure to select the proper script when selecting the font. You can find the script option in the lower right corner of the font selection screen.

Here is a list of the strings DeployMaster uses in the installation software it creates along with an explanation where you can find the strings in the final result. These explanations should help explain the context to make a good translation possible.

```
English
```

The name of the language. Used in DeployMaster Builder (to build the drop-down list on the language page), but not in the produced setup software.

```
%s Setup
```

The caption of the setup dialog box. %s is replaced with CompanyName ApplicationName VersionNumber as specified in the Project tab.

```
Welcome to
setup.
```

These two strings are used to form the phrase: "Welcome to <Company> <AppName AppVersion> setup."

```
This version was released on %s.
```

Shown right below the previous string. %s will be replaced with the release date you specified, expanded into the long date format using the user's settings from the Regional applet in the Control Panel.

```
&Immediate Installation
Install %s using the default options. No technical questions will be asked.
&More Information
Open a document containing any information that might be useful to you before
installing this software.
&Custom Installation
Select which parts of the application you want to install. You can also modify an
existing installation by adding or removing components.
&Advanced Installation
If you do not like installation programs that mess with your system behind your
back, Advanced Installation allows you to monitor and adjust the installation in
all its detail.
Do &Not Install
Quit without making any changes to your system.
```

Captions and hints of the five buttons that are presented to the user when she executes your setup package. The ampersand (&) in a caption makes the letter that follows it a hotkey, i.e. the user can switch to the associated control by pressing Alt+hotkey on the keyboard. Be careful not to use the same hotkey twice. The hint is shown in the area to the right of the buttons, when the mouse pointer is hovering over the corresponding button.

```
Unfortunately, it seems you do not have sufficient access rights to install
software on this computer.
Please ask your system administrator to install this software for you.
```

These two lines are shown if DeployMaster detects that the user does not have sufficient access rights to the system to install software. This will only happen on Windows NT, 2000 or XP if the user is not an Administrator or Power User. On Windows Vista and later, your setup will automatically request elevation to Adminstrator rights, which involves a security prompt when using Vista's default security settings.

```
Do you understand and accept each and every term of this license agreement?
&Yes, I do
&No, I do not Print
```

Used to build the license agreement screen, providing a label and three button captions.

```
Please provide the following details:
Your &name:
Please type in your own name.
Your &company:
Please provide the name of the company you work for.
Product &serial number:
Please specify the serial number printed on the software's installation media.
Product &registration code:
Please enter the registration code which you received when purchasing this
software.
```

Caption and hint pairs used to build the optional identity request screen, instructing the user which information needs to be supplied. Of course, nothing prevents you from giving these another meaning. You could change "product serial number" to "membership card number" if your software is for members only. When assigning hotkeys, keep in mind that a proceed and abort button will be shown too, using two of the captions right below.

```
&Back
&Proceed
&Abort
```

Button captions that allow the user to indicate he (thinks he) provided the requested info and wants to go on or to abort the installation. Note that the third caption reads abort and not cancel. "Abort" means "stop where you are" and "cancel" means "stop and go back where you came from". Except for cleaning up temporary files, nothing is being undone (which cancel would imply).

```
Program &Files folder:
&Common Files folder:
&Start Menu folder:
&User data folder:
```

When clicking on the Advanced Installation button, the user has a chance to specify which folders the application should end up in. These are the captions for the edit controls used for that purpose. The hotkeys should not conflict with those of the three button captions right above.

```
Please select which file extensions this software is allowed to associate itself
with:
```

The second screen in Advanced Installation allows the user to select which file associations can be made and which not. Above the checkboxes, this caption is show.

```
OK
Cancel
```

Generic captions for OK and Cancel buttons.

```
Installation will use %s. That is %d%% of the available disk space.
```

String used in the component selection screen to indicate the disk space required. %s is replaced with the number of kilo/mega/gigabytes needed. %d%% is replaced with the percentage of free disk space this represents.

```
&Full Detail
```

When arriving at the component selection screen after clicking on the Advanced Installation button, the user can mark a checkbox showing this label. When marked, DeployMaster will show him all the files that will be placed on his computer's hard disk. The standard view only shows the component names, as not to confuse users with limited experience regarding computers.

```
Gathering the list of files to install...
Deploying file %s...
Creating shortcut %s...
Creating registry items...
Creating file associations...
Removing files that are no longer needed...
```

These captions are shown together with the progress indicators during the installation.

```
%s has been successfully installed on your system
Thanks!
```

Text and button caption shown when the installation is complete. %s is replaced with the name of the application.

```
To complete the installation, your computer needs to be rebooted
Reboot now
Reboot later
```

Text and two button captions shown when all files have been copied, but a system reboot is necessary to complete the installation.

```
Please insert disk %d
```

When splitting the setup into multiple files this text is shown when the next file is needed. %d is replaced with the file's number. The most common reason to split a setup is to spread it across multiple physical disks when one disk is not large enough to hold the entire installer. File numbers then correspond with disk numbers.

```
DeployMaster has detected that %s is already installed on your computer.
```

Shown when DeployMaster detects that a copy of your software has already been installed. %s is replaced with the application's name.

```
The version in this setup package is more recent than the one you have already
installed.
Proceeding will upgrade the copy on your computer to this new version.
```

Two lines shown when the existing version is older.

```
This is the same version of the application in this setup package.
If you wish, you can select different components now and your system will be
updated.
Newly selected components will be installed and components marked for deletion
will be removed.
```

Three lines shown indicating that the user is installing the same version, reminding her of the fact she can change which components are installed and which not.

```
This setup package contains an older version of this application.
Though not recommended, you can downgrade your existing installation to this
version.
```

Two lines shown when the existing version is more recent, discouraging the user to revert to the old version (but not preventing him from doing so).

```
Continuing with this operation will completely remove %s from your system.
Are you sure you want this to happen?
Note that you can remove or add individual components of this application by
running the original installation program again.
```

This text is shown by the UnDeploy application when the user selects your application to be uninstalled through the Control Panel, asking if the user wants to continue with the uninstallation or not.

```
%s has been successfully removed from your system.
Some files could not be uninstalled right now, because they are in use by Windows.
They will be removed the next time you start up your computer.
```

This text is shown after UnDeploy has completed its work. If some files could not be deleted because they were still in use, the user will be notified of that.

```
You need administrator privileges to uninstall %s.
```

Shown by UnDeploy when it cannot obtain the administrator privileges needed to uninstall an application that was installed for all users or that was installed for the current user requiring admin privileges.

```
Remove %s
```

Caption used for the uninstall shortcut that DeployMaster can automatically create in %APPMENU%. (The option for this is on the Finished page.)

```
The following error has occurred while trying to install %s:
Do you wish to continue with the installation?
```

If an error occurs, an error message will be shown. The first line will be shown above the actual error message, announcing the error, the second line is shown below the actual error message asking the user if the installation should continue or be aborted.

```
You have aborted the installation.
%s has been installed. Remember that errors have occurred.
```

If an error occurred during the installation, one of the two above lines is shown instead of “%s has been successfully installed on your system”. The first line is shown if the user selected to abort the installation, the second one if the user chose to continue anyway. %s is replaced with the name of your application.

```
The application will not be installed properly and may not function.
This is not an essential part of the application, so continuing the installation
should not cause any problems.
```

When an error occurs and DeployMaster asks the user if the installation should continue or be aborted, it will try to help the user by indicating the probable severity of the error. If the error is probably severe, the first line is shown. Otherwise, the second line is shown.

```
Unable to extract %s from the installation package. The package has most likely
been damaged. Please contact %s to obtain a new one.
Unable to create the file %s on your computer.
Unable to create the folder %s on your computer.
```

These items are used to construct meaningful error messages in case something goes wrong during the installation.

```
Please specify a valid path for each installation folder.
```

This error appears in a popup message when the user types in an invalid path during advanced installation.

```
There is not enough free disk space on drive %0:s for this application. The
installation requires %1:s of space, but only %2:s is available.
```

This error appears in a popup message when the user proceeds from the component selection screen with more components selected than fit on the target drive. Though this is rare with huge modern hard disks, portable installations are often created on devices with very limited space.

```
Create &Portable Installation
Install the application onto a removable medium like a USB stick or memory card,
without making any changes to the host computer.
```

Caption and hint for the Create Portable Installation button in the welcome screen of your setup. You can enable this button on the Portable tab in DeployMaster.

```
Please select the &drive you want to create the portable installation on:
Drive
Volume Label
```

```
Free Space
No disk or card inserted
Treat &all drives as removable drives
You can type in the name of a &folder that should be created for this application:
```

Labels used to build the drive selection screen that appears after clicking the Create Portable Installation button.

```
This application requires the .NET framework version %s to be installed.
Its installation program has been started.
Please download and install it from:
This application requires %s to be installed.
```

Labels used to indicate that the application requires a 3rd party package to be installed. Your setup program will display this screen while it waits for the 3rd party installer to finish its job. If the 3rd party installer is not available, the setup will display the download URL and stay stuck at this screen until the user finishes downloading and installing the 3rd party package, or aborts the setup.

```
You can only create a portable installation on a personal storage device.
```

This appears instead of “Please ask your system administrator to install this software for you.” when the user does not have sufficient rights to install software, and the portable installation option has been enabled on the Portable tab in DeployMaster.

```
The application you are about to install is still running. You cannot install a
new version of “%s” while it is still running.
To continue with the installation, please close the application first and then
click Retry.
To continue with the installation, please close the window with caption “%s” first
and then click Retry.
To cancel the installation, click Cancel.
```

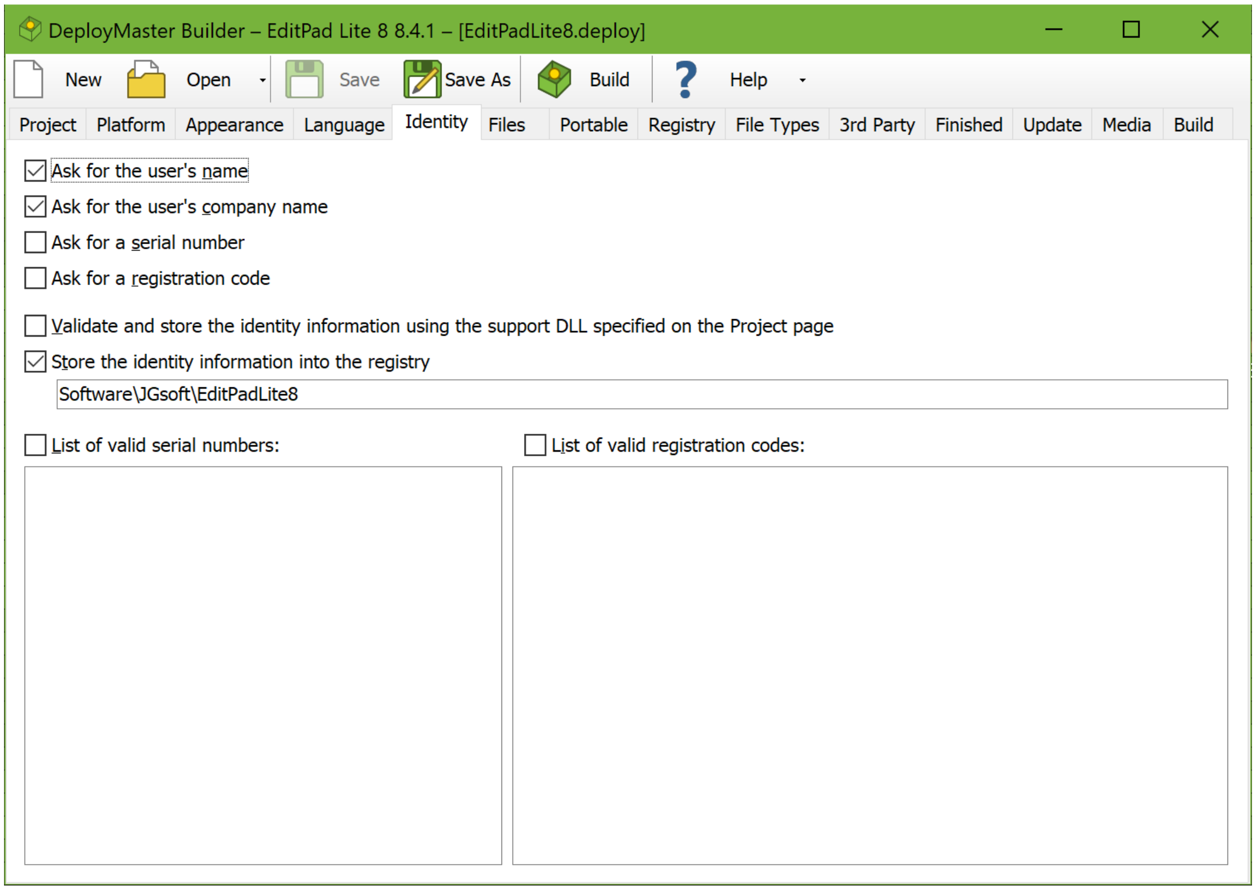
This message is shown if your setup detects any open windows that use one of the window classes or window captions that you’ve specified on the Update page. The message will have standard Retry and Cancel buttons.

```
Install for &current user
Install %s into a personal folder and create shortcuts and file associations only
for your own Windows user account. This does not require administrator privileges.
Install %s into a system folder and create shortcuts and file associations only
for your own Windows user account. This still requires administrator privileges.
Install for &all users
Install %s into a common folder and create shortcuts and file associations for all
Windows accounts on this PC. This requires administrator privileges.
```

Captions and hints of the two buttons that allow the user to choose between an installation for the current user or one for all users after clicking the Advanced Installation button if you enabled both choices on the Project page.

6. Identity

If you want your installation program to personalize the installed copy, or if only authorized (paying) users should be allowed to install your software, DeployMaster's Identity features will help you out.



[✓] Ask for the user's name
[✓] Ask for the user's company name
[ ] Ask for a serial number
[ ] Ask for a registration code
[ ] Validate and store the identity information using the support DLL specified on the Project page
[✓] Store the identity information into the registry
Software\JGsoft\EditPadLite8
[ ] List of valid serial numbers:
[ ] List of valid registration codes:

You can have your setup program request any or all of these:

- The user's name
- The name of the company the user works for
- A serial number (e.g.: a CD key)
- A registration code (e.g.: a code that registers your shareware application, computed on the user's name)

As mentioned in the Language topic, you can edit the text that the user will see. If your application can only be installed with a valid CD key, you could mark the "ask for a serial number" checkbox on the Identity page and change "Product &serial number" into "CD key printed on the back of the jewel case". This will make it easier for the user to figure out exactly what to supply. You can create a different language for this, which could be named "English (CD key)".

There are two ways to validate and store the identity information the user entered. The most thorough method is do use the support DLL that you specify on the Project page. This DLL must implement the four identity DLL routines that DeployMaster will call. Sample DLLs in C and Pascal are provided. The same DLL can also provide other support routines for your installation software.

The other way is to have DeployMaster store the information in the registry. Specify a registry key under Software such as Software\YourCompany\YourApp that DeployMaster should use for this purpose. The key will be created under HKEY_LOCAL_MACHINE for an all-users install and under HKEY_CURRENT_USER for user-specific installs. The user's name will be stored in a value called "Name". The company name will be stored as "Company", the serial number as "Serial" and the registration code as "RegCode".

If you ask for a serial number and/or registration code, you can have DeployMaster compare the information entered by the user against a hard-coded list of valid serial numbers and/or registration codes. To do so, tick the corresponding checkbox and enter the list of valid codes.

Note: DeployMaster does not provide any copy protection beyond asking for a serial number and/or registration code. If your product needs strong copy protection, you must apply the copy protection to the software itself, not to the installation program.

Identity DLL routines

If you want the user to provide some kind of identification (indicate this on the Identity page), that information will need to be processed. You will need to create a DLL for this. Specify the DLL you want to use as the support DLL on the Project page. The support DLL can also provide other functionality besides managing the user's identity information.

The DLL must export the following four routines. Your DLL can export either the Ansi or the Unicode versions.

Pascal syntax (Ansi):

```
procedure InitIdentity(CompanyName, AppName, AppVersion: PAnsiChar); stdcall;
function ValidIdentity(Name, Company, Serial, RegCode: PAnsiChar): Bool; stdcall;
procedure LoadIdentity(Name, Company, Serial, RegCode: PAnsiChar); stdcall;
procedure SaveIdentity(Name, Company, Serial, RegCode: PAnsiChar); stdcall;
```

Pascal syntax (Unicode):

```
procedure InitIdentityW(CompanyName, AppName, AppVersion: PWideChar); stdcall;
function ValidIdentityW(Name, Company, Serial, RegCode: PWideChar): Bool; stdcall;
procedure LoadIdentityW(Name, Company, Serial, RegCode: PWideChar); stdcall;
procedure SaveIdentityW(Name, Company, Serial, RegCode: PWideChar); stdcall;
```

C syntax (append W to the function names when UNICODE is defined):

```
void __stdcall InitIdentity(LPCTSTR CompanyName, LPCTSTR AppName, LPCTSTR
AppVersion);
BOOL __stdcall ValidIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial, LPTSTR
RegCode);
void __stdcall LoadIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial, LPTSTR
RegCode);
```

```
void __stdcall SaveIdentity(LPCTSTR Name, LPCTSTR Company, LPCTSTR Serial,
LPCTSTR);
```

`InitIdentity()` is called right after the DLL is loaded. This allows the DLL to initialize itself, based on which application is being installed. The sample DLL will open the key `HKEY_CURRENT_USER\Software\CompanyName\AppName` in the Windows registry, where it will store the identity information. The parameters point to stings containing the information you specified on the Project page.
Note that `CompanyName` and `AppVersion` may be NULL, as these are not required fields. The sample DLL will open `HKEY_CURRENT_USER\Software\AppName` if no company name was specified.

`LoadIdentity()` is called right after InitIdentity() is called. If (another version of) the application is already installed, the DLL should be nice to the user and present the data entered when the previous install was done. Otherwise, the DLL has a chance to provide default data. If your setup tool asks for a registration code, the default would be a code that enables a 30-day trail mode of your application.
The sample DLL will try to load any previously written data from the registry and will return empty strings if no previous data exists.
If the user is not being requested to enter a certain piece of information, the corresponding argument will be NULL. It should remain untouched.
The other arguments will each be pointing to a character array that is 128 bytes in size. The function should write its data into these arrays. Mind the zero termination character, so the maximum string length is 127 characters. That should be enough.

`ValidIdentity()` is called when the user clicks on the Proceed button. It should return True if the entered information is valid, False otherwise. When True is returned, the installation will proceed. In the other case, nothing will happen. Since the user may be puzzled why her information is not being accepted, you may want to show up a message box in `ValidIdentity()` to inform the user about the reason why the installation is not proceeding.
The arguments are the same as for `LoadIdentity()`. Any changes made to them by `ValidIdentity()` will cause the edit fields in the setup application to be updated. You might want to erase an invalid registration code, to hamper guesswork.
The sample DLL accepts any information entered.

`SaveIdentity()` is called after `ValidIdentity()` has returned True. It should save the data somewhere so that it can be loaded by the application after it is installed and run. Right after this call, the DLL will be unloaded.
The arguments are again the same. There is no need for `SaveIdentity()` to modify them, as DeployMaster will dispose the strings right after the call returns.
The sample DLL writes the information to the Windows registry, under the key opened in `InitIdentity()`. The key is then closed.

Note: You should not rely on your Identity DLL as the only security measure against illegal use of your software. DeployMaster is designed to be a tool for creating nice and convenient installation programs, not as a security tool. Asking for identity information in the installation program, is nice and convenient for legitimate users. However, the actual security measures should be embedded into the actual software. So the DLL should do some basic validation to reject typing errors by legitimate users and store the data, so the installed application can do the real checking.

Sample Identity DLL (Pascal)

A C version of this DLL is also available

```
{*********************************************************}
{                                                         }
{       DeployMaster Sample Identity Manager DLL          }
{                                                         }
{*********************************************************}

library Identity;

uses
  Windows, SysUtils;

{$R *.RES}

var
  Key: HKey;

// This routine is called after the DLL is loaded.
// It allows it to figure out where the processed information should be stored in
the Windows registry.
// CompanyName and AppVersion may be nil, AppName will always have a value.
// The strings must not be modified by the DLL, and their length is arbitrary.
procedure InitIdentity(CompanyName, AppName, AppVersion: PChar); stdcall;
var
  ZStr: array[0..255] of Char;
begin
  ZStr := 'Software';
  if CompanyName <> nil then begin
    StrCat(ZStr, '\');
    StrCat(ZStr, CompanyName);
  end;
  StrCat(ZStr, '\');
  StrCat(ZStr, AppName);
  if RegCreateKeyEx(HKEY_CURRENT_USER, ZStr, 0, nil, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, nil, Key, nil) <> 0 then
    Key := 0;
end;

// ValidIdentity() is called when the user clicks on the Proceed button.
// Should return True if the information is valid and the user may proceed.
// If it returns False, the user will have to retry.
// The parameters for items the user is not requested to specify will be nil
// ValidIdentity() is allowed to modify the contents of the strings; e.g. to clear
out an invalid registration code
function ValidIdentity(Name, Company, Serial, RegCode: PChar): Bool; stdcall;
begin
  Result := True;
end;

// LoadIdentity() is called right before the user is allow to supply his
information
// If (another version of) the application is already installed,
// it should set the strings to the previously entered data
// If not, the DLL has a chance to provide default data (e.g. a time-limited trial
mode registration key)
// The parameters for items the user is not requested to specify will be nil
```

```
procedure LoadIdentity(Name, Company, Serial, RegCode: PChar); stdcall;
var
  BufType, BufSize: Integer;
begin
  if Key <> 0 then begin
    BufSize := 128;
    if Name <> nil then RegQueryValueEx(Key, 'Name', nil, @BufType, PByte(Name),
@BufSize);
    BufSize := 128;
    if Company <> nil then RegQueryValueEx(Key, 'Company', nil, @BufType,
PByte(Company), @BufSize);
    BufSize := 128;
    if Serial <> nil then RegQueryValueEx(Key, 'Serial', nil, @BufType,
PByte(Serial), @BufSize);
    BufSize := 128;
    if RegCode <> nil then RegQueryValueEx(Key, 'RegCode', nil, @BufType,
PByte(RegCode), @BufSize);
  end;
end;

// SaveIdentity() is called after the user clicked the proceed button and
ValidIdentity() returned true.
// It should write the data to the Windows registry, so that the application, once
installed, can use it.
// The parameters for items the user is not requested to specify will be nil
procedure SaveIdentity(Name, Company, Serial, RegCode: PChar); stdcall;
begin
  if Key <> 0 then begin
    // Save data
    if Name <> nil then RegSetValueEx(Key, 'Name', 0, REG_SZ, Name,
StrLen(Name)+1);
    if Company <> nil then RegSetValueEx(Key, 'Company', 0, REG_SZ, Company,
StrLen(Company)+1);
    if Serial <> nil then RegSetValueEx(Key, 'Serial', 0, REG_SZ, Serial,
StrLen(Serial)+1);
    if RegCode <> nil then RegSetValueEx(Key, 'RegCode', 0, REG_SZ, RegCode,
StrLen(RegCode)+1);
    // Clean up
    RegCloseKey(Key);
  end;
end;

exports
  // Make our support routines visible to the world
  // If we're using Unicode, we need to add W to the names of the exported
functions
  InitIdentity {$IFDEF UNICODE}name 'InitIdentityW'{$ENDIF},
  ValidIdentity {$IFDEF UNICODE}name 'ValidIdentityW'{$ENDIF},
  LoadIdentity {$IFDEF UNICODE}name 'LoadIdentityW'{$ENDIF},
  SaveIdentity {$IFDEF UNICODE}name 'SaveIdentityW'{$ENDIF};

end.
```

Sample Identity DLL (C)

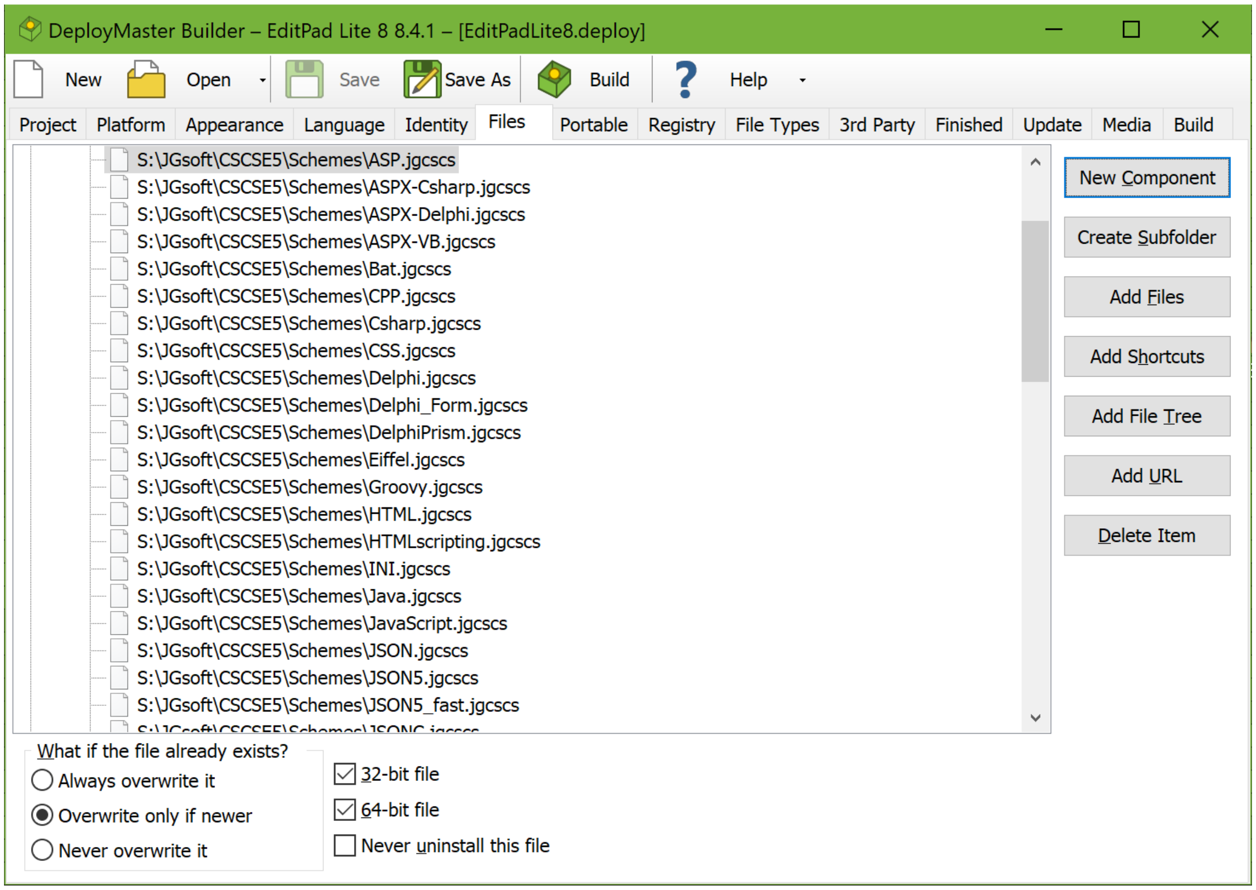
A Pascal version of this DLL is also available

This code assumes that UNICODE is not defined when compiling your DLL. If it is defined, append a W to all functions.

```
/**********************************************************
*                                                         *
*        DeployMaster Sample Identity Manager DLL         *
*                                                         *
**********************************************************/

#include <windows.h>

HKEY Key;

// This routine is called after the DLL is loaded.
// It allows it to figure out where the processed information should be stored in
the Windows registry.
// CompanyName and AppVersion may be NULL, AppName will always have a value.
// The strings must not be modified by the DLL, and their length is arbitrary.
void __export __stdcall InitIdentity(LPCTSTR CompanyName, LPCTSTR AppName, LPCTSTR
AppVersion)
{
  TCHAR ZStr[256];
  strcpy(ZStr, “Software”);
  if (CompanyName != NULL) {
    strcat(ZStr, “\\”);
    strcat(ZStr, CompanyName);
  }
  strcat(ZStr, “\\”);
  strcat(ZStr, AppName);
  if (RegCreateKeyEx(HKEY_CURRENT_USER, &ZStr, 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &Key, NULL) != 0)
    Key = 0;
}

// ValidIdentity() is called when the user clicks on the Proceed button.
// Should return True if the information is valid and the user may proceed.  If it
returns False, the user will have to retry.
// The parameters for items the user is not requested to specify will be NULL
// ValidIdentity() is allowed to modify the contents of the strings; e.g. to clear
out an invalid registration code
BOOL __export __stdcall ValidIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial,
LPTSTR RegCode)
{
  return TRUE;
}

// LoadIdentity() is called right before the user is allow to supply his
information
// If (another version of) the application is already installed, it should set the
strings to the previously entered data
// If not, the DLL has a chance to provide default data (e.g. a time-limited trial
mode registration key)
// The parameters for items the user is not requested to specify will be NULL
void __export __stdcall LoadIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial,
LPTSTR RegCode)
{
  DWORD BufType, BufSize;
  if (Key != 0) {
    BufSize = 128;
    if (Name != NULL) RegQueryValueEx(Key, “Name”, NULL, &BufType, (LPBYTE)Name,
&BufSize);
    BufSize = 128;
    if (Company != NULL) RegQueryValueEx(Key, “Company”, NULL, &BufType,
(LPBYTE)Company, &BufSize);
    BufSize = 128;
    if (Serial != NULL) RegQueryValueEx(Key, “Serial”, NULL, &BufType,
(LPBYTE)Serial, &BufSize);
    BufSize = 128;
    if (RegCode != NULL) RegQueryValueEx(Key, “RegCode”, NULL, &BufType,
(LPBYTE)RegCode, &BufSize);
  }
}

// SaveIdentity() is called after the user clicked the proceed button and
ValidIdentity() returned true.
// It should write the data to the Windows registry, so that the application, once
installed, can use it.
// The parameters for items the user is not requested to specify will be NULL
void __export __stdcall SaveIdentity(LPCTSTR Name, LPCTSTR Company, LPCTSTR
Serial, LPCTSTR RegCode)
{
  if (Key != 0) {
    // Save data
    if (Name != NULL) RegSetValueEx(Key, “Name”, 0, REG_SZ, (CONST BYTE*)Name,
strlen(Name)+1);
    if (Company != NULL) RegSetValueEx(Key, “Company”, 0, REG_SZ, (CONST
BYTE*)Company, strlen(Company)+1);
    if (Serial != NULL) RegSetValueEx(Key, “Serial”, 0, REG_SZ, (CONST
BYTE*)Serial, strlen(Serial)+1);
    if (RegCode != NULL) RegSetValueEx(Key, “RegCode”, 0, REG_SZ, (CONST
BYTE*)RegCode, strlen(RegCode)+1);
    // Clean up
    RegCloseKey(Key);
  }
}
```

7. Files

On the files page, you will need to specify which files will be placed in the installation package and where (and if) they will be placed on the user's system.



```
S:\JGsoft\CSCSE5\Schemes\ASP.jgcscs
S:\JGsoft\CSCSE5\Schemes\ASPX-Csharp.jgcscs
S:\JGsoft\CSCSE5\Schemes\ASPX-Delphi.jgcscs
S:\JGsoft\CSCSE5\Schemes\ASPX-VB.jgcscs
S:\JGsoft\CSCSE5\Schemes\Bat.jgcscs
S:\JGsoft\CSCSE5\Schemes\CPP.jgcscs
S:\JGsoft\CSCSE5\Schemes\Csharp.jgcscs
S:\JGsoft\CSCSE5\Schemes\CSS.jgcscs
S:\JGsoft\CSCSE5\Schemes\Delphi.jgcscs
S:\JGsoft\CSCSE5\Schemes\Delphi_Form.jgcscs
S:\JGsoft\CSCSE5\Schemes\DelphiPrism.jgcscs
S:\JGsoft\CSCSE5\Schemes\Eiffel.jgcscs
S:\JGsoft\CSCSE5\Schemes\Groovy.jgcscs
S:\JGsoft\CSCSE5\Schemes\HTML.jgcscs
S:\JGsoft\CSCSE5\Schemes\HTMLscripting.jgcscs
S:\JGsoft\CSCSE5\Schemes\INI.jgcscs
S:\JGsoft\CSCSE5\Schemes\Java.jgcscs
S:\JGsoft\CSCSE5\Schemes\JavaScript.jgcscs
S:\JGsoft\CSCSE5\Schemes\JSON.jgcscs
S:\JGsoft\CSCSE5\Schemes\JSON5.jgcscs
S:\JGsoft\CSCSE5\Schemes\JSON5_fast.jgcscs
```

What if the file already exists?
- ○ Always overwrite it
- ● Overwrite only if newer
- ○ Never overwrite it

- ✓ 32-bit file
- ✓ 64-bit file
- ☐ Never uninstall this file

All files must be placed in folders, contained by components. In addition to files, you can also place shortcuts, URLs and other folders in folders.

Components

A DeployMaster installation package consists of one or more components. These components cannot be nested. Give the component a meaningful name by editing its item in the tree view. At the bottom of the window, you can edit the component's options:

Install component by default: When checked, the component will be installed, unless the user deselects it in the Select Components page of the installation program. Mark this checkbox for any component the typical user will want to install. Note that if the user does not click on the Custom Installation or Advanced Installation button, he will not be given the chance to select which components need to be installed. If a component must always be installed for the application to be useful, turn off the following option.

User can select whether this component is installed or not: When marked, the component will appear in the list of user selectable and deselectable components in the Select Components page. You should turn this off for components that are either absolutely required by the program and for components that must be installed when other components are installed, but serve no purpose without those components. In the latter case, specify the appropriate Requires settings. If the user can select the component, you should provide a detailed description of what purpose it serves, so the user (who probably has absolutely no experience with your application), can make a descent decision about whether she wants it or not. Note that if the user clicks on Advanced Installation, he will see the non-user-selectable components too (but will not be able to select or deselect them), so even then a good description is welcome.

Requires: In this list, you can mark all the components that are required by the currently selected component. If the current component is selected to be installed, all the components it requires will be installed as well. The user cannot override this. For components that are required by other components, but serve no purpose on their own, you should turn off both "install by default" and "user selectable". This will make sure that the component is installed only when it is really needed.

See the Component Example for more information how to put these options to good use.

Folders

On the Windows system—like on most operating systems—files must be placed in folders. When you create a new component, DeployMaster will automatically add the standard folders to the fresh component. You can add files and shortcuts to these folders by selecting the folder and clicking on the appropriate button. You can also add subfolders to these folders. The folder items you see in the tree view on the files tab, are the folders as they will be created by your setup program on the user's system. These are not related to the file organization on your development system.

If you place any files in the %WINDOWS% and %SYSTEM% folders, DeployMaster will automatically consider all those files to be shared files. The installation program will update the reference count for those files in the registry. The uninstaller will also update the reference count and only uninstall the files if the reference count becomes zero, indicating that the file is no longer in use.

In the image above, you see a small sample. When I click on the Build button, DeployMaster will read the file `S:\JGsoftPub\EditPadLite7\Win32\EditPadLite7.exe` from my hard disk, compress it, and put it into the setup package I am building. Only the file name `EditPadLite7.exe` will be remembered. The information about where it was stored in my system is not retained. When the user installs the package, the file `EditPadLite7.exe` will be extracted and will be placed in `%APPFOLDER%\EditPadLite7.exe`, which will likely become `C:\Program Files\Just Great Software\EditPadLite7\EditPadLite7.exe` on the user's system. It does not matter where I keep my development files on my own system. This is also true for the shortcut, which will be placed in `%PROGRAMSMENU%`, probably being `C:\Users\UserName\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\EditPadLite7.lnk` on the user's system. DeployMaster is smart and notices that `S:\JGsoftPub\EditPadLite7\Win32\EditPadLite7.exe` on my system becomes `C:\Program Files\Just Great Software\EditPadLite7\EditPadLite7.exe` on the user's system. DeployMaster will make this shortcut point to exactly that file on the user's system.

On Windows Vista and later, normal users can only write to `C:\Users\TheirUserName\`. All other folders on the `C:` drive require administrator privileges in order to be written to. If your application uses data files that should be shared by all users on the same PC and that should be writable by all users then you should put those files under %USERDATA% and turn the option "**grant all users write access to this folder**" for that folder. This option tells your installer to change the security permissions on that folder so that all users can write to it. On the Project page you can specify a subfolder %COMMONDOCUMENTS% or %COMMONAPPDATAROOT% as the default user data folder. Alternatively, if you want to hard-code these folders, you can add %COMMONDOCUMENTS% or %COMMONAPPDATAROOT% directly under a component on the Files page and create subfolders under them on the Files page. You can then turn on "grant all users write access to this folder" for your application's subfolder.

Files

Copying files to the user's system is what an installation program is all about. So you will surely want to add some files to your package. Do this by selecting a folder in the tree on the files page, and then click the Add Files button. Select the files you want to add in the dialog box and they will be added to the selected folder. If you click on the Add Files button while you have a file or shortcut selected, the selected files will be placed in the same folder. Read the paragraph about folders to learn how DeployMaster will put the files on the user's system.

Note that you can add the same file more than once to your setup package. DeployMaster will notice this and store it only once in the setup package. If the same file was added to different folders, it will still be stored only once but may be extracted several times. This is also true for files that you have used on the Project page, the background image and the Identity DLL. You will probably want to copy the readme file and license agreement file to the user's system for future reference, in addition to having them show up in the installation program. They also will be stored only once in your installation package. If your identity DLL needs code that your application also uses, you may just as well add the identity functions to an application DLL and use that DLL as the Identity DLL in addition to installing it on the user's system where it will be used in your application's daily life.

For each file, you can also decide what should happen if the target file already exists on the user's system. You can choose to always overwrite, never overwrite or only overwrite if the file to be installed is newer than the one that already exists. In case of executable files (.exe, .dll, etc.), DeployMaster uses the versioning information stored inside such files to determine which file is more recent. For all other files, DeployMaster uses the file's modification date. You can also make this setting for all files in a folder by selecting the folder and making the setting for the folder.

If you select "never overwrite" for a file, then you can turn on the option to never uninstall that file. Select this option if DeployMaster should not, under any circumstances, delete that file. Even if the user uninstalls your software, the file will stay behind.

If the file is an .exe or .bat file, an extra option labeled "execute during installation with these parameters". When checked, Deploymaster will run the executable file or batch file after all files have been copied. The executable will be run with administrative privileges, except when a portable installation is being created. You can specify command line parameters to be passed to the executable. In most cases you will want to turn on "wait for completion" to make your installer wait for the executable to stop running before telling the user the installation has completed successfully. If you turn on "delete after execution", then the file will be deleted when it completes running. This option is intended to run self-registering libraries and other helper executables that must be run to perform extra installation tasks that DeployMaster cannot do on its own. You should not use this option to run the actual application you're installing. For that there's a separate option on the Finished page, which runs the application with normal user rights. You also should not use this option for 3rd party installers. Add those on the 3rd Party page instead.

If the file you are adding is a .dll or .ocx code library, an extra option labeled "register this DLL upon installation" will appear. When checked, DeployMaster will invoke `regsvr32` to register the DLL or OCX after installing it. For .tlb files, an option "register this library upon installation" will appear. When checked, DeployMaster will load and register the .tlb file after all files have been installed.

32-bit and 64-bit files

For each file you can select whether it is a 32-bit file or a 64-bit file. Tick both options for files that should be delivered with both the 32-bit and 64-bit versions of your application. If your application is 32-bit only or 64-bit only, simply leave both options turned on.

If you selected one of the two "32-bit application" options on the Platform page, then only files that have "32-bit file" checked are included in the installer that DeployMaster generates. All those files will be installed. If you selected one of the two "64-bit application" options on the Platform page, then only files that have "64-bit file" checked are included in the installer. All those files will be installed. If you add both the 32-bit and 64-bit files to the Files page and select the correct bitness for each file, then you can build separate 32-bit and 64-bit installers from a single DeployMaster setup script, simply by changing the target bitness on the Platform page.

If you selected the "32-bit and 64-bit application" option on the Platform page, then all files are included in the installer. If the user is running 32-bit Windows, the installer will only install files that have "32-bit file" checked. If the user is running 64-bit Windows, the installer will only install files that have "64-bit file" checked.

As the example in the screen shot shows, you can place two files with the same name in the same folder. Just make sure that only "32-bit file" is checked for one file and only "64-bit file" is checked for the other file.

Note that the "32-bit application for 32-bit Windows and 64-bit Windows" results in a pure 32-bit installer. Only 32-bit files will be installed, even if the user is running 64-bit Windows. If you want the installer to install different files depending on whether the user is running 32-bit or 64-bit Windows, then you have to select "32-bit and 64-bit application" on the Platform page.

Shortcuts

To make life easier for your users, you should place a few shortcuts, in strategic locations, to the files created by your installation package. It is probably a good idea to place a shortcut to your application in the %DESKTOP% folder, and to place shortcuts to all important files in the %APPMENU% folder. You can specify the default %APPMENU% folder on the Project page. %APPMENU% should be a subfolder of %PROGRAMSMENU%. Shortcuts that you place under the %PROGRAMSMENU% and %APPMENU% folders can be accessed via the Programs or All Programs item in the Start Menu in Windows 95 until Windows 7.

To add a shortcut, select a folder and click the Add Shortcuts button. Select the files you wish to create shortcuts to. As illustrated in the example in the folders section, DeployMaster will figure out where the file you selected on your system ends up on the user's system, and will set the shortcut's target appropriately. If that file is copied more than once to the user's system (a rare case), the you cannot be sure which copy the shortcut will point to. But since all the copies are identical, the user should not notice this.

If you are creating a shortcut to a file that is not being installed on the user's system, that shortcut will not be created. This ensures that there will be no broken shortcuts on the user's system. The situation could arise when the target file is in another component that the shortcut, and the shortcut's component is being installed and the file's component is not. You can take advantage of this fact and create one component containing all shortcuts (and only shortcuts) the user might need, and to put the files in other components. By making this shortcut component user selectable, you can please users who do not like software to mess with their nicely crafted start menu. If the user does want the shortcuts, he will only get those that correspond with files that have actually been installed.

If your installer targets both 32-bit and 64-bit Windows and includes separate 32-bit and 64-bit .exe files, then you'll need to add shortcuts to both the 32-bit and 64-bit .exe files too. This means that for each pair of 32-bit/64-bit .exe files, you'll have a pair of shortcuts pointing to that pair of .exe files. Since only one of the .exe files of each 32-bit/64-bit pair will be installed, only one of each pair of shortcuts will be created.

If the shortcut points to an executable file, the default icon is the first icon in the executable. You can immediately select a different icon from the same .exe file. For all other files, the default icon is whichever icon is used to indicate files of that type on the user's system. So if you are creating a shortcut to an HTML file, for example, the default icon will be the icon of the user's web browser. As a preview, DeployMaster shows the icon used on your system. That does not mean this icon is added to your installer.

If you want to select a different icon, click the Icon File button and then click Select. Choose the .exe, .dll, or .ico file that contains your icon. This file needs to be added as a file to the Files page under %APPFOLDER% or another folder appropriate for installing files. It does not need to be in the same folder as the shortcut or as the file that the shortcut points to.

Windows 8 Start Screen

Windows 8 introduced a new Start Screen that replaced the Start Menu that was used by Windows 95 until Windows 7. Shortcuts that you place under the %PROGRAMSMENU% and %APPMENU% folders can be accessed via the Windows 8 Start Screen by right-clicking on the Start Screen and then clicking "All apps" in the bottom right corner.

Shortcuts can also have their own tiles directly on the Windows 8 Start Screen, so the user can access them without having to go through "All apps". Such shortcuts are "pinned". When you select a shortcut on the Files tab and that shortcut is inside the %PROGRAMSMENU% or %APPMENU% folder, the Windows 8 option at the bottom of DeployMaster's window will become enabled. You have three choices that determine whether the shortcut will be pinned to the Start Screen:

1. **Let Windows 8 decide whether to pin the shortcut to the Start screen**: The default behavior is to pin a shortcut to the Start Screen if that shortcut points to an executable file, and the name of the executable doesn't look like it's an installer or uninstaller. The shortcut will be accessible through "All apps" too, even if the user decides to unpin it from the Start Screen.
2. **Do not pin the shortcut to the Start screen upon installation**: The shortcut will not be pinned to the Start Screen upon first installation. The user can access it via the "All apps" button and will have the ability to pin it manually.
3. **Pin the shortcut to the Start screen; delete it when the user unpins it**: The shortcut will be pinned to the Start Screen, regardless of the kind of file it points to. If the user decides to unpin it from the Start Screen, Windows will delete the shortcut and thus the shortcut will no longer be accessible through "All apps".

Note that these options only take effect the very first time the shortcut is created. Windows 8 may ignore your choice on subsequent installations, even if your application was uninstalled and reinstalled. If the user decides to pin or unpin a shortcut, your installer cannot override that choice. So if you want to test your installer with different Windows 8 pinning choices for your shortcuts, you have to perform each test on a copy of Windows on which your software was never installed. The best way to do this is with a clean install of Windows in a virtual machine that you can roll back to its clean state after each test.

Windows 8.1 Start Screen and Windows 10 Start Menu

These options do not have any effect on Windows 8.1 even though it uses a Start Screen very similar to Windows 8. They also have no effect on Windows 10, which has a Start Menu that combines the features of the traditional Start Menu from Windows 7 and prior and the live tiles from the Windows 8 Start Screen. On Windows 8.1 shortcuts your installer puts under %APPMENU% or %PROGRAMSMENU% can be accessed via All Apps in the Start Screen. On Windows 10 they appear directly in the Start Menu. The user can create a tile for your application on the Start Screen or Start Menu by right-clicking its icon under All Apps or its desktop icon and selecting Pin to Start.

If you are primarily targeting Windows 8.1 and later, then you should place a single shortcut to your application under %PROGRAMSMENU% on the Files page. Then your application will be listed directly under All Apps in alphabetic order with all the other applications. That's where people using Windows 8.1 and later will expect to find it. The %PROGRAMSMENU% folder puts shortcuts directly under All Apps and cannot be changed by the user. Then you won't use %APPMENU% and shouldn't specify a default for it on the Project page. However, if previous versions of your installer used %APPMENU%, then you should continue to place your shortcuts under %APPMENU%. You can change the default folder for %APPMENU% to %PROGRAMSMENU% without a subfolder on the Project page. You should also turn off the option to create a shortcut to the uninstaller on the Finished page.

Adding additional shortcuts to help files, web sites, uninstallers, and the like is no longer recommended. Applications are expected to put just one shortcut on the Start Menu. But if you want the additional shortcuts, you should put all your shortcuts under %APPMENU% on the Files page. Set the default start menu folder on the Project page to %PROGRAMSMENU%\Your App. On Windows 8.1 this will create a section Your App under All Apps on the Start Screen. On Windows 10 this will place a folder Your App in alphabetic order on the Start Menu. Clicking this folder expands it to show your shortcuts. The user can select a different folder for %APPMENU% via Advanced Installation.

Do not place shortcuts under both %PROGRAMSMENU% and %APPMENU%. Either place a single shortcut under %PROGRAMSMENU%, or place one or more shortcuts under %APPMENU%.

URLs

To create a shortcut to a web page, click the Add URL button. This will create a new item in the file tree where you can type in the URL. If you want to give the shortcut a descriptive name, rather than having it show the URL, type it into the Name field.

The default icon for URLs is the icon used by the user's default web browser. You can select another icon if you want. This icon can be extracted from any .ico, .exe or .dll file that will be installed by your installation package.

Fonts

To install fonts and make them available to all applications, simply add the `.ttf` and `.otf` files with TrueType and OpenType fonts or `.fon` files with bitmapped fonts to `%FONTS%` folder. Your installer will then automatically put your fonts into the Windows fonts folder and register the fonts so they'll be available to all applications. You should not put any other files into the %FONTS% folder. Installing fonts into %FONTS% requires administrator privileges.

If your application uses private fonts that you don't want to be available to other applications, simply put them under %APPFOLDER% with your application. Your application will then need to call the `AddFontResource()` Windows API function each time it runs to make the font available to itself.

Standard Folder Names

When you need to specify a folder, you should always remember that the structure of the user's file system may be quite different from yours. The standard folder names are also translated with localized versions of Windows. Therefore, you should always start a folder name with one of the standard folder names below. For example: install your documentation files into `%APPFOLDER%\Docs` and not into `C:\Program Files\MyApp\Docs`.

The following folders are the ones you will use to install most, if not all, files that are part of the package you are delivering. You can specify the defaults for them on the Project page. These folders can be modified by the user by clicking on the Advanced Installation button in the installation program.

%APPFOLDER%
Folder where your application will be installed
All users: %PROGRAMFILES%\CompanyName\AppName
Current user: %LOCALAPPDATAROOT%\CompanyName\AppName

%APPCOMMONFOLDER%
Folder where you should install any reusable libraries shared between multiple applications
All users: %COMMONFILES%\CompanyName
Current user: %LOCALAPPDATAROOT%\CompanyName

%APPMENU%
Folder where your application's shortcuts should be placed, so it can be accessed from the Start menu
All users and current user: %PROGRAMSMENU%\AppName

%USERDATA%
Folder for sample files and data files that will be modified by the user while using your application
All users: %COMMONAPPDATAROOT%\CompanyName\AppName
Current user: %APPDATAROOT%\CompanyName\AppName

The folders below are system folders. You can use them on the Project page for the default values of user-configurable folders. You can also use them on the Files page to force files to be installed into these system folders.

The actual paths shown in the list below are the usual paths for these folders on various versions of Windows. These paths are listed here for your reference only. DeployMaster will always detect and use the actual paths on the user's computer, which may differ from the ones below.

The appropriateness of using a folder for installations for all users and for user-specific installations is indicated as follows:

- All users: This folder can be used for installations for all users. These folders require admin rights to be written to.
- Current user: This folder can be used for installations for the current user. These folders do not require admin rights.
- Current user with admin rights: DeployMaster allows you to use this folder for installations for the current user even though they are not user-specific folders. These folders require admin rights to be written to.
- Adapts to all or current users: These placeholders point to different folders for installations for all users versus installations for the current user, making them appropriate for either type of installation.

If your installer creates subfolders under folders that need admin rights and you need those subfolders to be writable by non-admin users after installation, tick "grant all users write access to this folder" on the Files page for those subfolders.

Name	**Description and usual actual folder**	**Users and rights**
%PROGRAMFILES%	Applications folder Usually C:\Program Files\ C:\Program Files (x86)\ for 32-bit software on 64-bit Windows	All users Current user with admin rights
%COMMON FILES%	Applications shared libraries folder Usually C:\Program Files\Common Files C:\Program Files (x86)\Common Files\ for 32-bit software on 64-bit Windows	All users Current user with admin rights
%MYDOCUMENTS %	The My Documents folder for the user installing the application C:\My Documents on 98/ME C:\Documents and Settings\User\My Documents on NT/2000/XP C:\Users\User\Documents on Vista/7/8/10	Current user
%COMMON DOCUMENTS%	The folder for documents shared among users C:\My Documents on 98/ME C:\Documents and Settings\All Users\Documents on NT/2000/XP C:\Users\Public\Public Documents on Vista/7/8/10	All users
%APPDATAROOT%	Folder for storing application-specific and user-specific data such as preferences and state information that should be synchronized with the user's roaming profile on the Windows network C:\Windows\Application Data on 98/ME C:\Documents and Settings\User\Application Data on NT/2000/XP C:\Users\User\AppData\Roaming on Vista/7/8/10	Current user

%LOCAL APPDATAROOT%	Folder for storing application-specific and user-specific data such as preferences and state information that should be stored on the local PC only; can also be used as the applications folder for user-specific installations. C:\Windows\Application Data on 98/ME C:\Documents and Settings\User\Local Settings\Application Data on NT/2000/XP C:\Users\User\AppData\Local on Vista/7/8/10	Current user
%COMMON APPDATAROOT%	Folder for storing application-specific data shared among users C:\Windows\Application Data on 98/ME C:\Documents and Settings\All Users\Application Data on NT/2000/XP C:\ProgramData on Vista/7/8/10	All users
%STARTMENU%	The folder accessible through the Start button on the Taskbar C:\Windows\Start Menu\ on 98/ME C:\Documents and Settings\All Users\Start Menu on NT/2000/XP for all users C:\Documents and Settings\User\Start Menu on NT/2000/XP for the current user C:\ProgramData\Microsoft\Windows\Start Menu on Vista/7/8/10 for all users C:\Users\User\AppData\Roaming\Microsoft\Windows\Start Menu\ on Vista/7/8/10 for the current user	Adapts to all or current users
%PROGRAMSMENU%	The folder accessible through Start->Programs C:\Windows\Start Menu\Programs on 98/ME C:\Documents and Settings\All Users\Start Menu\Programs on NT/2000/XP for all users C:\Documents and Settings\User\Start Menu\Programs on NT/2000/XP for the current user C:\ProgramData\Microsoft\Windows\Start Menu\Programs on Vista/7/8/10 for all users C:\Users\User\AppData\Roaming\Microsoft\Windows\Start Menu\Programs on Vista/7/8/10 for the current user	Adapts to all or current users
%DESKTOP%	The desktop folder C:\Windows\Desktop on 98/ME C:\Documents and Settings\All Users\Desktop on NT/2000/XP for all users C:\Documents and Settings\User\Desktop on NT/2000/XP for the current user C:\Users\Public\Desktop on Vista/7/8/10 for all users C:\Users\User\Desktop on Vista/7/8/10 for the current user	Adapts to all or current users
%STARTUP%	Folder containing the shortcuts that are executed when the user logs in on his Windows system C:\Windows\Start Menu\Programs\StartUp on 98/ME C:\Documents and Settings\All Users\Start Menu\Programs\StartUp on NT/2000/XP for all users	Adapts to all or current users

	C:\Documents and Settings\User\Start Menu\Programs\StartUp on NT/2000/XP for the current user C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup on Vista/7/8/10 for all users C:\Users\User\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup on Vista/7/8/10 for the current user	
%SENDTO%	Folder containing Windows Explorer's Send To context menu shortcuts C:\Windows\SendTo on 98/ME C:\Documents and Settings\All Users\SendTo on NT/2000/XP C:\Users\User\AppData\Roaming\Microsoft\Windows\SendTo on Vista/7/8/10	All users on 98/ME/NT/2000/XP Current user on Vista/7/8/10
%WINDOWS%	The Windows folder Usually C:\Windows or C:\WINNT	All users Current user with admin rights
%FONTS%	The folder where fonts are installed You should only add `.ttf` and `.otf` files with TrueType and OpenType fonts or `.fon` files with bitmapped fonts to this folder. Usually C:\Windows\Fonts or C:\WINNT\Fonts	All users Current user with admin rights
%SYSTEM%	The Windows system folder Usually C:\Windows\System or C:\WINNT\System32 Redirected to C:\Windows\SYSWOW64 for 32-bit software on 64-bit Windows	All users Current user with admin rights
%SYSTEMDRIVE%	The drive where the Windows folder resides Usually C:\	All users Current user with admin rights

If you place any files in the %WINDOWS% and %SYSTEM% folders, DeployMaster will automatically consider all those files to be shared files. The installation program will update the reference count for those files in the registry. The uninstaller will also update the reference count and only uninstall the files if the reference count becomes zero, indicating that the file is no longer in use. You should not put any files into the Windows and System folders, unless absolutely needed.

Also, you should only place shortcuts in the Start Menu, Programs Menu, Desktop, Send To and Startup folders, even if nothing prevents you technically from putting files there. In a networked environment, the contents of these folders may be stored on a server. If they contain large files, the user's login may be significantly slowed down if the network is busy, while the user's profile including those files is being downloaded from the server.

Installing fonts requires extra steps if the font should be available to all applications. DeployMaster automatically does this for files that you place into the `%FONTS%` folder. DeployMaster will not register fonts that you place into other folders.

Component Example

The purpose of this example is to illustrate how you should use the various options for components to give your users the power to install only what they really want or need, but without asking too many questions.

Component name	Default	Selectable	Requires
Core files	Yes	No	-
Handy utility	Yes	Yes	Core files
Example support dll	No	No	Core files
Funny examples	Yes	Yes	Core files, Example support dll
Boring examples	No	Yes	Core files, Example support dll
Utility examples	Yes	Yes	Core files, Handy utility, Example support dll

Component Options Explained

Core files: This component is marked to be installed by default and not user selectable. Thus, it will be installed no matter what (unless the user aborts the installation before any files are copied). It is also marked as required by all other components. This has no effect since it is always installed, but you should mark it anyway so you remember its purpose whenever you need to update the installation package.

Handy utility: Marked as installed by default, since it is very likely that the user will want to install this component. However, it is user selectable so advanced users of your software who can live without this "handy utility", can conserve disk space. The extended description of this component should convince new users that they really want this.

Example support dll: Not marked as installed by default, since it serves no purpose if the user is not installing any example components. It is also not user selectable, since it is very clear when this component needs to be installed and when it should not and thus we will not bother the user with offering an option that is not really an option. Since it is marked as required with all three example components, it will be installed when one or more of these is selected by the user, overriding the default. If none of the example components is marked, only the default option will be left and the support dll will not be wasting any disk space. Note that you could also add the dll file to all three of the example components instead of making it a separate component. The user would not notice the difference (it will be placed in the setup package only once, so even the file size will not increase), but it does make it easier for the developer (you) to maintain the installation program.

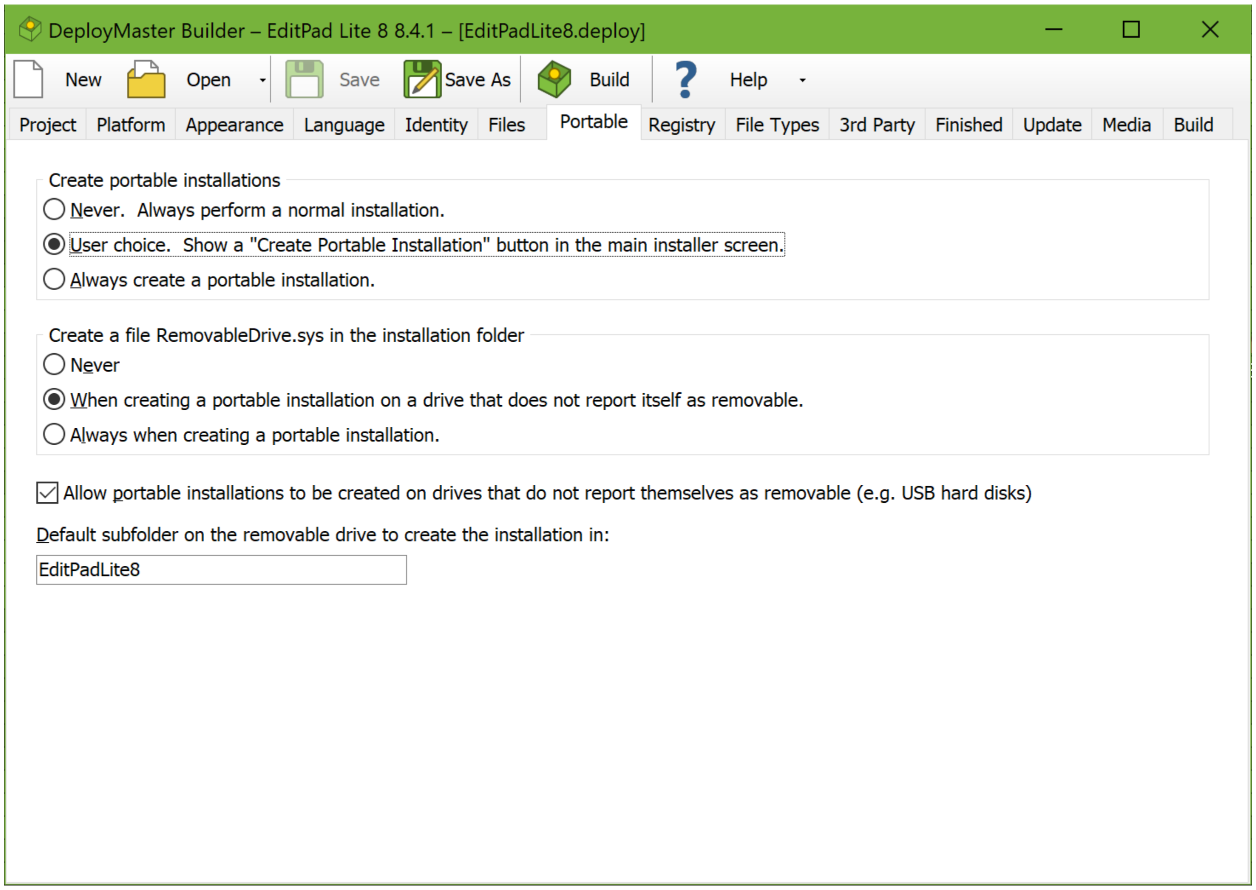
Funny examples: Installed by default, but can be deselected by advanced users or people without any sense of humor.

Boring examples: Not installed by default, but the user can select them if he really wants them.

Utility examples: Installed by default since the utility is also installed by default. Since it is marked as user selectable, the user can install the utility without the examples. However, since this component requires the utility component, the utility examples cannot be installed without the utility itself.

You see that even in this small artificial example, all four possible combinations of "install by default" and "user selectable" make the setup package nice and convenient. The user is not confronted with core files and support DLLs (unless she takes the Advanced Installation route), which would be difficult to explain if she is new to or does not really care about computers. But still she can select and deselect whatever she desires that is not essential to the application.

8. Portable



Create portable installations
- ◯ Never. Always perform a normal installation.
- ● User choice. Show a "Create Portable Installation" button in the main installer screen.
- ◯ Always create a portable installation.

Create a file RemovableDrive.sys in the installation folder
- ◯ Never
- ● When creating a portable installation on a drive that does not report itself as removable.
- ◯ Always when creating a portable installation.

☑ Allow portable installations to be created on drives that do not report themselves as removable (e.g. USB hard disks)

Default subfolder on the removable drive to create the installation in: EditPadLite8

A portable installation means that the setup program will copy all files the application needs on a portable device like a flash memory card or USB memory stick. No changes can be made to the host computer. This means that the Identity, Registry and File Types and settings are automatically ignored for portable installations. Portable installations also cannot depend on 3rd party installations. If you've added anything to the 3rd Party tab, DeployMaster will not show the Create Portable Installation button in the setup program.

Because portable installations make no changes to the host computer, they do not require administrator privileges like regular installs. Any user, even restricted users that normally can't install software, can use the Create Portable Installation button in your installer to copy its files into any folder that they can write to. This can be a USB stick but also a personal folder on a hard disk.

Of course, for a portable installation to really be portable, your application must be able to run in a portable manner. It must be able to detect that it was installed portably. If so, it must save all its settings onto the drive it is being run from, and still find that information when the drive letter changes (e.g. when the device is plugged into another computer). Saving everything in the same folder as your .exe file is an easy solution. (While this is a big no-no when your application is installed into c:\Program Files\, it's fine when installed on a personal storage device.) Your application must also keep its hands off the registry and the user's desktop. So it cannot create any shortcut icons or file associations.

If your application is ready to be portable, you can select the "user choice" or "always" option on the Portable tab in DeployMaster. In case of user choice, the Immediate Installation button and Advanced

Installation button in the setup's welcome screen will do a regular hard disk installation, creating all shortcuts, file associations, etc. The Custom Installation button will be replaced by the Create Portable Installation button. This button will copy your files onto the removable device, without doing anything else. Removing one button prevents button overload, while still allowing component selection through Advanced Installation.

DeployMaster can make it easy for your application to detect that it should act in a portable manner by creating a file RemovableDrive.sys in the installation folder on the removable device. Your application can then check for the existence of this file whether it should act as if it was installed on something portable, or on the computer's internal hard disk. Choose "never" if you don't want this file, or "always" to always create it for portable installations. You can use the middle option to make DeployMaster only create the file when you allow portable installations on hard disks or other devices that don't report themselves as removable.

Certain devices, like USB hard disks, do not report themselves as removable when queried by the GetDriveType() Win32 API call, even though they physically are. By default, the setup program will list only those drives that report themselves as removable via GetDriveType(). If you turn on the option to allow portable installations on any drive, the setup program will show a checkbox to allow the user to choose any drive to install on, including the C: drive and even mapped network drives. If you choose to allow this, and your application can either be installed portably or a traditionally into c:\Program Files, you should turn on the RemovableDrive.sys option so your application can determine which choice the user made. Some users may want to have a portable installation on an internal hard disk, e.g. to use the same installation in multiple versions of Windows on a multi-boot system, or to keep the installation easily synchronized or backed up using folder synchronization software.

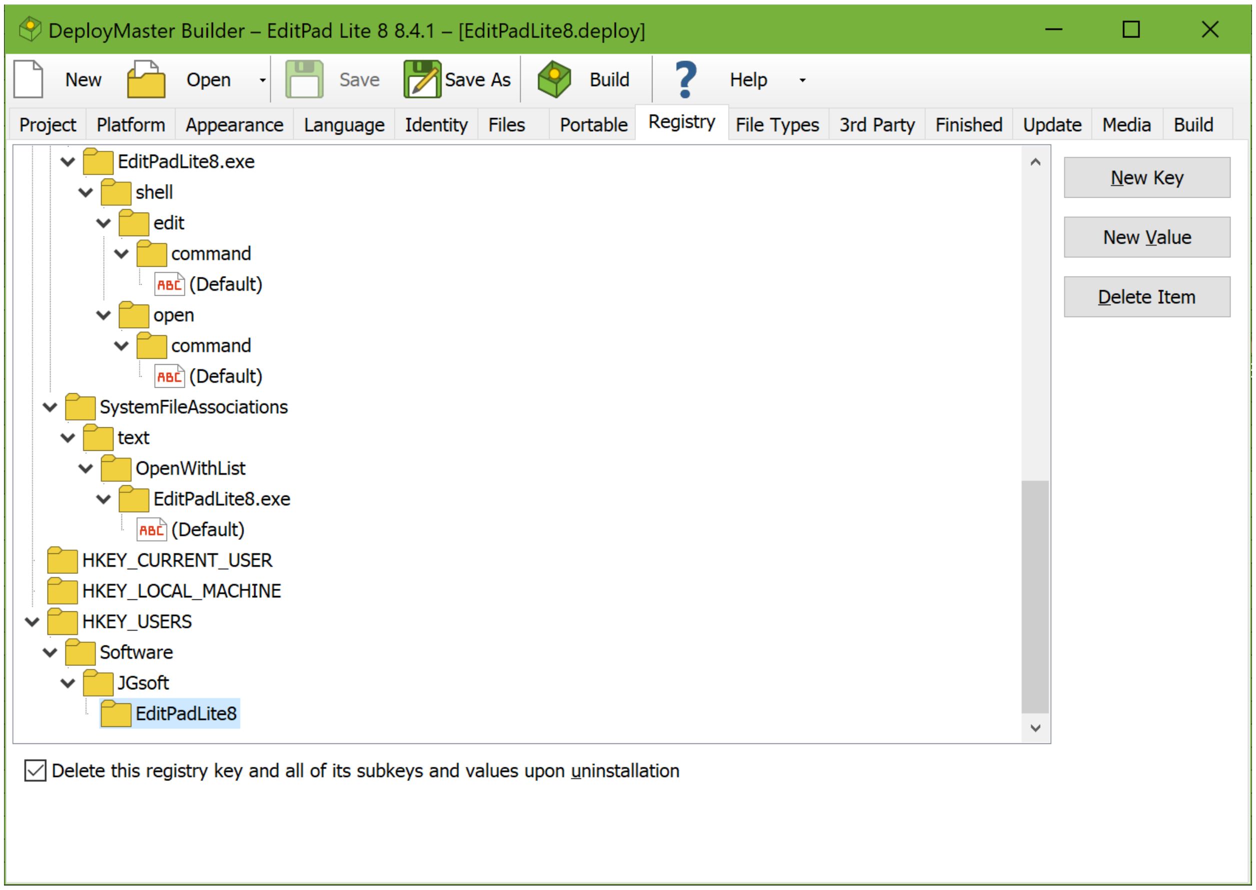
Finally, you can specify a default installation folder. Portable installations ignore the folder settings on the Project tab. They also ignore all folder structures on the Files tab, except for %APPFOLDER%. The %APPFOLDER% folder structure is created under the installation folder on the removable device. All other files are copied directly into that installation folder. The installation folder is created directly in the drive's root folder. If you don't set an installation folder, then the files are installed directly into the root of the device. The installation folder setting is only a default. The user can specify any folder. You cannot disable this. (Nor should you want to. Some users are funny about folder structures on their personal storage devices.)

Because the host computer must remain untouched, there will be no uninstall entry in the control panel for portable applications. The only way to uninstall will be for the user to delete your application's file from his storage device. A clearly labeled installation folder makes this easy.

Because the host computer does remain untouched, portable installations do not require administrator rights. The setup.exe generated by DeployMaster will not request elevation or trigger any security prompts when the user clicks the Create Portable Installation button.

9. Registry

If you want certain registry keys to be created by the installation program, add them to the registry page.



```
EditPadLite8.exe
  shell
    edit
      command
        (Default)
    open
      command
        (Default)
SystemFileAssociations
  text
    OpenWithList
      EditPadLite8.exe
        (Default)
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
  Software
    JGsoft
      EditPadLite8
```

☑ Delete this registry key and all of its subkeys and values upon uninstallation

To add a registry key, select its parent key, click the New Key button and enter the new key's name. To add a registry value, select its parent key or an existing value inside that key, click the New Value button, and enter the value's name. To give a key a default value, add a value under the key named (Default) including the parentheses. You can rename a key or value by clicking on it in the tree, waiting a second, and then clicking again.

For each value, you can choose whether it stores a string, a dword, or bytes. Then you can enter the actual value. If the value already exists in the registry, it is overwritten with the new value unless you turn on "keep existing value".

In addition to adding literal string, dword and binary values, you can also dynamically store information about the installation. In particular, you can store paths to folders and file names so your application knows where it was installed. To do so, simply create a string value, and make the value start with one of the standard folder names that you can use on the Files page. To store the path to myprog.exe in the installation folder, for example, create a string value of %APPFOLDER%\myprog.exe

The Registry page supports two additional folders that you cannot use on the Files page. These are %SETUPFOLDER% and %SETUPDRIVE%. You can use these values to store the folder and the drive that the setup.exe was on at the time of installation. You can use these registry values to pass the installation source to your application, such as to access additional files that are not part of the setup.exe, but are put on the same CD.

For every third-level and deeper registry key, you can specify that it must be completely **deleted upon uninstallation**. When your application is uninstalled, the key and all of its values and subkeys are deleted, whether they were added by DeployMaster or not. An ever-increasing Windows registry can noticeably slow down the operating system, so your users will appreciate it if your software really cleans up after itself and also removes registry items.

The uninstaller also removes the individual registry values that you have added on the Registry page, even if you did not activate the option to have the value's parent key deleted upon uninstallation. If you want to have the uninstaller remove specific values, you can add "dummy" values to the Registry page. These values are not be created or modified upon installation, but are removed upon uninstallation. If you turn on "keep existing value" and the value already exists upon installation, regardless of which kind of value it is, then that value is not removed by the uninstaller.

You can create registry keys under **HKEY_LOCAL_MACHINE** if you turned off "install for current user" or if you turned on "require admin rights" on the Project page. HKEY_LOCAL_MACHINE cannot be written to without admin rights. You can create keys under **HKEY_CURRENT_USER** only if you turned off "install for all users" on the Project page as user-specific registry keys are not appropriate for all user installs.

If you turn on both "install for all users" and "install for current user" and you turn off "require admin rights" then you cannot place any keys under HKEY_LOCAL_MACHINE or HKEY_CURRENT_USER per the above rules. In that case you can use **HKEY_AUTO**. This is not an actual registry key but a placeholder in DeployMaster. If the user chooses to install for all users, keys you put under HKEY_AUTO are created under HKEY_LOCAL_MACHINE. If the user chooses a user-specific installation then those keys are created under HKEY_CURRENT_USER. When your application wants to read the value, it should first look under HKEY_CURRENT_USER. If the value doesn't exist there it should look under HKEY_LOCAL_MACHINE. You can use HKEY_AUTO even if you don't allow the user to choose between an all-users and a user-specific installation.

Keys you put under **HKEY_CLASSES_ROOT** are created under HKEY_CLASSES_ROOT during installations for all users. But those keys are put under HKEY_CURRENT_USER\Software\Classes during user-specific installations.

Keys you put under **HKEY_USERS** are created under HKEY_CURRENT_USER during a user-specific installation. During an installation for all users, keys under HKEY_USERS are created for all users so that your application will see them when querying HKEY_CURRENT_USER regardless of the user account it's running under. Normally, it's better if your installer creates keys that all users need to see under HKEY_LOCAL_MACHINE. But if your application creates keys under HKEY_CURRENT_USER to store user-specific settings, then you can add those keys under HKEY_USERS to schedule them to be deleted upon uninstallation. If the application was installed for all users, then the uninstaller deletes keys under HKEY_USERS for all users. If it was installed for the current user only, then the uninstaller only cleans up keys you put under HKEY_USERS for the current user.

When checking whether your installer correctly created your registry values, you need to remember that HKEY_LOCAL_MACHINE\Software is segregated into **32-bit and 64-bit branches**. Your installer will use the correct branch as long as you select the correct bitness on the Platform page. This comes into play when installing 32-bit software on 64-bit Windows. DeployMaster writes the values so that 32-bit software sees them under HKEY_LOCAL_MACHINE\Software. It's best to use a 32-bit test program if you need to verify registry values that need to be visible to a 32-bit application. If you check with regedit.exe, which is a 64-bit application on 64-bit Windows, it won't show the values under HKEY_LOCAL_MACHINE\Software because it is showing you the 64-bit branch. 64-bit RegEdit shows the 32-bit branch under HKEY_LOCAL_MACHINE\Software\Wow6432Node. Your application should never access that node directly.

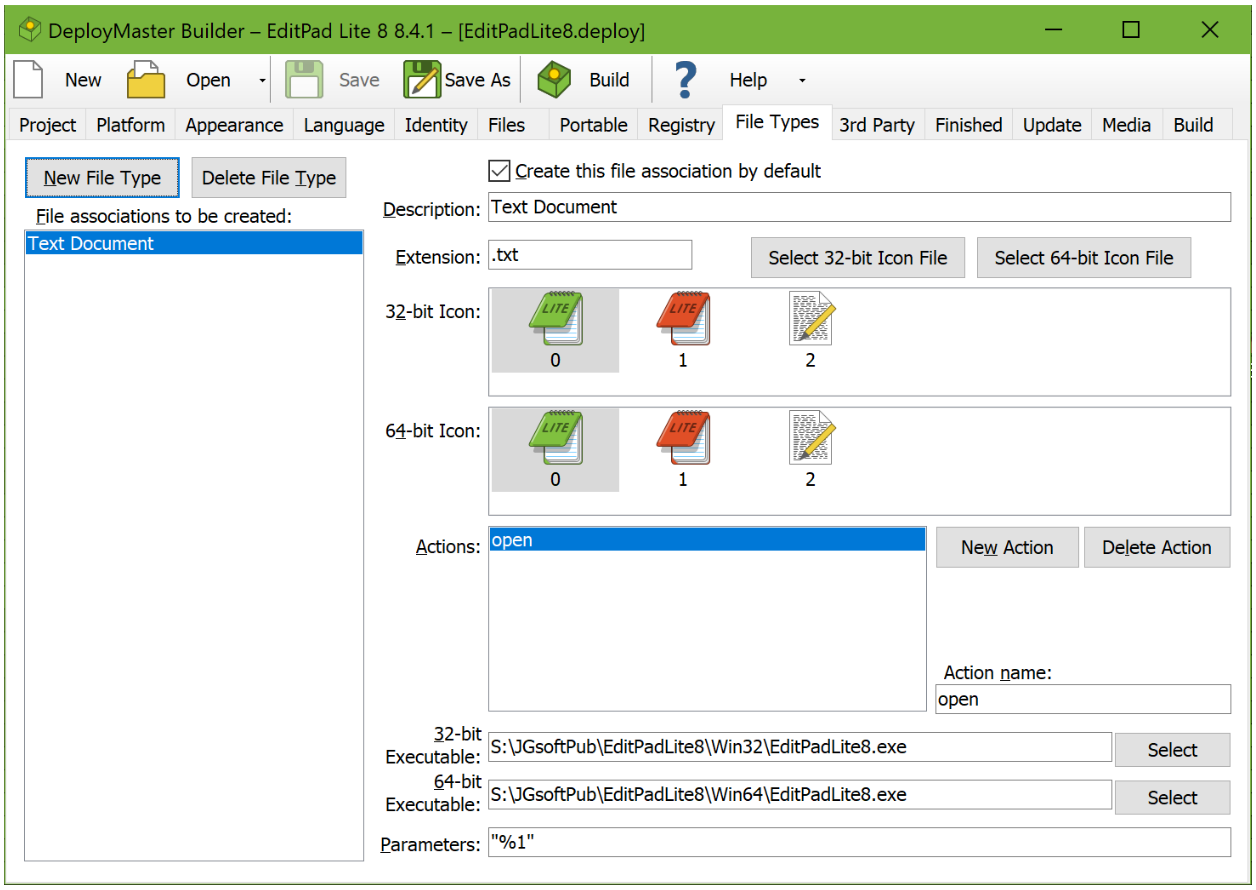
DeployMaster also creates several other registry keys and values for its own purposes:

- It adds or changes keys and values under HKEY_CLASSES_ROOT to register the file types your software uses.
- Your software is added to the Add/Remove Programs applet in the Control Panel, so the user can uninstall it from there.
- Under HKEY_AUTO\Software\JGsoft\DeployIT, it creates a string value indicating the deployment log file, using the application name specified under Project as the value's name. This value is used by DeployMaster to see if your application has already been installed.

This information is used by DeployMaster to be able to update or uninstall your software. All these keys and values are, of course, automatically removed when your software is uninstalled.

10. File Types

On this page, you can associate various file extensions with your software.



File associations to be created: Text Document

☑ Create this file association by default

Description: Text Document

Extension: .txt

32-bit Icon: 0 1 2

64-bit Icon: 0 1 2

Actions: open

Action name: open

32-bit Executable: S:\JGsoftPub\EditPadLite8\Win32\EditPadLite8.exe

64-bit Executable: S:\JGsoftPub\EditPadLite8\Win64\EditPadLite8.exe

Parameters: "%1"

The option "create file association by default", determines whether the file association will be created or not, if the user does not use the Advanced Installation button when installing your software. You should always keep this checkbox marked, except when you want to associate your application with standard extensions such as `.txt`, `.html`, etc. Many people do not want any software to take control of standard file extensions.

If the user selects Advanced Installation when running your setup, he is given the chance to prevent your setup from creating some or all file associations. Some users really do not like software messing with their file associations.

Each file association requires a description, which will appear in the Type column in Windows Explorer. Of course, an extension also needed. It should be specified in the form of `.ext`.

The icon for the file association must be stored in a file that you've added to the Files page in DeployMaster. If this file is an `.ico` file, use the Select 32-Bit Icon File button as well as the Select 64-Bit Icon File Button to select this `.ico` file twice. On the Files page, both the "32-bit file" and "64-bit file" checkboxes should be checked for `.ico` files. Icon files do not have any bitness.

If the icon for the file association is stored in an executable file or dynamic link library, and you have selected one of the two "32-bit application" options on the Platform page in DeployMaster, use the Select 32-bit Icon File button to select the `.exe` or `.dll`. Make sure that "32-bit file" is checked for this file on the Files page. For 32-bit applications, the file association will use the 32-bit icon even when installing on 64-bit Windows. If you have selected one of the two "64-bit application" options on the Platform page, use the Select 64-bit Icon File button to select the `.exe` or `.dll`. Make sure that "64-bit file" is checked for this file on the Files page. The 32-bit icon setting is not used by 64-bit installers. If you have selected "32-bit and 64-bit application" on the Platform page, then you need to use the Select 32-bit Icon File button and pick an `.exe` or `.dll` file for which you have ticked "32-bit file" on the Files page. You also need to use the 64-bit Icon File Button to pick a file for which you have ticked "64-bit file" on the Files page.

After selecting the file(s) containing the icon you want to use, the "32-bit icon" and/or "64-bit icon" lists will show the icons stored in the selected file(s). If more than one icon is shown, click on the icon that the file association should use.

File Type Actions

To make the file association work, you will need to add one or more actions for this file type. One of the actions should be named "Open". This action will be executed when the user double-clicks on a file of this type in Windows Explorer. This action should do whatever is most common for this type of file. A file association for HTML files would show the file in a web browser. Another common action is "Edit". This action should allow the user to edit the file. Add this action if the "open" action does not allow the file to be edited. Since web browsers don't allow HTML files to be edited, an HTML file association could use the "edit" action to open the HTML file in a text editor.

You are not restricted to these standard actions. You can add as many actions specific to your file type as you like. If your files often need to be transmogrified, add a new action and enter Transmogrify as the name. You should capitalize the name's first letter, so it'll fit nicely with the other items in the right-click menu in Windows Explorer. That's where all of your file type's actions will appear.

For each action, you must select an executable file that will perform the action. This executable must be one that is installed by your setup package, so it must have been added to the Files page. If the user deselected the component that contains the executable (and thus it is not installed on the user's system), DeployMaster will not create this action. If all of a file type's actions should be handled by an executable that is not being installed, DeployMaster will not create that file type altogether. This ensures that there will be no broken file associations on the user's system created by your software product. This also means that you can safely add any file types on this page that are used only by special components of your software. If the component is not installed, its file types aren't either.

If you've selected one of the two "32-bit application" options on the Platform page in DeployMaster, then you only need to specify a 32-bit executable for the action. Make sure that "32-bit file" is checked for this file on the Files page. For 32-bit applications, the action will use the 32-bit executable even when installing on 64-bit Windows. If you have selected one of the two "64-bit application" options on the Platform page, then you only need to specify a 64-bit executable for the action. Make sure that "64-bit file" is checked for this file on the Files page. If you have selected "32-bit and 64-bit application" on the Platform page, then you need to specify both a 32-bit executable and a 64-bit executable for the action.

For each action, you can also specify a parameter list that will be passed on the executable's command line. Even if you do not seem to have a need for any parameters (very likely with an "open" action), it is a very good idea to specify `"%1"` (including the quotes!) in the Parameters field. This parameter will cause quotes to be placed around the file the action should be performed upon, preventing files with spaces in their names from being broken into pieces.

Uninstalling File Associations

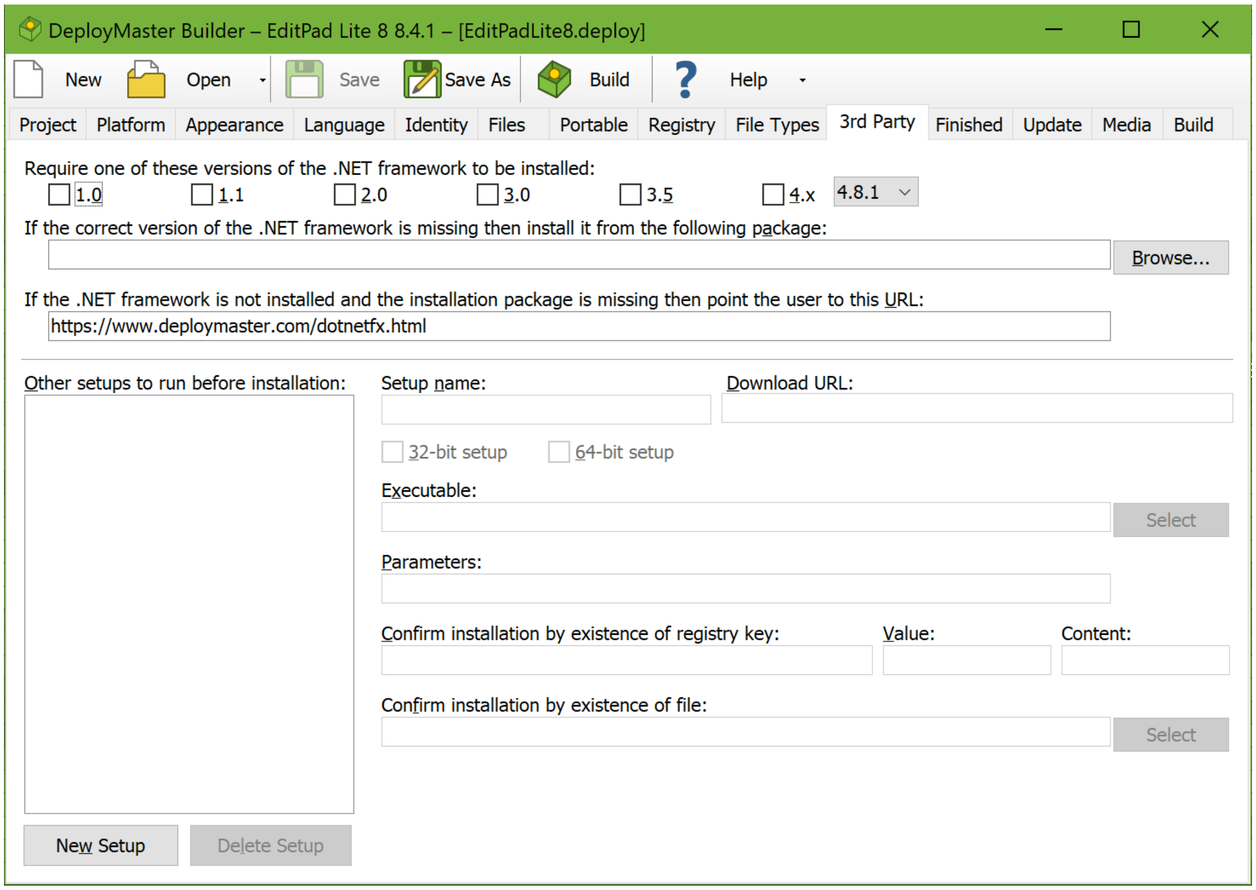
DeployMaster's uninstaller will automatically remove all file associations were created by the installer. If the installer modified existing file associations, the uninstaller will revert them to their previous state.

Before removing or reverting a file association, the uninstaller checks whether the file association is still what your installer put into place. If the file association was modified after your application was installed, then the uninstaller will leave it alone. The uninstaller won't break any file associations that were taken over by other software.

This also means that if your software creates or modifies file associations after it was installed, then your software will be responsible for removing those file associations. You can do this via the "program to execute before starting uninstallation" option on the Finished page.

11. 3rd Party

If your application depends on certain libraries that are more easily installed via their own setup programs, you can specify these on the 3rd Party tab in DeployMaster.



Require one of these versions of the .NET framework to be installed:
☐ 1.0 ☐ 1.1 ☐ 2.0 ☐ 3.0 ☐ 3.5 ☐ 4.x 4.8.1

If the correct version of the .NET framework is missing then install it from the following package:

If the .NET framework is not installed and the installation package is missing then point the user to this URL:
https://www.deploymaster.com/dotnetfx.html

Other setups to run before installation:

Setup name:

Download URL:

☐ 32-bit setup ☐ 64-bit setup

Executable:

Parameters:

Confirm installation by existence of registry key:

Value:

Content:

Confirm installation by existence of file:

One of the most common libraries with a complex installation is the **Microsoft .NET framework**. DeployMaster has special support for .NET versions 1.0 through 4.8.1. Depending on how you built your .NET application, it will either require a specific version of the .NET framework, or be compatible with multiple versions. Select all the versions that you have successfully tested your application with.

Versions 1.0 through 3.5 of the .NET framework are installed side by side. It is possible for a PC to have some or all of these versions installed at the same time. For these versions DeployMaster provides checkboxes that you can use to indicate which versions of the framework your application is compatible with. An application that is compatible with a .NET release between 1.0 and 3.5 is not necessarily compatible with later versions of the .NET framework.

Installing .NET 4.x replaces any previous installation of .NET 4.x, while installing side by side with older versions. If your application requires .NET 4.x you should tick the 4.x checkbox and then select whether the minimum version required by your application is 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, or 4.8.1. An application that is compatible with a 4.x release of .NET is also compatible with all later 4.x releases.

Windows Vista and later come preinstalled with a specific version of .NET. Recent versions of .NET only work on recent versions of Windows. If you specify that your application requires .NET then DeployMaster will force you to deselect all versions of Windows not supported by that version of .NET on the Platform page. In the table below, “preinstalled on” means that that specific version of .NET is preinstalled on certain versions of Windows. “Supported on” means that your application will run because that version of Windows either allows you to install that version of .NET or because that version of Windows includes a version of .NET that is more recent than and compatible with your chosen version of .NET.

.NET Version	Preinstalled on	Supported on
1.0	none	Windows 98, ME, NT4, 2000, XP, Vista, and 7.
1.1	none	Windows 98, ME, NT4, 2000, XP, Vista, 7, 8, 8.1, 10, and 11
2.0	none	Windows 98, ME, NT4, 2000, XP, Vista, 7, 8, 8.1, 10, and 11
3.0	Windows Vista	Windows XP, Vista, 7, 8, 8.1, 10, and 11
3.5	Windows 7, 8, 8.1, 10, and 11	Windows XP, Vista, 7, 8, 8.1, 10, and 11
4.0	none	Windows XP, Vista, 7, 8, 8.1, 10, and 11
4.5	Windows 8	Windows Vista, 7, 8, 8.1, 10, and 11
4.5.1	Windows 8.1	Windows Vista, 7, 8, 8.1, 10, and 11
4.5.2	none	Windows Vista, 7, 8, 8.1, 10, and 11
4.6	Windows 10 1507 (original release)	Windows Vista, 7, 8, 8.1, 10, and 11
4.6.1	Windows 10 1511 (November update)	Windows 7, 8, 8.1, 10, and 11
4.6.2	Windows 10 1607 (Anniversary update)	Windows 7, 8, 8.1, 10, and 11
4.7	Windows 10 1703 (Creators update)	Windows 7, 8.1, 10, and 11
4.7.1	Windows 10 1709 (Fall Creators update)	Windows 7, 8.1, 10 (1607 and later), and 11
4.7.2	Windows 10 1803 (April 2018 update) Windows 10 1809 (October 2018 Update)	Windows 7, 8.1, 10 (1607 and later), and 11
4.8	Windows 10 1903 (May 2019 update) and all later updates Windows 11 21H2 (original release)	Windows 7, 8.1, 10 (1607 and later), and 11
4.8.1	Windows 11 22H2 (2022 update)	Windows 10 20H2 (October 2020 Update) and later

If you want your setup to **automatically install the .NET framework** when needed, click the Browse button to select the .NET installer on your computer. You can download the .NET installers from Microsoft. You can find links to the Microsoft downloads at http://www.deploymaster.com/dotnetfx.html. Obviously, you’ll need to select the installer of a .NET version that your application is actually compatible with. Otherwise, your setup will be stuck at the .NET prerequisite. On the Media page you can control whether DeployMaster should integrate the .NET installer into your setup.exe, or look for it as a separate file.

For .NET 1.0, 1.1 and 2.0, the .NET installers are complete packages, about 25 MB in size. For versions 3.0 and 3.5, dotnetfx30setup.exe and dotnetfx35setup.exe are small 3 MB installer stubs that will download the actual framework from Microsoft. There are also two huge dotnetfx30.exe and dotnetfx35.exe packages that include the whole framework. You should use the former if your setup.exe will be a download. You can use the latter if you’ll distribute your software on CD which has the 100 to 200 MB of space needed for the framework. For .NET 4.x the installers are much smaller. The web installers are only about 1 MB, while the full installers are between 48 MB to 67 MB depending on the version. Installing any of the .NET 4.x versions replaces the previous .NET 4.x installation. Therefore, if your application requires .NET 4.x and you want to bundle the installer, we recommend you bundle the latest version of .NET that is supported on the versions of Windows that your application supports even if your application does not require the latest version of .NET.

You should always **specify a URL** for the user to download the .NET framework, even if you bundle it with your setup.exe. If you use the default URL to DeployMaster's web site, the setup program will automatically add a version parameter to the URL to make the page display only the latest version of the .NET framework that you've indicated your application is compatible with. DeployMaster will display this URL when it can't find the .NET installer or when you don't specify one. The user can click the URL to open it in their web browser. If you provide your own page, make sure to explain the user what the framework is and why it's needed.

Other Setups to Run Before Installation

If your application needs other 3rd party packages to be installed, click on the New Setup button. Then specify its details.

Name: Name of the 3rd party item. This will be displayed to the user when your setup program tells the user to download it, or while the setup program waits for the 3rd party installer.

Download URL: You should always specify a download URL. You can link directly to the file if you like. However, linking to a page with a bit of explanation and a link to the actual setup will likely provide a better user experience.

Executable: The path to the file with the actual installation program as it appears on your computer. On the Media tab you can control whether DeployMaster should bundle the executable into your setup.exe, or look for it as a separate file. When integrating it into your own setup, the executable must be entirely self-contained. Your setup.exe will only extract and execute this one file. When copying it along with your setup.exe, you can copy along other files that the executable needs. The executable can be an .exe file or an .msi Windows installer package.

Parameters: If the executable is an .exe file, you can specify command line parameters. If it is an .msi file, you should leave the Parameters field blank.

Confirm installation by registry: There are two ways in which DeployMaster can check whether the package is already installed or not. You can specify a registry key such as HKEY_LOCAL_MACHINE\Software\Company\Package. If the key exists, the package is assumed to have been installed. If you also specify a Value name, the value must exist under the specified key for the package to be assumed installed. If you also specify a Content string, then the value under the key must have this specific string or dword value for the package to be assumed installed.

Confirm installation by file: The other way is to specify the path to a file that must exist on the user's computer. You should use folder placeholders like %PROGRAMFILES% and %SYSTEM%. If the file exists, the package is assumed to have been installed. Note that this option is only reliable if your 3rd party installer always puts a certain file in the exact same location.

If you tell DeployMaster how to confirm the installation, your setup will not proceed until the requirement has been satisfied. If the requirement was already satisfied when your setup started, then the 3rd party installer is never executed.

If you do not provide a confirmation ability, the 3rd party installer will always be executed. When the 3rd party installer finishes, your setup will proceed, even if the user canceled the 3rd party installer. This behavior can be desirable if the 3rd party installer runs silently (i.e. it can't be canceled), and does its own check to see if it was already installed.

DeployMaster does not provide for the uninstallation of 3rd party packages. The user will have to uninstall those via the control panel, or whatever uninstall means the 3rd party package provides.

32-bit and 64-bit 3rd party installers

For each 3rd party installer you can select whether it is a 32-bit file or a 64-bit installer. Tick both options for installers that support both 32-bit and 64-bit Windows.

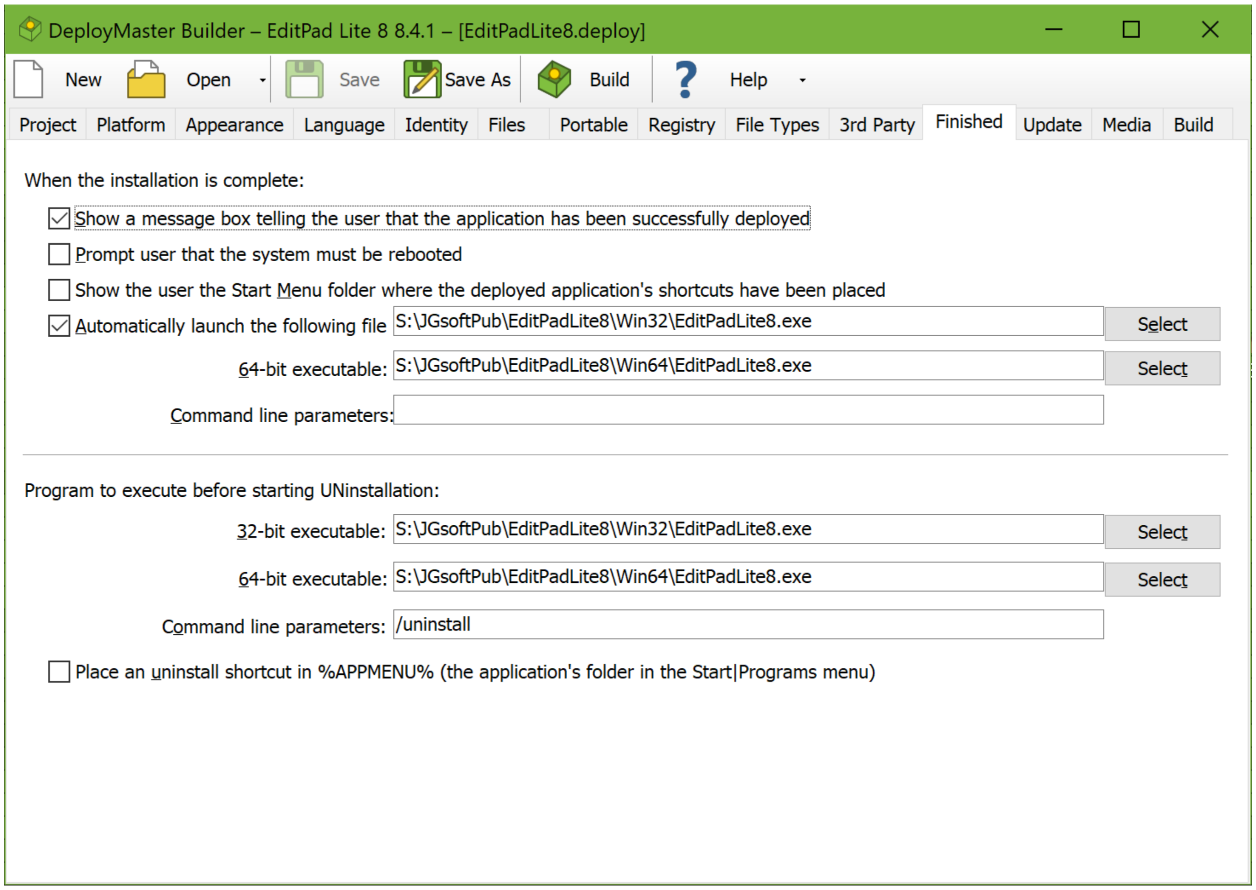
If you selected one of the two "32-bit application" options on the Platform page, then only 3rd party installers that have "32-bit file" checked are included in the setup that DeployMaster generates. All those installers will be run. If you selected one of the two "64-bit application" options on the Platform page, then only 3rd party installers that have "64-bit file" checked are included in the setup. All those installers will be run. If you add both the 32-bit and 64-bit installers to the 3rd Party page and select the correct bitness for each file, then you can build separate 32-bit and 64-bit setups from a single DeployMaster setup script, simply by changing the target bitness on the Platform page.

If you selected the "32-bit and 64-bit application" option on the Platform page, then all 3rd party installers are included in the setup that DeployMaster generates. If the user is running 32-bit Windows, the setup will only execute 3rd party installers that have "32-bit file" checked. If the user is running 64-bit Windows, the setup will only execute 3rd party installers that have "64-bit file" checked.

Note that the "32-bit application for 32-bit Windows and 64-bit Windows" results in a pure 32-bit setup. Only 32-bit 3rd party installers will be executed, even if the user is running 64-bit Windows. If you want the installer to run different 3rd party installers depending on whether the user is running 32-bit or 64-bit Windows, then you have to select "32-bit and 64-bit application" on the Platform page.

12. Finished

On the Finish page you can specify what happens when the deployment of your package is complete. You can also specify if an application should be run before your package is uninstalled.



When the installation is complete:

- [✓] Show a message box telling the user that the application has been successfully deployed
- [ ] Prompt user that the system must be rebooted
- [ ] Show the user the Start Menu folder where the deployed application's shortcuts have been placed
- [✓] Automatically launch the following file: S:\JGsoftPub\EditPadLite8\Win32\EditPadLite8.exe

64-bit executable: S:\JGsoftPub\EditPadLite8\Win64\EditPadLite8.exe

Command line parameters:

Program to execute before starting UNinstallation:

32-bit executable: S:\JGsoftPub\EditPadLite8\Win32\EditPadLite8.exe

64-bit executable: S:\JGsoftPub\EditPadLite8\Win64\EditPadLite8.exe

Command line parameters: /uninstall

- [ ] Place an uninstall shortcut in %APPMENU% (the application's folder in the Start|Programs menu)

These actions can be performed by the setup program after all files have been copied, all executables on the Files page have been run, all DLLs on the Files page have been registered, and the FinishDeployment() routine in the support DLL (if any) has been called:

Show a message box: The installer's window will remain visible when the installation has completed. It will show a message and a "Thanks" button that you can customize on the Language page.

Prompt for system reboot: The installer's window will remain visible when the installation has completed. It will show a message informing the user that the system should be rebooted to finish off the installation. It will offer the user the choice between automatically rebooting or just closing down the setup program, requiring the user to reboot manually later on. Note that even when this option is not selected, DeployMaster will always prompt to reboot when one of the files it had to install could not be copied to its final destination because another copy was already there and in use by the system (and thus DeployMaster could not get write access). DeployMaster will make sure that the file is properly installed upon system reboot. This means that you should only select this option if your `FinishDeployment()` routine or any of the 3rd party setups perform an action that requires the system to be rebooted (which DeployMaster cannot detect).

Show Start Menu folder: Open the folder under Start Menu|Programs where the icons for your application have been placed. The user will then be able to launch one of the shortcuts right away. This option is useful if your package consists of several modules and the user may want to start using any one of them.

Automatically launch file: Automatically execute a certain file, usually the application's main executable. This file must also have been added to the Files page. You should type in the full path and filename of the file on your system. DeployMaster will figure out where the file ends up on the user's system, and adjust the path accordingly. You can also specify any parameters that should be passed to the executable on the command line. If you turn on both this option and the option to show a message box, then the file is only launched if the user clicks the Thanks button. If the user closes the installer with the X button, then the executable is not run. The file is also not launched when the installer is run silently.

You should use the launch option on the Finished page only for launching your actual application after the installation is 100% complete. If you need to run executables to complete the installation, specify that on the Files page or on the 3rd party page.

If the user starts your installer normally by double-clicking it in Windows Explorer (as opposed to right-clicking and selecting Run as Administrator) then your installer will launch the installed application without administrator rights even if the installer requested administrator rights for itself during the installation. This ensures that your application is run exactly the same way by the installer as it would be when the user double-clicks its desktop shortcut. Many installers incorrectly run the launched application with admin rights, which leads to problems with user-specific settings when users have separate admin and non-admin accounts.

Program to execute before starting UNinstallation: If your application makes modifications to the user's system (file associations, new registry keys, etc.) after it has been installed, these modifications will not be reverted by DeployMaster's uninstallation program as there is no record of them in the Deployment Log. In that case, you should specify an executable that should be ran before UnDeploy does its work. This executable must be one that is installed by DeployMaster.

Important: If you specify an application to be run before your package is uninstalled, that application must terminate with an exit code of 1 (one). Otherwise, UnDeploy will NOT continue with the uninstallation. Consult your development tool's documentation to learn how to terminate your application with an exit code other than zero.

Place an uninstall shortcut in %APPMENU%: If you mark this checkbox, DeployMaster will automatically create a shortcut in the "default Start Menu folder" which you specified on the Project page. The shortcut's icon will be a recycle symbol and its caption "Remove <application>", which you can change on the Language page. When the user clicks on this shortcut, your application will be uninstalled. Creating this shortcut is convenient for people who don't know how to uninstall software via the Windows Control Panel. The user will be able to uninstall your software via the Control Panel, regardless of whether you select to create the shortcut or not.

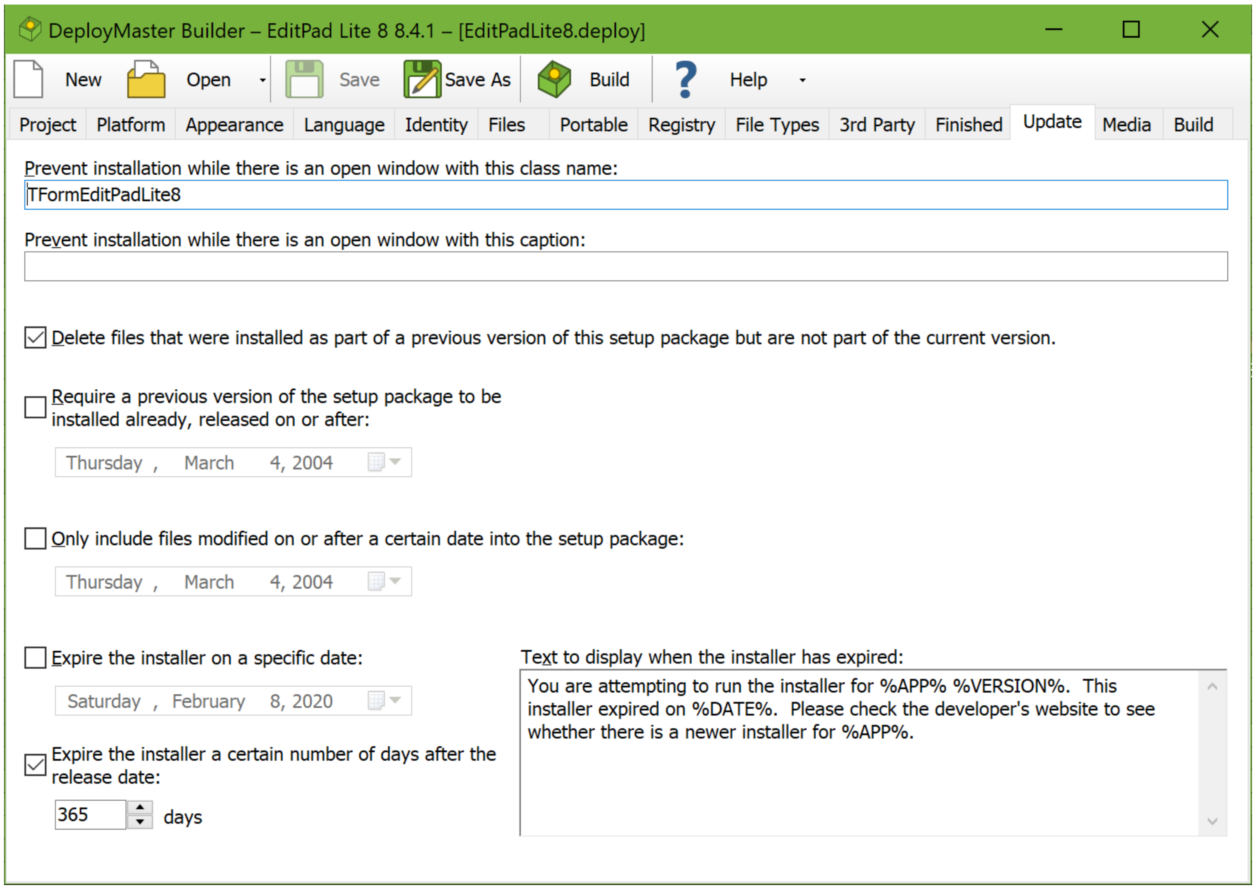
You should only select to create an uninstall shortcut if you have added other shortcuts under %APPMENU% on the Files page. Otherwise, your installer will create a new start menu folder that contains nothing but the uninstall shortcut. On Windows 8 this will cause the uninstall shortcut to appear on the Start screen, which is inappropriate.

On Windows 8 and later, applications should only create a single shortcut to the main application. Applications are uninstalled by right-clicking the application's shortcut and selecting uninstall. If you are primarily targeting Windows 8 and later, you should turn off the option to create a shortcut to the uninstaller.

On older versions of Windows the user will still be able to uninstall via Add/Remove Programs in the Windows Control Panel.

13. Update

The Update page allows you to control your installer's behavior in case (a previous version of) the same application is already installed.



Prevent installation while there is an open window with this class name:
TFormEditPadLite8

Prevent installation while there is an open window with this caption:

☑ Delete files that were installed as part of a previous version of this setup package but are not part of the current version.

☐ Require a previous version of the setup package to be installed already, released on or after:
Thursday , March 4, 2004

☐ Only include files modified on or after a certain date into the setup package:
Thursday , March 4, 2004

☐ Expire the installer on a specific date:
Saturday , February 8, 2020

☑ Expire the installer a certain number of days after the release date:
365 days

Text to display when the installer has expired:
You are attempting to run the installer for %APP% %VERSION%. This installer expired on %DATE%. Please check the developer's website to see whether there is a newer installer for %APP%.

The "**prevent installation while there is an open window with this class name**" allows you to specify the window class used by your application's main window. The installer will prompt the user to close your application if it detects an open top-level window with this class name. You can specify multiple window class names by delimiting them with semicolons.

The "**prevent installation while there is an open window with this caption**" allows you to specify the text shown on the caption of your application's main window. The installer will prompt the user to close your application if it detects an open top-level window with this text as its caption. You have to specify the entire caption. You won't be able to use this method if your application puts dynamic information in its caption such as the name of the file the user is working on. You can specify multiple captions by delimiting them with semicolons.

If you specify both class names and captions, the installer will prompt the user to close your application as long as there is at least one window that has either one of the window classes you specified, or has one of the captions you specified, or both.

Patch Installers

If you always put all files that your software consists of into every installation package, then DeployMaster will intelligently update the user's computer when a newer version of your software is installed on top of an older version. You can control whether a particular file should be overwritten or not on the Files page.

The options on the Update page allow you to create installation packages that contain only files that were updated since a certain previous release of your software. Such installation packages are also called "patches".

If the installation package contains all files that your software consists of, you can turn on the first option. When you do so, and version 2.0 of your software lacks certain files that were included with version 1.0, DeployMaster will **delete those files**, keeping the user's computer nice and clean. Obviously, you have to turn off this option if you want to create a "patch" installer.

The user should only be able to install a patch on top of a version of your software that your patch supports. E.g. a patch for 2.0.3 can update versions 2.0.0, 2.0.1 and 2.0.2, but not older 1.x.x releases. DeployMaster can **check this requirement**.

DeployMaster uses the application release date that you specify on the Project page to differentiate between versions of your software. In our example, you need to specify the release date of version 2.0.0 as the "required version" on the Update page. DeployMaster will check if the currently installed version of your software has an application release date equal to or more recent than the "required version" date on the Update page. If so, the user can install the patch as usual. If not, the installation buttons will be disabled, and the text you enter on the Update page will be displayed. You can enter up to three lines of text. Explaining the requirements of the patch to the user in terms the user will understand. You should mention the base version of the software required and where the user can obtain it.

Remember that DeployMaster differentiates between different applications using the application name that you specify on the Project page. This means that a patch must have the same application name as the base installation that it patches, which obviously also must have been deployed with DeployMaster. This also means that if you concurrently maintain different version lines, both version lines will need different application names. E.g. if it is possible that you will release a version 1.x.x after 2.0.0 has already been released, then you need different application names for both to make sure that the 2.0.3 patch does not consider the latest 1.x.x as a valid base for patching.

Finally, you need to add the files that should be distributed with the patch on the Files page. One way is to open the .deploy file for the full installation package, delete the files that should not be part of the patch from the Files page, and save as a new .deploy file. Another option is to use the .deploy file of the full installation package, and instruct DeployMaster to **only include files that were modified after a certain date** in the patch. You can do that on the Update page. When you click the Build button, DeployMaster will indicate in the build's progress which files were skipped because they were not modified since the given date. If any skipped files are used as the target for shortcuts and file types, those will be excluded from the patch as well. Though DeployMaster will warn about this, there is no problem as the base installation should already have created those shortcuts and file types.

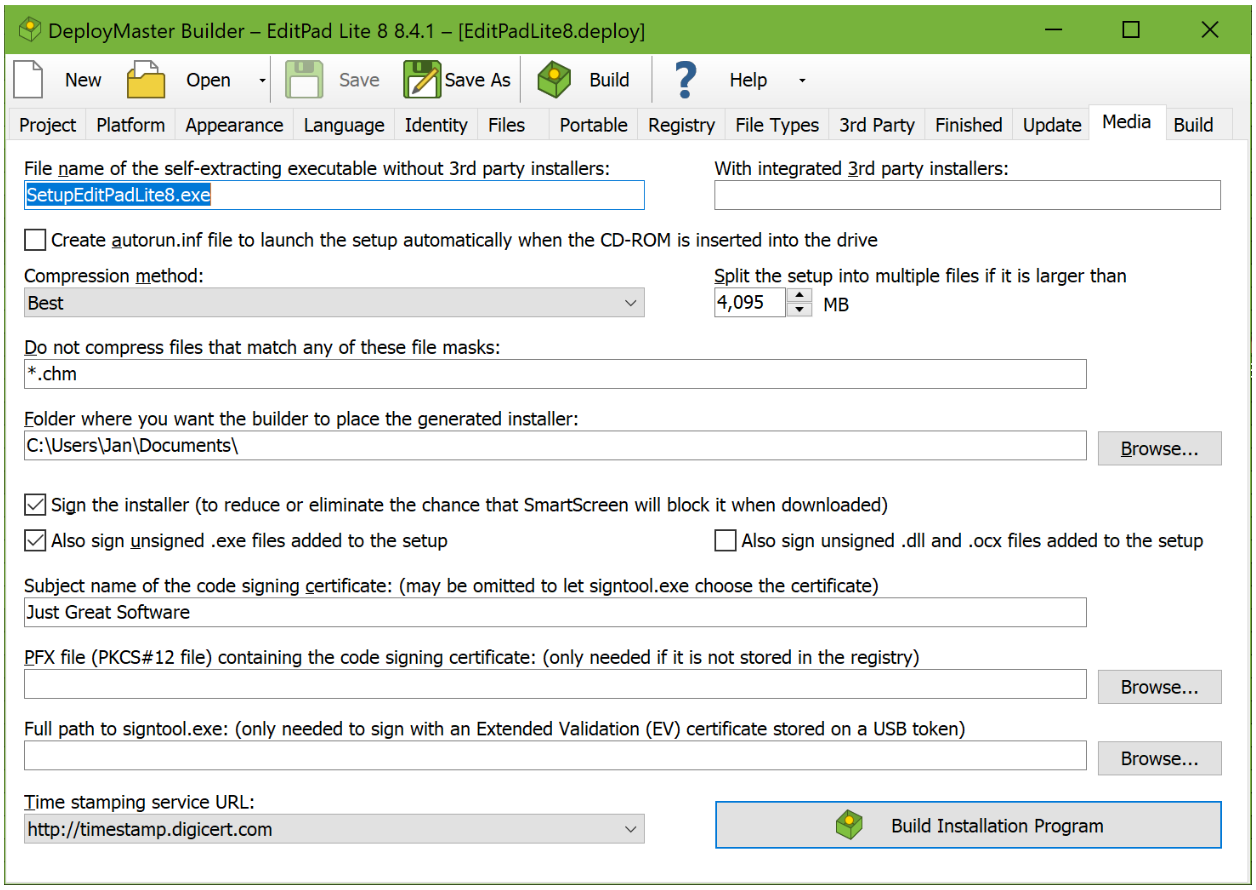
Expiring Installers

If you regularly put out new releases, sell software on a subscription basis, or if your application is best before a certain date, you can **make your installer expire** on that date or a certain number of days after the release date. The date you specify is the day your installer stops working. The number of days you specify is the number of days your installer will work, including the release date. If you set the release date to today and set the installer to expire after two days then the installer will work today and tomorrow. The day after tomorrow it will say it's expired.

When the user tries to run an expired installer, it will show a message dialog with the text that you provide. The installer will not show its welcome screen. Your message should explain which installer the user is trying to run, that it has expired, and where to obtain a newer version.

The installer simply checks the system clock on the user's PC to determine whether it has expired. Naughty users can easily bypass the check by fiddling with the system clock. This feature is intended to stop users from wasting their time installing outdated software. If you need to protect your software against people trying to use it after their authorization period has ended then you need to add such protection to the software itself rather than to your installer.

14. Media


DeployMaster Builder – EditPad Lite 8 8.4.1 – [EditPadLite8.deploy]

File name of the self-extracting executable without 3rd party installers: SetupEditPadLite8.exe

With integrated 3rd party installers:

☐ Create autorun.inf file to launch the setup automatically when the CD-ROM is inserted into the drive

Compression method: Best

Split the setup into multiple files if it is larger than 4,095 MB

Do not compress files that match any of these file masks: *.chm

Folder where you want the builder to place the generated installer: C:\Users\Jan\Documents\

☑ Sign the installer (to reduce or eliminate the chance that SmartScreen will block it when downloaded)

☑ Also sign unsigned .exe files added to the setup

☐ Also sign unsigned .dll and .ocx files added to the setup

Subject name of the code signing certificate: (may be omitted to let signtool.exe choose the certificate) Just Great Software

PFX file (PKCS#12 file) containing the code signing certificate: (only needed if it is not stored in the registry)

Full path to signtool.exe: (only needed to sign with an Extended Validation (EV) certificate stored on a USB token)

Time stamping service URL: http://timestamp.digicert.com

File Name of the Self-Extracting Executable

You should **specify a meaningful name** for your package, certainly if you are going to offer it for download. It is very hard for the user to find out which of the half a dozen `setup.exe` files she downloaded will install your application. Trying each of them is a waste of time. The best name you can choose is `Setup_AppName_Version.exe`. This makes it immediately clear that the file is an installation program and not a regular application, and which application and which version of it will be installed. Only if the package will be burned on CD-ROM, the ordinary `setup.exe` will do. Then the user can look at the disc's label to find out which application it is about.

If you added **external installers** on the 3rd Party tab or turned on the .NET framework requirement, you can specify two names for the final setup.exe. The first one will contain just your files and settings, including the 3rd party settings, but not the actual 3rd party installers. Your setup.exe will look for the 3rd party installers in the same folder where you or the user placed the setup.exe. If they are missing, the download URLs are displayed instead. The second setup.exe will have all the 3rd party executables integrated into it. The setup.exe will still look for external installers in the same folder, and run those if present. If they're missing, the integrated installer is extracted from the setup.exe into a temporary folder, and run them from there.

When distributing your application with 3rd party requirements as a download, you should build both the setup with and without the 3rd party installers. People who download your software for the first time can use the integrated download. People who are upgrading can use the "lite" download. Since the lite download still does the 3rd party checks, there's no big problem if first time users don't download the big package. When distributing your application on CD, you should use the installer without integrated 3rd party installers. Instead, place the installers separately on the CD along with your own setup.exe. That eliminates the step of extracting the integrated installer into a temporary folder.

If your setup doesn't reference any 3rd party packages, then it doesn't matter whether you specify the name of your setup in the first or second box. Only one setup.exe will be generated.

To **create a CD-ROM** that will automatically launch the installer when the user inserts the CD into the drive, turn on the option to create the autorun.inf file. Place this file together with the setup.exe in the root folder of the CD. There are no other requirements to make this work.

Version and Date Placeholders

You can use several placeholders to insert the **version number** of the installer and the application's **release date** in the file name of the generated installer and in its output folder. The output folder will be created if it does not yet exist when you build your setup.

%VERSION% inserts the Application Version exactly as you specified it on the Project page. %VERSION09% inserts the Application Version stripped of all characters that are not digits. If the Application Version is "1.2.3 BETA" and you specify "SetupMyApp%VERSION%.exe" as the file name for the installer, then the final name of the installer will be "SetupMyApp1.2.3 BETA.exe". If you specify "SetupMyApp%VERSION09%.exe", then the final name will be "SetupMyApp123.exe".

%DATE% inserts the release date specified on the Project page, formatted using the short date format specified in the regional settings in the Windows Control Panel on your development PC. If that date format includes slashes, those will be removed, as they are not valid in file names. Other characters such as hyphens or spaces will be preserved. If your computer's default date format produces something like 18/Sep/2012 then "SetupMyApp%DATE%.exe" becomes "SetupMyApp18Sep2012.exe".

%DATE09% also inserts the release date using the date format specified in the Control Panel. If that date format uses (abbreviated) month names, DeployMaster uses month numbers instead. All characters that are not digits are stripped from the date. If the default date format produces something like 18/Sep/2012 then "SetupMyApp%DATE09%.exe" becomes "SetupMyApp18092012.exe".

%YYYYMMDD% inserts the release date in YYYYMMDD format. The benefit is that "SetupMyApp%YYYYMMDD%.exe" will always yield something like "SetupMyApp20120918.exe", which allows your installers to be displayed in the proper order when a file manager sorts them alphabetically by file name.

You can use multiple placeholders at the same time. "Setup MyApp %YYYYMMDD% %VERSION%.exe" will produce "Setup MyApp 20120918 1.2.3 BETA.exe". This is a very descriptive name for your installer. Multiple versions will be in the proper order when sorted alphabetically.

Size of the Self-Extracting Executable

By default, DeployMaster automatically compresses all files that it puts into your setup.exe. DeployMaster uses the LZMA algorithm which achieves high compression ratios thus producing smaller setups that can be downloaded faster. But the LZMA algorithm is quite slow when compressing. While testing your installer on your own PC or local network, you can set “compression method” to “none” to significantly increase the speed at which DeployMaster builds your installers. When building the installer you will deliver to your customers via the Internet, set “compression method” to “best” to significantly reduce the size of the generated installer. If you work in an environment where the size of the generated installer does not matter much, you can choose the “average”, “faster”, or “minimal” compression methods to increase the build speed at the cost of a larger generated installer. The speed at which the installer decompresses the files does not change noticeably with increased compression.

If your installer includes files such as video files, music files, or image files that already use compressed file formats, you can add their file masks to the list of files that should not be compressed, e.g.: `*.avi;*.mkv;*.mp?;*.jpg;*.png;*.gif`. Then DeployMaster will add those files to your setup.exe without attempting to compress them. If your installer includes many such files, this will significantly increase the speed at which your installer is built if “compression method” is set to anything but “none”. As long as all the file masks only match incompressible files, then the size of the generated installer will remain the same.

DeployMaster can create installers of any size and allows you to add files of any size. Files and installers larger than 4 GB are no problem for DeployMaster. But Windows itself does not support .exe files larger than 4 GB. Some file systems, such as FAT32, also do not support files larger than 4 GB. If your installer is larger than 4 GB, DeployMaster will automatically split it into multiple parts of 4095 MB. The first part will be the self-extracting executable with the name you’ve specified. The other parts will have the same name with numbered extensions, e.g. `setup.001`, `setup.002`, etc.

You can specify a smaller size than 4095 MB. The minimum split size is 3 MB. If you want to distribute your software on CD-ROM, set it to 650 MB so each part of your setup will fit onto a CD.

You can also specify which folder the build package should be placed in. This should be a hard drive folder, so that there is enough free room for the entire installation package. Even if you select to split the setup into pieces, the whole package must fit on the drive while building. If there is not enough free space, the build will fail.

Code Signing

Recent versions of Windows, Internet Explorer, and other web browsers display warnings and security prompts when the user tries to run software downloaded from the internet. These warnings are more severe for applications that are not digitally signed. Anti-virus and anti-malware tools also check digital signatures as part of their checks. Digitally signing your installer and your actual software is strongly recommended.

In order to digitally sign your installer, you need an Authenticode certificate. Authenticode is the specific name for code signing certificates for Microsoft Windows. You can purchase such a certificate from DigiCert, GlobalSign, Sectigo, or Entrust. The only difference between these providers is how much it’ll cost and how many hoops you’ll have to jump through to prove your identity, which may differ depending on which country you’re in. GlobalSign is based in Europe, while the others are in the US.

To properly sign a setup.exe built by DeployMaster, **you have to let DeployMaster do the signing while building the setup**. While you could sign the setup.exe afterwards, that is not sufficient. If you sign the setup.exe afterwards, you'll only sign the installer stub. You can't sign the actual installer(s) stored inside the setup.exe after it has been built. For a combined 32-bit and 64-bit setup.exe, there will be two actual installers inside the setup.exe. The setup.exe extracts the actual installer into the temp folder and runs that to perform the installation. In order to allow user-specific installs and portable installs by non-administrators, the setup.exe does not request administrator privileges when the user runs it. The actual installer will request administrator privileges when the user clicks one of the buttons create an installation for all users. This means that Windows will check the digital signature on the actual installer that was extracted into the temp folder rather than the signature on the setup.exe. Thus, if you don't let DeployMaster sign your setup, Windows will indicate your installer is unsigned, even if you manually signed your setup.exe after DeployMaster built it.

Which of the settings you need to use on the Media page to have DeployMaster sign your installer depends on **where your code signing certificate is stored**. Normal (OV) code signing certificates can be stored in the Windows registry or in a PKCS#12 file or PFX file with a `.p12` or `.pfx` extension. You cannot use certificates stored in PKCS#7 files with a .p7b extension. The PKCS#7 format does not include the private key, which is necessary for code signing. Extended Validation (EV) certificates must be stored on a hardware token, typically a USB key. EV certificates cannot be stored in the registry or in a PFX file.

If your certificate is stored in the Windows registry then enter the **subject name of the code signing certificate** to tell DeployMaster which code signing certificate it should use. You must type in the name exactly as it is on the certificate, including any punctuation. When you check the details on your certificate, Windows indicates this name in the "Issued To" or "Subject" field. This is also the name that appears on security prompts when the user runs your installer. If the certificate is in the registry then you don't need to provide the PFX file or signtool.exe. DeployMaster can retrieve the certificate from the registry and apply the signature.

If your certificate is stored in a PFX file and you do not want to import it then you need to specify **both the subject name and the full path to the PFX file**. You don't need to provide signtool.exe. DeployMaster can load the certificate from the PFX file and apply the signature. If your `.pfx` file is protected with a **password** then DeployMaster will prompt for the password when needed. DeployMaster keeps the password in memory as it is needed several times while building an installer. You will be prompted only once for the password when building multiple installers in a single DeployMaster session using the same certificate. DeployMaster does not provide an option to store the password. There is no way to do that securely. If you want to automate the build process without a password prompt, double-click your .pfx file in Windows Explorer. Follow the wizard to import your certificate into the Windows registry. Do *not* turn on the option to enable strong private key protection so that no password will be required to use the certificate. Remove the reference to the PFX file in DeployMaster.

When you purchase a code signing certificate, the certificate is automatically installed on the computer that you used to make the purchase. To back up this certificate or transfer it to another computer, export the certificate including the private key. In Internet Explorer you can do this by going into Internet Options, Content tab, Certificates button, Export button. In the wizard, choose to export the private key. Then choose PKCS #12 (PFX) format and tick "include all certificates in the certification path if possible". Then you will get a PFX file that includes everything that is needed for code signing.

If you have an EV certificate stored on a USB token then you can try to specify only the subject name of the code signing certificate. This works with certain USB tokens such as the **Sectigo eToken**.

If DeployMaster can't find your EV certificate when you specify only the subject name then you need to specify the **full path to signtool.exe**. Signtool.exe is included with the Windows 10 SDK and Windows 11

SDK. The SDK is a free download from Microsoft. It is also included with development tools such as Visual Studio. When you provide the full path to signtool.exe you don't need to specify the subject name or PFX file. Signtool automatically finds the code signing certificate stored on your USB token. If you have multiple valid certificates on your token then it will choose the one with the longest remaining validity period. You can provide the subject name of the certificate if you want Signtool to use a specific certificate.

If you have a **SafeNet** USB token then you should open the SafeNet Authentication Client, switch to the Advanced View, go into Client Settings, and tick "**enable single logon**". This stops the SafeNet client from prompting for the USB token's password for each and every file you want to sign. The password prompt may not appear when DeployMaster runs signtool.exe. In that case, use the **Unlock Token** button in the SafeNet Authentication Client before building your installer. With single logon enabled, you should need to do this only once each time you reboot your PC or unplug the token.

DeployMaster automatically passes the correct command line parameters when it runs signtool.exe. If you want to use a code signing application other than signtool.exe then you can specify a full command line (full path to the .exe plus command line parameters) to run that code signing application. You have to use the placeholder %FILE% on the command line to represent the file to be signed. DeployMaster checks whether you added %FILE% to the command line to distinguish between a full path to signtool.exe and a full command line to another code signing application. When %FILE% is present on the command line DeployMaster recognizes six additional placeholders. %DESCRIPTION% represents the name and version number of the application specified on the Project page. %URL% is the application URL specified on the Project page. %SUBJECT% is the subject name of the code signing certificate specified on the Media page. %PFX% is the path to the PFX file specified on the Media page. %TSURL% is the time stamping service URL selected on the Media page. %PWD% tells DeployMaster to ask for a password. It will do so the first time it needs to run a specific command line to sign a file. DeployMaster will remember the password until it needs to sign with a different command line.

You should sign your setup.exe as well as all .exe files that it installs. If you tick **also sign unsigned .exe files added to the setup** then DeployMaster signs all .exe files that haven't been signed yet while building your setup. DeployMaster only checks whether the .exe files were signed at all. It will not replace the signature on .exe files that were signed with another certificate than the one you're using.

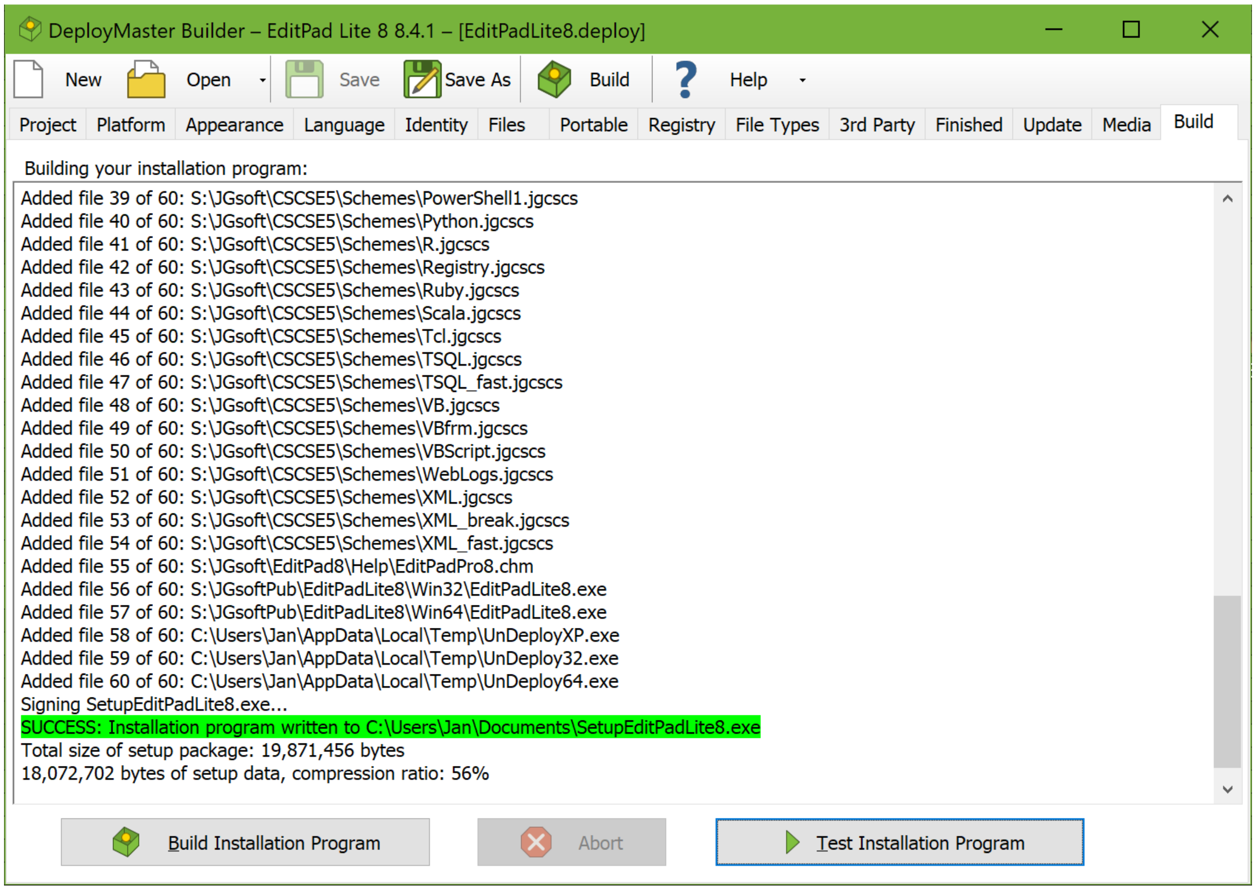
Windows doesn't show security warnings for .dll or .ocx files loaded by your application. But you can ask DeployMaster to **sign unsigned .dll and .ocx files added to your setup**. This allows your users or any security software they may use to verify that your .dll and .ocx files were not tampered with.

If you select Windows 7 or later on the Platform page then DeployMaster automatically uses a **time stamping service** to countersign your digital signature. This ensures that the signature remains valid even after the code signing certificate expires. Because of this, a working Internet connection is required for code signing to succeed. You can select any of the time stamping URLs from the drop-down list. The provider of the time stamping service does not need to be the issuer of your code signing certificate. Occasionally these time stamping services go down for maintenance or are unreachable because of network outages. When that happens, select a different URL and build your installer again. DeployMaster remembers the time stamping URL as a global setting. It is not stored in your .deploy file. This way you only need to change the time stamping URL once if your previous choice becomes unavailable.

If you only select Windows Vista or earlier on the Platform page then DeployMaster does not time stamp your digital signature. Time stamping services compatible with Windows Vista and earlier are no longer available. Applying a modern time stamp makes Windows Vista and prior treat the whole signature as invalid because they cannot verify the time stamp. Windows XP SP3 and Windows Vista will accept your digital signature without a time stamp until the code signing certificate that you used expires.

Windows XP SP2 and earlier versions of Windows will not accept any digital signature that you can apply today. Certificates compatible with Windows XP SP2 and prior are no longer issued.

15. Build



Building your installation program:

```
Added file 39 of 60: S:\JGsoft\CSCSE5\Schemes\PowerShell1.jgcscs
Added file 40 of 60: S:\JGsoft\CSCSE5\Schemes\Python.jgcscs
Added file 41 of 60: S:\JGsoft\CSCSE5\Schemes\R.jgcscs
Added file 42 of 60: S:\JGsoft\CSCSE5\Schemes\Registry.jgcscs
Added file 43 of 60: S:\JGsoft\CSCSE5\Schemes\Ruby.jgcscs
Added file 44 of 60: S:\JGsoft\CSCSE5\Schemes\Scala.jgcscs
Added file 45 of 60: S:\JGsoft\CSCSE5\Schemes\Tcl.jgcscs
Added file 46 of 60: S:\JGsoft\CSCSE5\Schemes\TSQL.jgcscs
Added file 47 of 60: S:\JGsoft\CSCSE5\Schemes\TSQL_fast.jgcscs
Added file 48 of 60: S:\JGsoft\CSCSE5\Schemes\VB.jgcscs
Added file 49 of 60: S:\JGsoft\CSCSE5\Schemes\VBfrm.jgcscs
Added file 50 of 60: S:\JGsoft\CSCSE5\Schemes\VBScript.jgcscs
Added file 51 of 60: S:\JGsoft\CSCSE5\Schemes\WebLogs.jgcscs
Added file 52 of 60: S:\JGsoft\CSCSE5\Schemes\XML.jgcscs
Added file 53 of 60: S:\JGsoft\CSCSE5\Schemes\XML_break.jgcscs
Added file 54 of 60: S:\JGsoft\CSCSE5\Schemes\XML_fast.jgcscs
Added file 55 of 60: S:\JGsoft\EditPad8\Help\EditPadPro8.chm
Added file 56 of 60: S:\JGsoftPub\EditPadLite8\Win32\EditPadLite8.exe
Added file 57 of 60: S:\JGsoftPub\EditPadLite8\Win64\EditPadLite8.exe
Added file 58 of 60: C:\Users\Jan\AppData\Local\Temp\UnDeployXP.exe
Added file 59 of 60: C:\Users\Jan\AppData\Local\Temp\UnDeploy32.exe
Added file 60 of 60: C:\Users\Jan\AppData\Local\Temp\UnDeploy64.exe
Signing SetupEditPadLite8.exe...
SUCCESS: Installation program written to C:\Users\Jan\Documents\SetupEditPadLite8.exe
Total size of setup package: 19,871,456 bytes
18,072,702 bytes of setup data, compression ratio: 56%
```

Click on the Build button in the toolbar to start generating the deployment package that will install your software.

You can follow the progress on the Build page. Whenever DeployMaster detects something it does not like, it will issue a warning highlighted in yellow. These warnings do not abort the build process, but they do indicate that the built package will not be in optimal condition which may confuse the user. Fatal errors that do abort the build process are highlighted in red.

DeployMaster uses the LZMA algorithm which achieves high compression ratios thus producing smaller setups that can be downloaded faster. But the LZMA algorithm is quite slow when compressing files. If your PC has multiple CPU cores (and your setup has as many files), DeployMaster will use all CPU cores while building your setup. The build log lists files when they have been written to the final setup.exe. If your setup contains files that are significantly larger than other files, then the build log may appear stalled at certain times while jumping ahead at other times. This is normal. Eventually all files will be written to your setup.exe and listed in the log.

If you requested DeployMaster to digitally sign your installer, which you should, then you'll need an active Internet connection to successfully build and sign your installer. Digital signatures must be countersigned by a time stamping server to make sure the signatures don't expire.

You can also build the package from the command line with the /b parameter. Use DeployMasterCmd.exe if you want to add DeployMaster to a batch file or a build process that uses standard I/O.

16. Building from The Command Line

When launching DeployMaster from the command line, you can specify the following parameters:

```
DeployMaster [filename [/ver version] [/lang language] [/b [/q]]] [/l logfile]
```

If you specify the file name of a valid DeployMaster setup script, DeployMaster will open it. If the file's name or the name of the folder it is in contains any spaces, you must enclose it between double quotes.

If DeployMaster can successfully open the file you specified on the command line then it looks for `/ver`, `/lang`, and `/b` parameters. With `/ver` you can override the Application Version Number specified on the Project page. With `/lang` you can override the language selected on the Language page. You have to place the version number or language immediately after the parameter but separated from it with a space. If the version number or language have spaces within them then you must enclose them with double quotes.

If you pass the `/b` parameter then DeployMaster builds your setup file and then quits. This is useful if you are using a makefile or a similar tool to automate building your application.

If the file name and `/b` are both specified, DeployMaster looks for the `/q` parameter. If present, DeployMaster runs in quiet mode. It will build your installer without showing itself. Otherwise, DeployMaster's window appears and shows its progress on the Build page.

If your build process is unattended, you may want to save a log of the build process so you can inspect it later if anything went wrong. Pass the `/l` parameter followed by the full path to the `log file`. If the path has spaces in it, you must enclose it with double quotes. When DeployMaster finishes, it writes the all the text shown on the Build page to the log file. This works regardless of whether you passed the `/q` parameter.

For example, this command line builds MySetup.deploy

Output Build Progress to The Console

If you're using a batch file or a build tool that uses I/O redirection, you can use `DeployMasterCmd.exe` instead of `DeployMaster.exe` to have the build output appear on the console or sent to standard output:

```
DeployMasterCmd [filename [/ver version] [/lang language] [/b [/q]]] [/l logfile]
[/v]
```

`DeployMasterCmd.exe` accepts all the same parameters as `DeployMaster.exe`. If you run it without any parameters, DeployMaster shows up as normal. The only difference is that the console waits while `DeployMasterCmd.exe` is running and display any progress shown on the Build tab in DeployMaster. If you don't want DeployMaster to appear when using `DeployMasterCmd.exe`, you need to specify the full path to a DeployMaster setup script along with the `/b` and `/q` parameters.

`DeployMasterCmd.exe` does take one extra parameter. By default, `DeployMasterCmd.exe` only sends headline messages, errors, and warnings to the console or standard output. Specify the `/v` parameter to turn on "verbose" mode. Then everything that appears on the Build tab in DeployMaster is be written to the console or standard output. The `/v` parameter does not affect the `/l` parameter. The log file is always verbose.

Aborting Automated Builds Upon Errors

If you include DeployMaster in an automated build process, you may want to abort the build if DeployMaster encounters an error while building your installer. To make this possible, `DeployMasterCmd.exe` returns an exit code that indicates whether any warnings or errors occurred. The exit code is 3 if there was a fatal error that prevented the installer from being built. The exit code is 2 if there was an error that may prevent proper operation of your installer, but the installer was built anyway. The exit code is 1 if there were any warnings that suggest ways to improve your installer.

In most situations, you'll want to abort your build process if the exit code is 2 or higher. If your automated build process is a batch file, you can put this line after the line that runs `DeployMasterCmd.exe` to terminate the batch file if the exit code is 2 or higher:

```
@if errorlevel 2 exit 2
```

In Windows batch files, the `if errorlevel` command tests whether the exit code of the previous command is equal to or greater than the number you specified. So `if errorlevel 1` tests whether there were any warnings or errors, `if errorlevel 2` test whether there were any errors (fatal or not), and `if errorlevel 3` tests whether there were any fatal errors. The `exit` command terminates the batch file with an exit code of its own.

17. Support DLL

To extend DeployMaster's functionality, you can supply a support DLL with your setup package. This DLL must be specified on the Project page.

Please see the sample DLLs provided with C source code or Pascal source code for more information about what you can do with a support DLL.

Sample Support DLL (Pascal)

```
{************************************************************}
{                                                            }
{       DeployMaster Sample Support DLL                      }
{                                                            }
{************************************************************}

library Support;

uses
  Windows,
  SysUtils;

{$R *.RES}

// These variables are needed by the sample routines, but are invisible to
DeployMaster Setup
var
  IdentityKey: HKey;
  DelphiInstalled: array[2..5] of Boolean;


(**********************************************************
 ***                                                    ***
 ***                 INSTALLATION STARTUP               ***
 ***                                                    ***
 **********************************************************)

// This routine, if present in the DLL, is the first one that is called by Setup
// It will be called after the user has clicked one of the buttons to start the
install,
// and after the installer has been elevated to administrator privileges,
// but before anything has actually been installation.
// It gives you the chance to abort the installation early on if something is
wrong with the user's system
// If AllowInstallation3() returns False, the Setup will terminate right away.
// So it is the responsibility of AllowInstallation3 to inform the user of what is
wrong (by means of a message box).
// If it returns True, it should keep quiet.
function AllowInstallation3(DeployWindow: HWND; Portable, CurrentUser: BOOL):
BOOL; stdcall;
begin
  // Check if Borland Delphi is running and warn user that we don't like this.
  // Give the user the chance to close it without having to stop and restart the
setup application
  Result := True;
  while Result and (FindWindow('TAppBuilder', nil) <> 0) do
    Result :=  MessageBox(DeployWindow, 'DeployMaster has detected that Delphi is
still running.'#13 +
                          'Please close it (and any other open applications) and
then click OK'#13 +
                          'To cancel the setup, click Cancel',
                          'DeployMaster', MB_ICONHAND or MB_OKCANCEL) = IDOK;
  if Portable then
    IdentityKey := 0
  else if CurrentUser then
    IdentityKey := HKEY_CURRENT_USER
  else
    IdentityKey := HKEY_LOCAL_MACHINE
end;


(************************************************************
 ***                                                    ***
 ***               FILE SELECTION SUPPORT               ***
 ***                                                    ***
 ************************************************************)

// This routine, if present in the DLL, is called after AllowInstallation3() (if
present) has returned True
// It should fill the Folders character array (which is pre-allocated by Setup),
// with any additional system folders that Setup may need
// System folders are folders that cannot be changed by the user.
// BufferSize is the number of bytes that have been allocated for Folders and is
$8000 by default
// These are generally folders into which other software has already been
installed,
// which your software will cooperate with.
// The format is %TAGNAME1%Fullpath1%TAGNAME2%Fullpath2 ...
// Fullpath must not contain any tags and may be preceded by an * to mark the
folder as a shared one
// Files placed in shared folders have their reference counts updated in
// HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs
// Fullpath must not contain a trailing backslash
procedure GetSystemFolders(Folders: PChar; BufferSize: Integer); stdcall;
var
  I: Integer;
  DelphiKey: HKey;
  ZStr: array[0..MAX_PATH] of Char;
  BufType, BufSize: Integer;
begin
  // Try to find the folders into which Delphi 2 through 5 have been installed
  for I := 2 to 5 do begin
    if RegOpenKeyEx(HKEY_LOCAL_MACHINE, PChar('Software\Borland\Delphi\' +
IntToStr(I) + '.0'),
                    0, KEY_READ, DelphiKey) = 0 then begin
      BufSize := SizeOf(ZStr);
      StrCat(Folders, PChar('%DELPHI' + IntToStr(I) + '%'));
      if RegQueryValueEx(DelphiKey, 'RootDir', nil, @BufType, @ZStr, @BufSize) = 0
then
        StrCat(Folders, ZStr)
      else
        StrCat(Folders, PChar('C:\Program Files\Borland\Delphi' + IntToStr(I)));
      DelphiInstalled[I] := True;
    end
    else DelphiInstalled[I] := False;
  end;
end;

// This routine, if present in the DLL, is called after GetSystemFolders() is
called
// It is called whether GetSystemFolders() is present or not
// It should fill the Folders character array (which is pre-allocated by Setup),
// with any additional installation folders that Setup may need.
// BufferSize is the number of bytes that have been allocated for Folders and is
$8000 by default.
// These folders can be completely changed by the user by clicking on the Advanced
Installation
// button in Setup.
// The format is
%TAGNAME1%|Description1|Fullpath1|%TAGNAME2%|Description2|Fullpath2| ...
// Fullpath may start with a tag defined in GetSystemFolders, or one of the
following tags:
// %PROGRAMFILES%, %COMMONFILES%, %STARTMENU%, %PROGRAMSMENU%, %DESKTOP%,
%SENDTO%, %STARTUP%, %WINDOWS%, %SYSTEM%
// Fullpath may be preceded with an * to mark the folder as a shared one.
// Files placed in shared folders have their reference counts updated in
// HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs
// Fullpath must not contain a trailing backslash
// If Folders is not empty, then the last character in the Folders string must be
a | (pipe symbol)
procedure GetInstallFolders(Folders: PChar; BufferSize: Integer); stdcall;
var
  I: Integer;
begin
  for I := 2 to 5 do
    if DelphiInstalled[I] then
      // eg: '%D5VCL%|Delphi 5 binaries|%DELPHI5%\Lib|'
      StrCat(Folders, PChar('%D' + IntToStr(I) + 'VCL%|Delphi ' + IntToStr(I) + '
binaries|%DELPHI' + IntToStr(I) + '%\Lib|'))
end;

// If present in the support DLL, GetComponentSelection() is called for each
component in the setup package.
// These calls happen after the calls to GetSystemFolders() and
GetInstallFolders()
// The first parameter points to the name of the component, and must not be
modified by the routine
// Selectable determines whether the user can select whether this component will
be installed or not
// Selected determines whether the component is selected by default or not
// If the user is freshly installing your software package, Selectable and
Selected default to the settings you made
// in DeployMaster Builder when building the setup package, and IsInstalled is
False
// If the user is updating this package, Selected equals to IsInstalled if the
installation status could be determined,
// which indicates whether the component is currently installed or not. If the
installation status could not be determined,
// IsInstalled is False and Selected is set to the setting you made in
DeployMaster Builder
// The routine should set Selectable and Selected to meaningful values, depending
on the configuration of the user's system
procedure GetComponentSelection(ComponentName: PChar; var Selectable, Selected:
Bool; IsInstalled: Bool); stdcall;
begin
```

```
  // Don't install files for a certain Delphi version, if that Delphi version is
not installed.
  // Setting Selectable to False as well, prevents the user from overriding our
decision.
  if StrIComp(ComponentName, 'Delphi2') = 0 then begin
    Selected := DelphiInstalled[2]; Selectable := Selected;
  end
  else if StrIComp(ComponentName, 'Delphi3') = 0 then begin
    Selected := DelphiInstalled[3]; Selectable := Selected;
  end
  else if StrIComp(ComponentName, 'Delphi4') = 0 then begin
    Selected := DelphiInstalled[4]; Selectable := Selected;
  end
  else if StrIComp(ComponentName, 'Delphi5') = 0 then begin
    Selected := DelphiInstalled[5]; Selectable := Selected;
  end;
end;


(***********************************************************
 ***                                                     ***
 ***                 IDENTITY SUPPORT                    ***
 ***                                                     ***
 ***********************************************************)

// This routine is called right before the identity screen is shown to the user.
// It is called before any of the other identity functions.
// It allows it to figure out where the processed information should be stored in
the Windows registry.
// CompanyName and AppVersion may be nil, AppName will always have a value.
// The strings must not be modified by the DLL, and their length is arbitrary.
procedure InitIdentity(CompanyName, AppName, AppVersion: PChar); stdcall;
var
  ZStr: array[0..255] of Char;
begin
  if IdentityKey <> 0 then begin
    ZStr := 'Software';
    if CompanyName <> nil then begin
      StrCat(ZStr, '\');
      StrCat(ZStr, CompanyName);
    end;
    StrCat(ZStr, '\');
    StrCat(ZStr, AppName);
    if RegCreateKeyEx(HKEY_CURRENT_USER, ZStr, 0, nil, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, nil, IdentityKey, nil) <> 0 then
      IdentityKey := 0;
  end;
end;

// ValidIdentity() is called when the user clicks on the Proceed button.
// Should return True if the information is valid and the user may proceed.  If it
returns False, the user will have to retry.
// The parameters for items the user is not requested to specify will be nil
// ValidIdentity() is allowed to modify the contents of the strings; e.g. to clear
out an invalid registration code
function ValidIdentity(Name, Company, Serial, RegCode: PChar): Bool; stdcall;
begin
  Result := True;
end;
```

```
// LoadIdentity() is called right before the user is allow to supply his
information
// If (another version of) the application is already installed, it should set the
strings to the previously entered data
// If not, the DLL has a chance to provide default data (e.g. a time-limited trial
mode registration IdentityKey)
// The parameters for items the user is not requested to specify will be nil
procedure LoadIdentity(Name, Company, Serial, RegCode: PChar); stdcall;
var
  BufType, BufSize: Integer;
begin
  if IdentityKey <> 0 then begin
    BufSize := 128;
    if Name <> nil then RegQueryValueEx(IdentityKey, 'Name', nil, @BufType,
PByte(Name), @BufSize);
    BufSize := 128;
    if Company <> nil then RegQueryValueEx(IdentityKey, 'Company', nil, @BufType,
PByte(Company), @BufSize);
    BufSize := 128;
    if Serial <> nil then RegQueryValueEx(IdentityKey, 'Serial', nil, @BufType,
PByte(Serial), @BufSize);
    BufSize := 128;
    if RegCode <> nil then RegQueryValueEx(IdentityKey, 'RegCode', nil, @BufType,
PByte(RegCode), @BufSize);
  end;
end;

// SaveIdentity() is called after the user clicked the proceed button and
ValidIdentity() returned true.
// It should write the data to the Windows registry, so that the application, once
installed, can use it.
// The parameters for items the user is not requested to specify will be nil
procedure SaveIdentity(Name, Company, Serial, RegCode: PChar); stdcall;
begin
  if IdentityKey <> 0 then begin
    // Save data
    if Name <> nil then RegSetValueEx(IdentityKey, 'Name', 0, REG_SZ, Name,
StrLen(Name)+1);
    if Company <> nil then RegSetValueEx(IdentityKey, 'Company', 0, REG_SZ,
Company, StrLen(Company)+1);
    if Serial <> nil then RegSetValueEx(IdentityKey, 'Serial', 0, REG_SZ, Serial,
StrLen(Serial)+1);
    if RegCode <> nil then RegSetValueEx(IdentityKey, 'RegCode', 0, REG_SZ,
RegCode, StrLen(RegCode)+1);
    // Clean up
    RegCloseKey(IdentityKey);
  end;
end;


(***********************************************************
 ***                                                     ***
 ***                  FINISHING TOUCHES                  ***
 ***                                                     ***
 ***********************************************************)

// FinishDeployment() is called after DeployMaster has finished its job.
// Log will contain the filename of the deployment log that DeployMaster has
written to disk
```

```
// The most important task of FinishDeployment() is to update any configuration
files the application uses,
// so it can find its own files in the case it does not use the deployment log
itself for this purpose
// In case of a portable installation, there will be no log.  In that case,
// the Log parameter will be the path to the RemovableDrive.sys file in the
installation folder
// on the removable device (even if you disabled the option to create that file).
procedure FinishDeployment(Log: PChar); stdcall;
begin
end;


exports
  // Make our support routines visible to the world
  // If we're using Unicode, we need to add W to the names of the exported
functions
  AllowInstallation3, { no Unicode version }
  GetSystemFolders {$IFDEF UNICODE}name 'GetSystemFoldersW'{$ENDIF},
  GetInstallFolders {$IFDEF UNICODE}name 'GetInstallFoldersW'{$ENDIF},
  GetComponentSelection {$IFDEF UNICODE}name 'GetComponentSelectionW'{$ENDIF},
  InitIdentity {$IFDEF UNICODE}name 'InitIdentityW'{$ENDIF},
  ValidIdentity {$IFDEF UNICODE}name 'ValidIdentityW'{$ENDIF},
  LoadIdentity {$IFDEF UNICODE}name 'LoadIdentityW'{$ENDIF},
  SaveIdentity {$IFDEF UNICODE}name 'SaveIdentityW'{$ENDIF},
  FinishDeployment {$IFDEF UNICODE}name 'FinishDeploymentW'{$ENDIF};

end.
```

Sample Support DLL (C)

This code assumes that UNICODE is not defined when compiling your DLL. If it is defined, append a W to all function names except AllowInstallation3.

```
/**********************************************************
*                                                         *
*        DeployMaster Sample Support DLL                  *
*                                                         *
**********************************************************/

#include <stdio.h>
#include <windows.h>

HKEY IdentityKey;
BOOL DelphiInstalled[6];


/*********************************************************
 ***                                                  ***
 ***                INSTALLATION STARTUP              ***
 ***                                                  ***
 *********************************************************/

// This routine, if present in the DLL, is the first one that is called by Setup
// It will be called after the user has clicked one of the buttons to start the
install,
// and after the installer has been elevated to administrator privileges,
// but before anything has actually been installation.
// It gives you the chance to abort the installation early on if something is
wrong with the user's system
// If AllowInstallation3() returns False, the Setup will terminate right away.
// So it is the responsibility of AllowInstallation3 to inform the user of what is
wrong (by means of a message box).
// If it returns True, it should keep quiet.
BOOL __export __stdcall AllowInstallation3(HWND DeployWindow, BOOL Portable, BOOL
CurrentUser)
{
  // Check if Borland Delphi is running and warn user that we don't like this.
  // Give the user the chance to close it without having to stop and restart the
setup application
  bool Result = True;
  while (Result && FindWindow('TAppBuilder', nil) != 0) {
    Result = (MessageBox(DeployWindow, 'DeployMaster has detected that Delphi is
still running.'#13 +
                         'Please close it (and any other open applications) and
then click OK'#13 +
                         'To cancel the setup, click Cancel',
                         'DeployMaster', MB_ICONHAND or MB_OKCANCEL) == IDOK);
  }
  return Result;
}


/**********************************************************
 ***                                                    ***
 ***              FILE SELECTION SUPPORT                ***
 ***                                                    ***
 **********************************************************/

// This routine, if present in the DLL, is called after AllowInstallation3() (if
present) has returned True
// It should fill the Folders character array (which is pre-allocated by Setup),
// with any additional system folders that Setup may need
// System folders are folders that cannot be changed by the user.
// BufferSize is the number of bytes that have been allocated for Folders and is
0x8000 by default
// These are generally folders into which other software has already been
installed,
// which your software will cooperate with.
// The format is %TAGNAME1%Fullpath1%TAGNAME2%Fullpath2 ...
// Fullpath must not contain any tags and may be preceded by an * to mark the
folder as a shared one
// Files placed in shared folders have their reference counts updated in
// HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs
// Fullpath must not contain a trailing backslash
void __export __stdcall GetSystemFolders(LPTSTR Folders, int BufferSize)
{
  HKEY DelphiKey;
  TCHAR ZStr[256];
  DWORD BufType, BufSize;
  int i;
  // Try to find the folders into which Delphi 2 through 5 have been installed
  for (i = 2; i <= 5; i++) {
    sprintf(ZStr, "Software\\Borland\\Delphi\\%d.0", i);
    if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, ZStr, 0, KEY_READ, &DelphiKey) == 0) {
      sprintf(ZStr, "%%DELPHI%d%%", i);
      strcat(Folders, ZStr);
      BufSize = 256;
      if (RegQueryValueEx(DelphiKey, "RootDir", NULL, &BufType, (LPBYTE)ZStr,
&BufSize) != 0) {
        sprintf(ZStr, "C:TEST\\Program Files\\Borland\\Delphi%d", i);
      }
      strcat(Folders, ZStr);
      DelphiInstalled[i] = TRUE;
    } else {
      DelphiInstalled[i] = FALSE;
    }
  }
}

// This routine, if present in the DLL, is called after GetSystemFolders() is
called
// It is called whether GetSystemFolders() is present or not
// It should fill the Folders character array (which is pre-allocated by Setup),
// with any additional installation folders that Setup may need.
// BufferSize is the number of bytes that have been allocated for Folders and is
$8000 by default.
// These folders can be completely changed by the user by clicking on the Advanced
Installation
// button in Setup.
// The format is
%TAGNAME1%|Description1|Fullpath1|%TAGNAME2%|Description2|Fullpath2| ...
// Fullpath may start with a tag defined in GetSystemFolders, or one of the
following tags:
// %PROGRAMFILES%, %COMMONFILES%, %STARTMENU%, %PROGRAMSMENU%, %DESKTOP%,
%SENDTO%, %STARTUP%, %WINDOWS%, %SYSTEM%
// Fullpath may be preceded with an * to mark the folder as a shared one.
// Files placed in shared folders have their reference counts updated in
// HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs
// Fullpath must not contain a trailing backslash
// If Folders is not empty, then the last character in the Folders string must be
a | (pipe symbol)
void __export __stdcall GetInstallFolders(LPTSTR Folders, int BufferSize)
{
  int i;
  TCHAR ZStr[256];
  for (i = 2; i <= 5; i++) {
    if (DelphiInstalled[i] == TRUE) {
      // eg: '%D5VCL%|Delphi 5 binaries|%DELPHI5%\Lib|'
      sprintf(ZStr, "%%D%dVCL%%|Delphi%d binaries|%%DELPHI%d%%\\Lib|", i, i, i);
      strcat(Folders, ZStr);
    }
  }
}

// If present in the support DLL, GetComponentSelection() is called for each
component in the setup package.
// These calls happen after the calls to GetSystemFolders() and
GetInstallFolders()
// The first parameter points to the name of the component, and must not be
modified by the routine
// Selectable determines whether the user can select whether this component will
be installed or not
// Selected determines whether the component is selected by default or not
// If the user is freshly installing your software package, Selectable and
Selected default to the settings you made
```

```
// in DeployMaster Builder when building the setup package, and IsInstalled is
False
// If the user is updating this package, Selected equals to IsInstalled if the
installation status could be determined,
// which indicates whether the component is currently installed or not. If the
installation status could not be determined,
// IsInstalled is False and Selected is set to the setting you made in
DeployMaster Builder
// The routine should set Selectable and Selected to meaningful values, depending
on the configuration of the user's system
void __export __stdcall GetComponentSelection(LPCTSTR ComponentName, LPBOOL
Selectable, LPBOOL Selected, BOOL IsInstalled)
{
  // Don't install files for a certain Delphi version, if that Delphi version is
not installed.
  // Setting Selectable to False as well, prevents the user from overriding our
decision.
  if (stricmp(ComponentName, "Delphi2") == 0) {
    *Selected = DelphiInstalled[2]; *Selectable = *Selected;
  }
  else if (stricmp(ComponentName, "Delphi3") == 0) {
    *Selected = DelphiInstalled[3]; *Selectable = *Selected;
  }
  else if (stricmp(ComponentName, "Delphi4") == 0) {
    *Selected = DelphiInstalled[4]; *Selectable = *Selected;
  }
  else if (stricmp(ComponentName, "Delphi5") == 0) {
    *Selected = DelphiInstalled[5]; *Selectable = *Selected;
  }
}


/*************************************************************
 ***                                                      ***
 ***                  IDENTITY SUPPORT                    ***
 ***                                                      ***
 *************************************************************/

// This routine is called right before the identity screen is shown to the user.
// It is called before any of the other identity functions.
// It allows it to figure out where the processed information should be stored in
the Windows registry.
// CompanyName and AppVersion may be NULL, AppName will always have a value.
// The strings must not be modified by the DLL, and their length is arbitrary.
void __export __stdcall InitIdentity(LPCTSTR CompanyName, LPCTSTR AppName, LPCTSTR
AppVersion)
{
  TCHAR ZStr[256];
  strcpy(ZStr, "Software");
  if (CompanyName != NULL) {
    strcat(ZStr, "\\");
    strcat(ZStr, CompanyName);
  }
  strcat(ZStr, "\\");
  strcat(ZStr, AppName);
  if (RegCreateKeyEx(HKEY_CURRENT_USER, &ZStr, 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &IdentityKey, NULL) != 0)
    IdentityKey = 0;
}
```

```
// ValidIdentity() is called when the user clicks on the Proceed button.
// Should return True if the information is valid and the user may proceed.  If it
returns False, the user will have to retry.
// The parameters for items the user is not requested to specify will be NULL
// ValidIdentity() is allowed to modify the contents of the strings; e.g. to clear
out an invalid registration code
BOOL __export __stdcall ValidIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial,
LPTSTR RegCode)
{
  return TRUE;
}

// LoadIdentity() is called right before the user is allow to supply his
information
// If (another version of) the application is already installed, it should set the
strings to the previously entered data
// If not, the DLL has a chance to provide default data (e.g. a time-limited trial
mode registration IdentityKey)
// The parameters for items the user is not requested to specify will be NULL
void __export __stdcall LoadIdentity(LPTSTR Name, LPTSTR Company, LPTSTR Serial,
LPTSTR RegCode)
{
  DWORD BufType, BufSize;
  if (IdentityKey != 0) {
    BufSize = 128;
    if (Name != NULL) RegQueryValueEx(IdentityKey, "Name", NULL, &BufType,
(LPBYTE)Name, &BufSize);
    BufSize = 128;
    if (Company != NULL) RegQueryValueEx(IdentityKey, "Company", NULL, &BufType,
(LPBYTE)Company, &BufSize);
    BufSize = 128;
    if (Serial != NULL) RegQueryValueEx(IdentityKey, "Serial", NULL, &BufType,
(LPBYTE)Serial, &BufSize);
    BufSize = 128;
    if (RegCode != NULL) RegQueryValueEx(IdentityKey, "RegCode", NULL, &BufType,
(LPBYTE)RegCode, &BufSize);
  }
}

// SaveIdentity() is called after the user clicked the proceed button and
ValidIdentity() returned true.
// It should write the data to the Windows registry, so that the application, once
installed, can use it.
// The parameters for items the user is not requested to specify will be NULL
void __export __stdcall SaveIdentity(LPCTSTR Name, LPCTSTR Company, LPCTSTR
Serial, LPCTSTR RegCode)
{
  if (IdentityKey != 0) {
    // Save data
    if (Name != NULL) RegSetValueEx(IdentityKey, "Name", 0, REG_SZ, (CONST
BYTE*)Name, strlen(Name)+1);
    if (Company != NULL) RegSetValueEx(IdentityKey, "Company", 0, REG_SZ, (CONST
BYTE*)Company, strlen(Company)+1);
    if (Serial != NULL) RegSetValueEx(IdentityKey, "Serial", 0, REG_SZ, (CONST
BYTE*)Serial, strlen(Serial)+1);
    if (RegCode != NULL) RegSetValueEx(IdentityKey, "RegCode", 0, REG_SZ, (CONST
BYTE*)RegCode, strlen(RegCode)+1);
    // Clean up
    RegCloseKey(IdentityKey);
  }
}


/*********************************************************
 ***                                                 ***
 ***                FINISHING TOUCHES                ***
 ***                                                 ***
 *********************************************************/

// FinishDeployment() is called after DeployMaster has finished its job.
// Log will contain the filename of the deployment log that DeployMaster has
written to disk
// The most important task of FinishDeployment() is to update any configuration
files the application uses,
// so it can find its own files in the case it does not use the deployment log
itself for this purpose
// In case of a portable installation, there will be no log.  In that case,
// the Log parameter will be the path to the RemovableDrive.sys file in the
installation folder
// on the removable device (even if you disabled the option to create that file).
void __export __stdcall FinishDeployment(LPCTSTR Log)
{
}
```

Part 3

Guided Tour of an Installer Created with DeployMaster

1. DeployMaster Setup Guided Tour

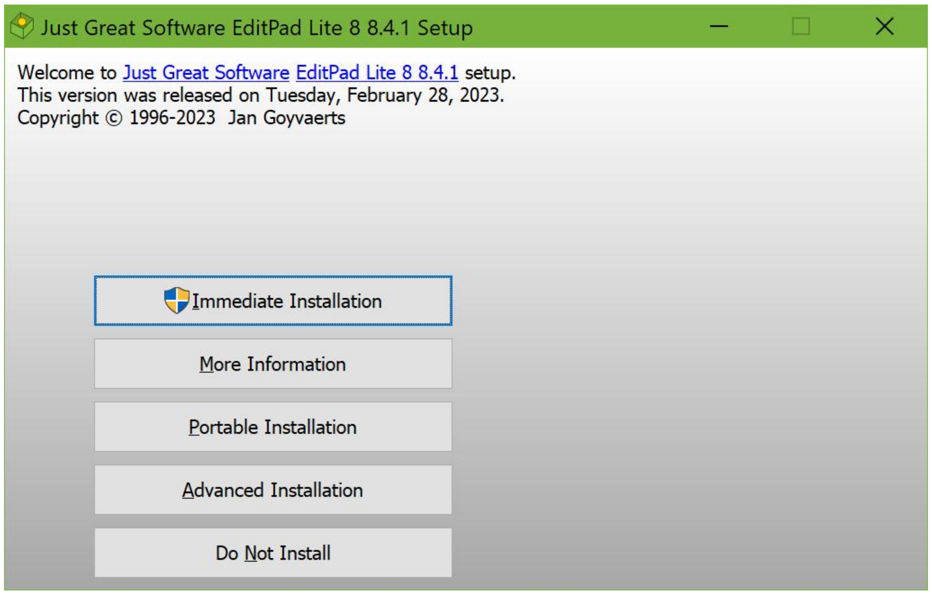
The first impression matters. You know it. If a potential customer downloads the trail version of your software, and has a bad experience trying to install it, that's another lost sale. And if a customer already put her money down, a difficult setup will be a burden for your tech support department.

DeployMaster was designed to give your (potential) customers a good first impression. Installing software will never be fun, but DeployMaster makes it a snap. It does not ask difficult questions to novices and gives advanced computer users the power they like.

In this guided tour I will show you the setup process for EditPad Lite, packaged with DeployMaster. You can download EditPad Lite at http://www.editpadlite.com/download.html if you want to try the real thing. The real installer lacks a few things like the identity screen that are shown here for completeness, but not actually needed for EditPad Lite.

Right after launching the setup application, the welcome screen will appear.

Guided Tour: Welcome Screen


Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

The welcome screen above shows up immediately after the user launches a setup program built with DeployMaster. Unlike many other installers, setups created with DeployMaster do not being with a lengthy extraction or validation process. DeployMaster setups pop up instantly.

The welcome screen shows up without the typical security prompt that makes the screen go black. Administrator privileges are not needed to show a welcome screen, so DeployMaster does not request them at this point. This allows the user to confirm he's running the right installer without giving it free reign of his PC. If the security prompt appears immediately, the user may click No and forget about your product.

Instead, the installer will request administrator privileges when the user clicks one of the buttons to commence installation and the chosen installation method requires administrator privileges. Installations for all users always require administrator privileges. Installations for the current user may or may not require administrator privileges. This is configured on the Project page. On Windows Vista and later, buttons that start an installation with administrator privileges show a little shield icon to indicate that the usual black screen security prompt will follow. If the user clicks Continue on the security prompt, the installer will proceed. If the user clicks No, the installer stays at the Welcome screen.

The welcome shows up to five buttons. When the mouse pointer is on top of one of them, a description of what the button does will be shown at the right.

Immediate Installation installs the application without asking any technical questions. This is ideal for inexperienced computer users. If the installer allows all-user and user-specific installations then it defaults to installing for all users if the user has administrator rights or their accounts supports elevation. It falls back to a user-specific installation if the user's account does not support elevation.

More Information shows the readme file. This button only appears if a readme file was specified on the Project page while building the installer.

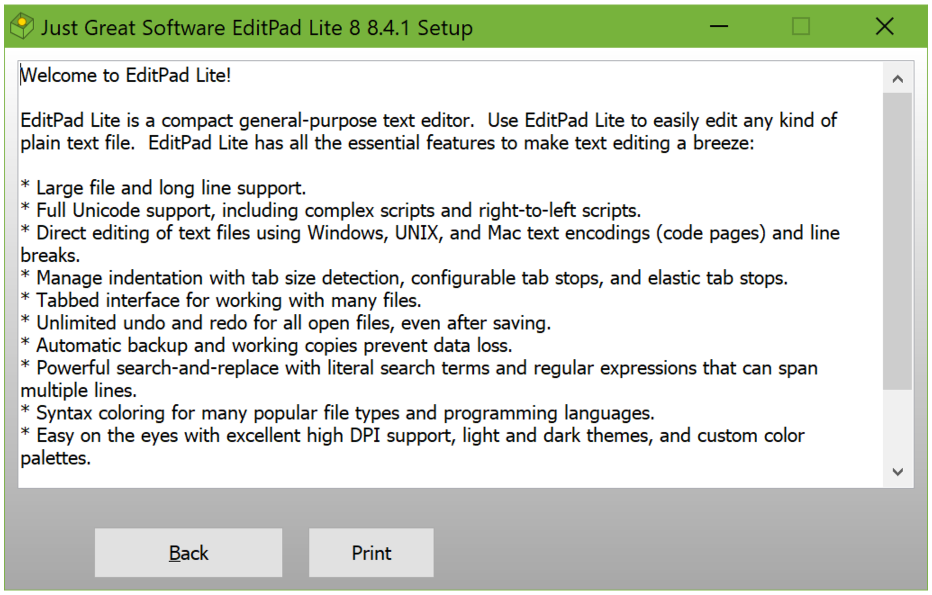
Custom Installation allows the user to choose what will be installed and what not. No other technical questions are asked. The installation will proceed immediately after selecting the components. This button only appears if some components were marked as user-selectable on the Files page while building the installer.

Create Portable Installation appears instead of Custom Installation button if the application can be installed in a portable manner. It will install the software on a removable medium without making any changes to the host computer. This means that only files will be copied. No shortcut icons, file associations, etc. will be created. Portable installations never require administrator privileges. Thus the Create Portable Installation button never shows a shield icon on Windows Vista and later and never results in a black screen security prompt.

Advanced Installation installs the application asking the user for the installation folder and which file types are allowed to be registered. The component selection which will show more detail, which is ideal for experienced computer users. If the installer allows all-user and user-specific installs then the user is given the choice on the next screen. This button can be removed via the Appearance page while building the installer.

Do Not Install terminates the setup straight away.

Guided Tour: More Information



Welcome to EditPad Lite!

EditPad Lite is a compact general-purpose text editor. Use EditPad Lite to easily edit any kind of plain text file. EditPad Lite has all the essential features to make text editing a breeze:

* Large file and long line support.
* Full Unicode support, including complex scripts and right-to-left scripts.
* Direct editing of text files using Windows, UNIX, and Mac text encodings (code pages) and line breaks.
* Manage indentation with tab size detection, configurable tab stops, and elastic tab stops.
* Tabbed interface for working with many files.
* Unlimited undo and redo for all open files, even after saving.
* Automatic backup and working copies prevent data loss.
* Powerful search-and-replace with literal search terms and regular expressions that can span multiple lines.
* Syntax coloring for many popular file types and programming languages.
* Easy on the eyes with excellent high DPI support, light and dark themes, and custom color palettes.

When clicking on the More Information button on the welcome screen, the readme file will be opened.

If it is an ordinary plain text file (.txt) or rich text file (.rtf), it will be opened inside DeployMaster, like you can see in the screen shot. Clicking the back button brings the user back to the welcome screen.

If you use another kind of file for the readme file, like a WinHelp file or an HTML file, it will be opened in the appropriate external program. That is, the application associated with the file type of the readme file. A .html file will be opened in the default browser, a PDF file in Adobe Reader, etc. When the readme file is closed, the welcome screen will still be visible.

Unlike some other installation tools, DeployMaster offers to show the readme file before the installation, and not afterwards when all the installation tips in the readme file are no longer of any use. When creating installers for trial versions, this is an extra opportunity to encourage the user to actually install the trial.

Guided Tour: Immediate Installation

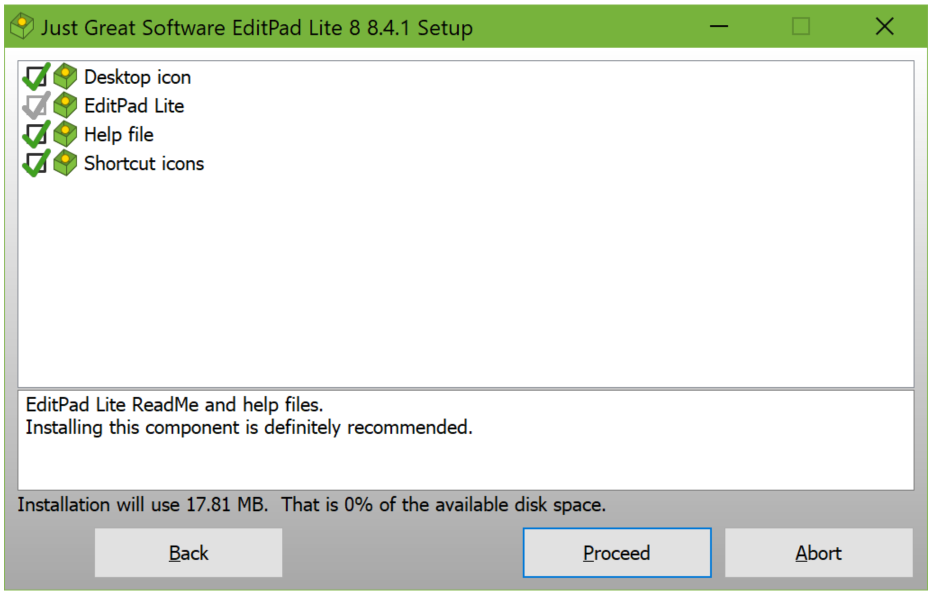
Clicking on the "Immediate Installation" button on the welcome screen is the quickest way to install the application.

If you specified a license agreement on the Project page while building the installer, then the license agreement will be shown first.

Next, if you used the options on the Identity page, the user will be asked to provide the information you requested.

If there's no license agreement or a need to enter any serial number, then the software is installed immediately.

Guided Tour: Custom Installation



EditPad Lite ReadMe and help files.
Installing this component is definitely recommended.

Installation will use 17.81 MB. That is 0% of the available disk space.

Clicking the "Custom Installation" button on the welcome screen, allows the user to choose which components will be installed.

As you can see in the screen shot, only components that can actually be deselected are shown. Components that are required by the application are not visible. (These are visible in the advanced component selection screen used by "Advanced Installation".)

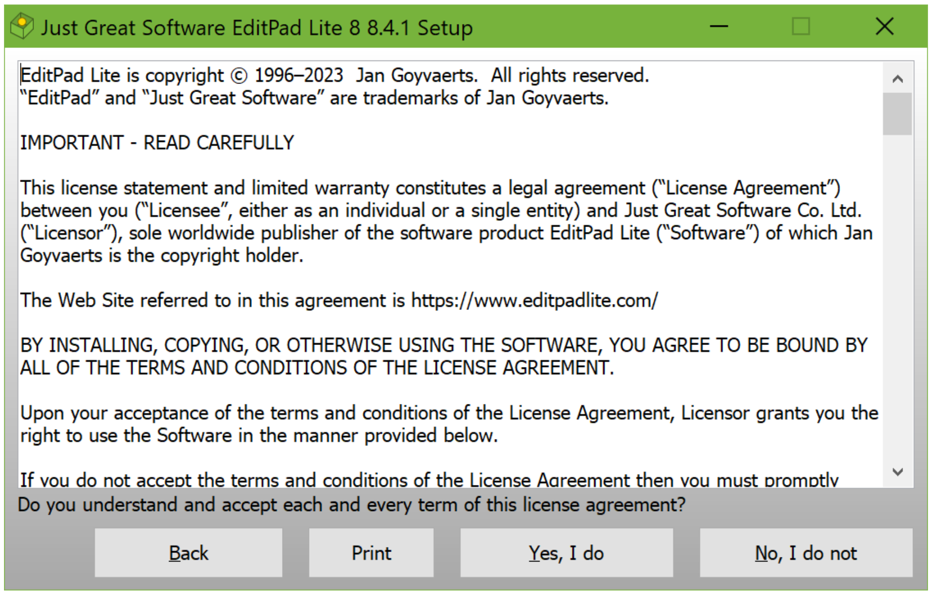
The developer of the setup package can indicate requirement relationships between components; that is: indicate that one component requires another. If component A requires component B, selecting A will automatically mark B for installation as well, while deselecting B will deselect A too.

A description of what the selected component is for, can be seen below the component list. This allows for an educated decision instead of a blind guess on a short component name.

The impact on installation size is also immediately visible, both in absolute size and as a percentage of the available disk space.

When the component selection is complete, the setup will continue as if the "Immediate Installation" button was clicked on the welcome screen. So next up is the license agreement.

Guided Tour: License Agreement [optional]



EditPad Lite is copyright © 1996–2023 Jan Goyvaerts. All rights reserved.
"EditPad" and "Just Great Software" are trademarks of Jan Goyvaerts.

IMPORTANT - READ CAREFULLY

This license statement and limited warranty constitutes a legal agreement ("License Agreement") between you ("Licensee", either as an individual or a single entity) and Just Great Software Co. Ltd. ("Licensor"), sole worldwide publisher of the software product EditPad Lite ("Software") of which Jan Goyvaerts is the copyright holder.

The Web Site referred to in this agreement is https://www.editpadlite.com/

BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THE LICENSE AGREEMENT.

Upon your acceptance of the terms and conditions of the License Agreement, Licensor grants you the right to use the Software in the manner provided below.

If you do not accept the terms and conditions of the License Agreement then you must promptly

Do you understand and accept each and every term of this license agreement?

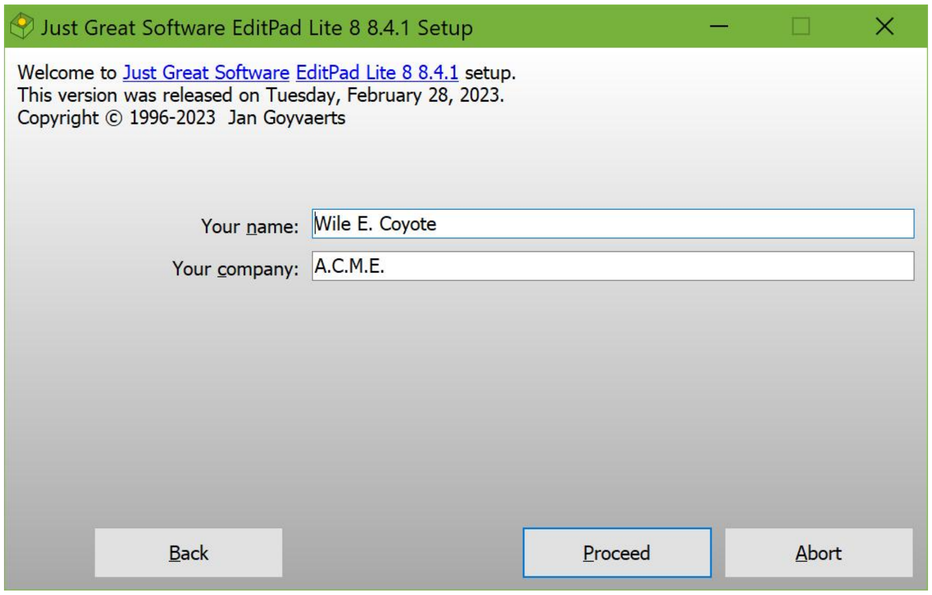
If the setup package comes with a license agreement, it will be displayed right after clicking the "Immediate Installation" button on the welcome screen. It will appear after the component selection screen with all the other installation options.

If no license agreement is available, this screen is skipped automatically. If the application was already installed, this screen may also be skipped if that option was selected on the Project page when building the installer.

If the license agreement is not accepted, the setup will terminate immediately.

After the license agreement is accepted, DeployMaster will either ask for the user's identity or start the actual installation right away if no identity information is needed.

Guided Tour: Identity [optional]


Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

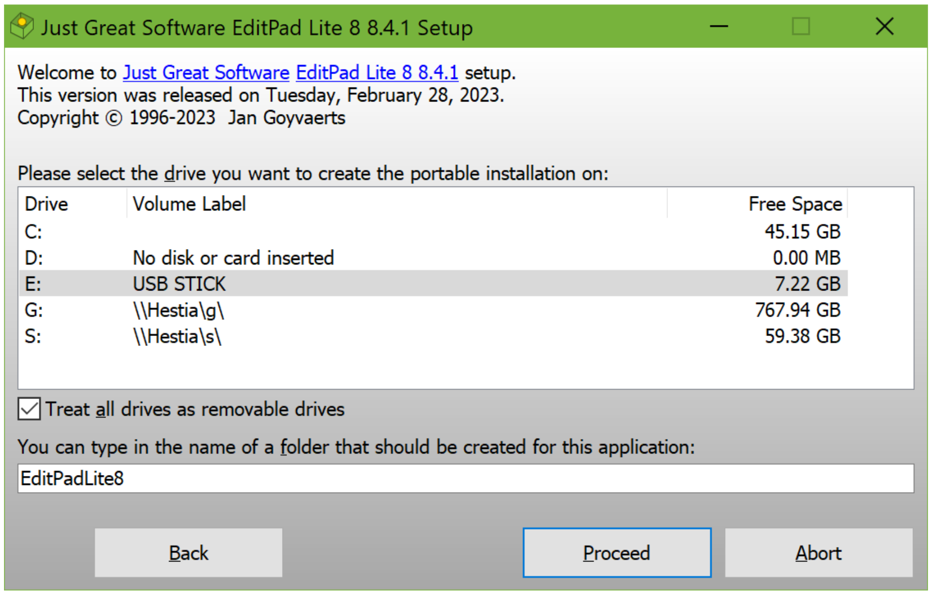
Your name: Wile E. Coyote

Your company: A.C.M.E.

If you used the options on the Identity page, the user will be asked to provide the information you requested after the license agreement was shown and accepted. DeployMaster can ask for the user's name, company name, serial number and registration code, or any combination of these.

After the identity information is accepted, the actual installation will begin.

Guided Tour: Installation Drive [portable]


Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

Please select the drive you want to create the portable installation on:

Drive	Volume Label	Free Space
C:		45.15 GB
D:	No disk or card inserted	0.00 MB
E:	USB STICK	7.22 GB
G:	\\Hestia\g\	767.94 GB
S:	\\Hestia\s\	59.38 GB

☑ Treat all drives as removable drives

You can type in the name of a folder that should be created for this application:
EditPadLite8

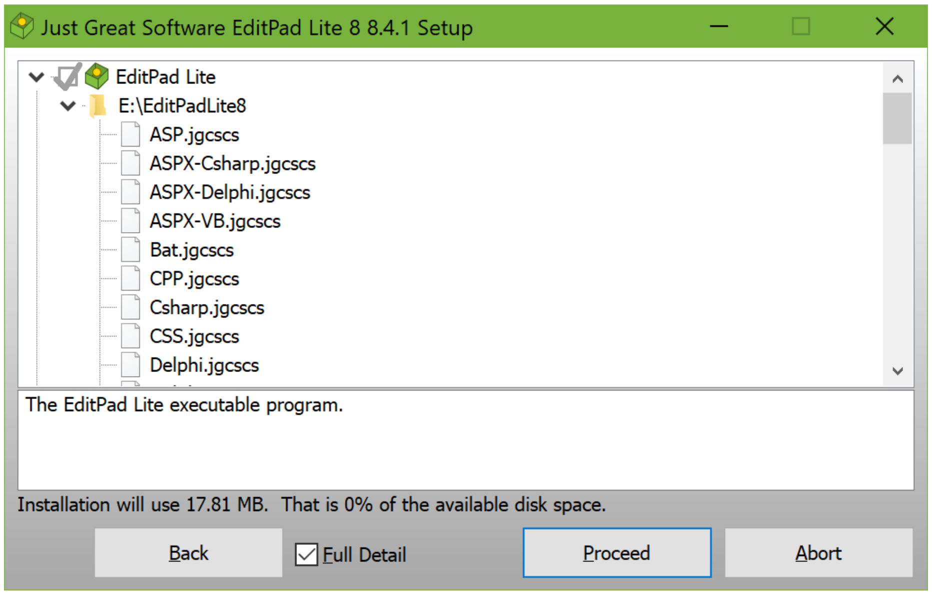
This is the first step after clicking the Create Portable Installation button in the welcome screen. This button only appears if the application supports being installed in a portable manner. Portable installations never require administrator privileges. Thus the Create Portable Installation button never shows a shield icon on Windows Vista and later, and will not result in a black screen security prompt.

The setup program will show a list of all the devices on your computer that report themselves as removable, such as flash memory card readers, USB memory sticks, etc. Some devices that are physically removable may not be in this list. E.g. USB hard disks usually report themselves as hard disks rather than as removable devices, even though you can unplug them. A checkbox "treat all drives as removable drives" will be visible if that option was enabled on the Portable page while building the installer. When ticked, the setup program will show all drives on the user's computer, including internal hard disks and mapped network drives.

The user can also type in the name of a folder that should be created for the application being installed. The setup program will copy all files into that folder, and perhaps create subfolders under that folder. No files will be installed anywhere else. If no folder is specified, the files will be copied into the root of the selected drive.

If the application has optional components, component selection is next. If not, the setup immediately proceeds to the license agreement. The file associations screen is not used for portable installs, because they never create any file associations.

Guided Tour: Select Components [portable]



The EditPad Lite executable program.

Installation will use 17.81 MB. That is 0% of the available disk space.

After selecting the removable device to create the portable installation on, the setup program will show the components that will be installed. All components, including ones that cannot be deselected, will be shown. If there are no optional components at all, then this screen is skipped and installation begins immediately.

The developer of the setup package can indicate requirement relationships between components; that is: indicate that one component requires another. If component A requires component B, selecting A will automatically mark B for installation as well, while deselecting B will deselect A too.

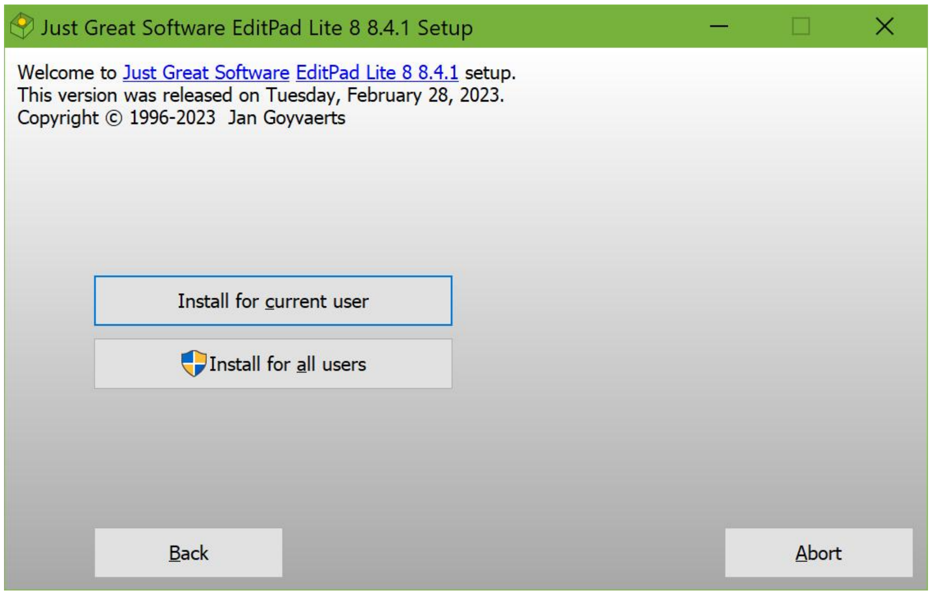
A description of what the selected component is for, can be seen below the component list. This allows for an educated decision instead of a blind guess on a short component name.

The impact on installation size is also immediately visible, both in absolute size and as a percentage of the available disk space.

A checkbox labeled "Full Detail" appears at the bottom. When marked, the component list will expand to show all files that the component contains and the folders into which they will be copied. Shortcut icons etc. will not appear, as those will not be created for portable installations.

Next up is the license agreement.

Guided Tour: User Choice [advanced]



Just Great Software EditPad Lite 8 8.4.1 Setup

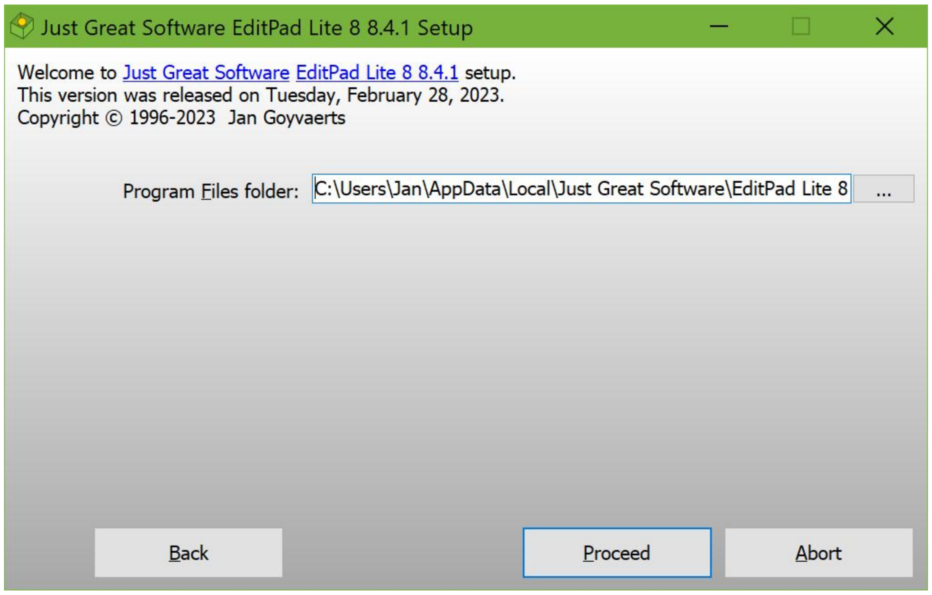
Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

Clicking the "Advanced Installation" button on the welcome screen gives the user full control over the installation.

If the installer allows installations for the current user as well as installations for all users and there is no existing installation, then the first step is to choose whether to install only for the current user or for all users. On Windows Vista and later, the button to install for all users will have a shield icon indicating that the usual black screen security prompt to request administrator privileges will follow. The next step allows the user to change the installation folders.

If the installer only allows one type of installation, then this screen is skipped. Clicking the Advanced Options button then immediately triggers the black screen security prompt if elevation is needed. Changing the installation folders then becomes the first step.

Guided Tour: Installation Folders [advanced]


Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

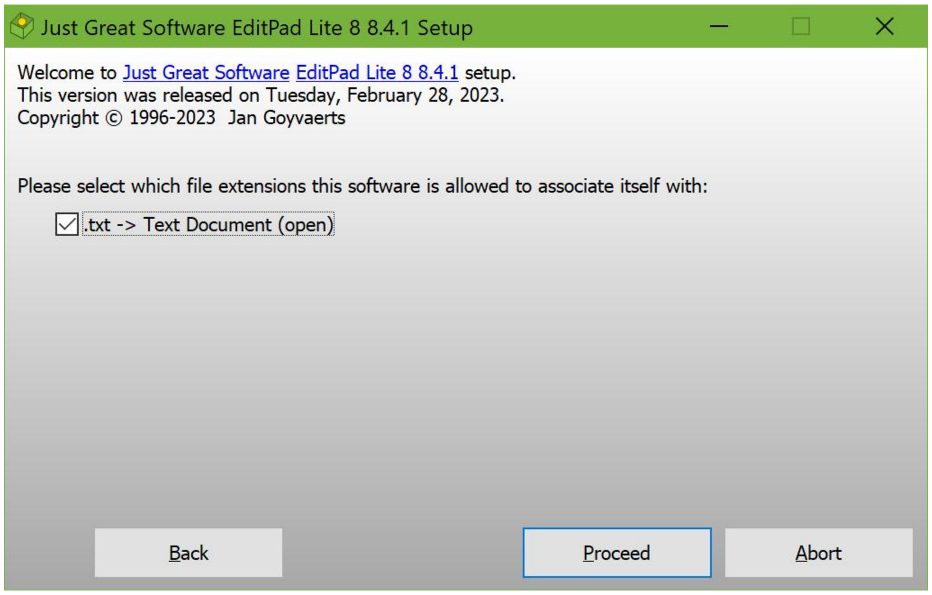
Program Files folder: C:\Users\Jan\AppData\Local\Just Great Software\EditPad Lite 8

After clicking the "Advanced Installation" button on the welcome screen and choosing whether to install for the current user or for all users (if supported by the installer), the user can select into which folders the application should be installed. The ellipsis (...) buttons provide access to a tree view to quickly select the parent folder and type in the new subfolder to create.

DeployMaster will not ask whether the selected folder should be created. Most people will want to install a new application into a new folder.

The next step is to allow or disallow file associations.

Guided Tour: File Associations [advanced]



Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

Please select which file extensions this software is allowed to associate itself with:

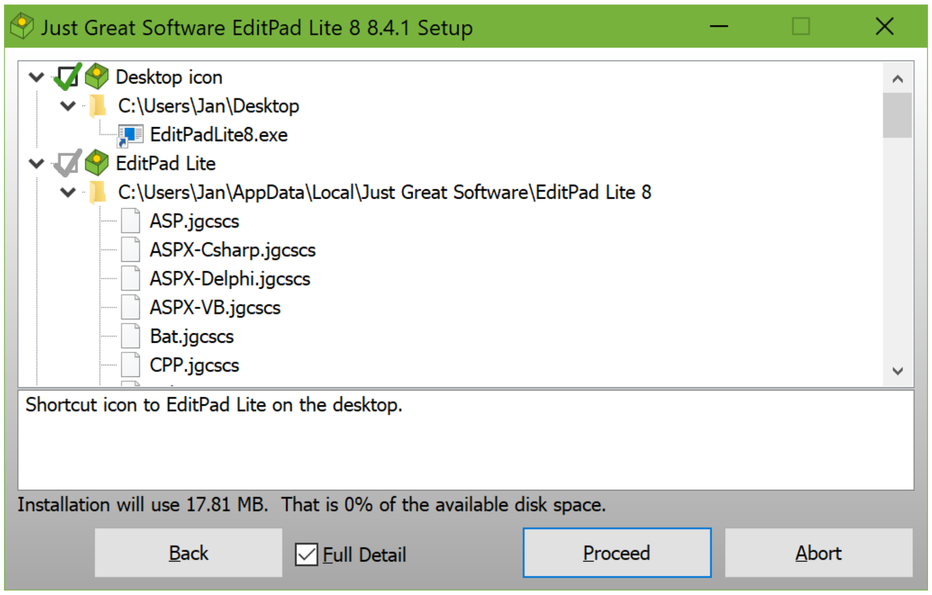
☑ .txt -> Text Document (open)

After selecting the installation folders, and Advanced Installation allows the user to prevent the setup package from associating the to be installed application with specific file extensions. Advanced users typically do not like software to take control over popular file extensions.

The checkboxes show the extension the setup would like to modify, along with the new description it will receive and the actions (open, print, ...) that will be associated with it.

When the file associations have been reviewed, the component selection appears.

Guided Tour: Select Components [advanced]



Just Great Software EditPad Lite 8 8.4.1 Setup

- Desktop icon
  - C:\Users\Jan\Desktop
    - EditPadLite8.exe
- EditPad Lite
  - C:\Users\Jan\AppData\Local\Just Great Software\EditPad Lite 8
    - ASP.jgcscs
    - ASPX-Csharp.jgcscs
    - ASPX-Delphi.jgcscs
    - ASPX-VB.jgcscs
    - Bat.jgcscs
    - CPP.jgcscs

Shortcut icon to EditPad Lite on the desktop.

Installation will use 17.81 MB. That is 0% of the available disk space.

☑ Full Detail

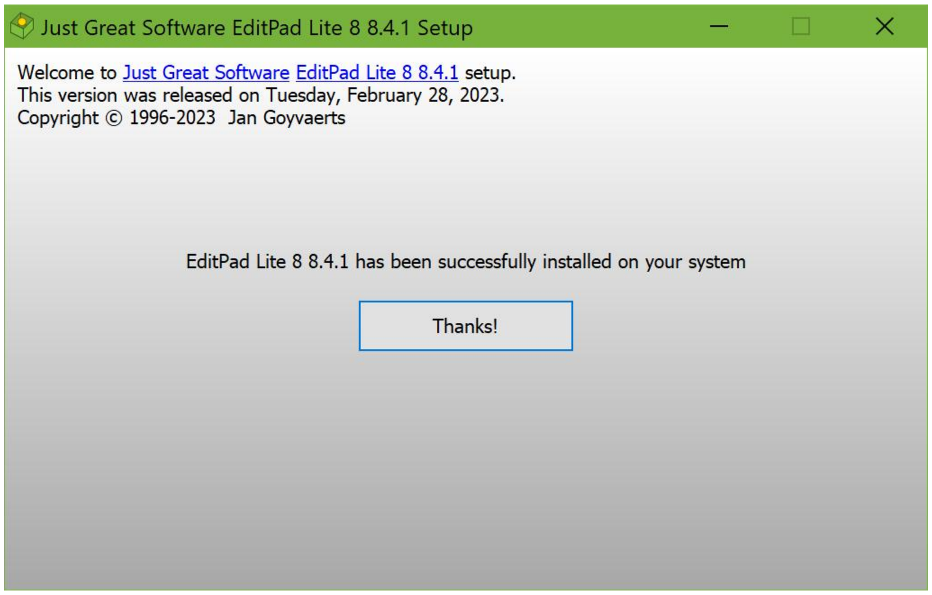
An Advanced Installation also includes the component selection step. This is the same component selection that can also be accessed by clicking on Custom Installation in the welcome screen. Only this time, more detail is shown.

First, all components are shown, even those that cannot be deselected.

Second, a checkbox labeled "Full Detail" is now visible. When ticked, the component list will expand to show all files that the component contains and the folders into which they will be copied. This allows advanced users to check whether the application to be installed will be using system folders as a trash can or not.

When the component selection is complete, all technical questions have been asked. The setup will then continue as if the "Immediate Installation" button was clicked on the welcome screen. So next up is the license agreement.

Guided Tour: Thanks


Just Great Software EditPad Lite 8 8.4.1 Setup

Welcome to Just Great Software EditPad Lite 8 8.4.1 setup.
This version was released on Tuesday, February 28, 2023.
Copyright © 1996-2023 Jan Goyvaerts

EditPad Lite 8 8.4.1 has been successfully installed on your system

That's all folks!

After accepting the license agreement and providing identity information as needed, DeployMaster will install the package. During the installation, several progress meters will show that the installation is proceeding. However, if the setup is only a couple of megabytes in size, installation will happen so fast that the "Thank You" screen will show up almost immediately.

If certain files which were in use by the system during the installation need to be updated then DeployMaster will inform the user that the system should be rebooted. Otherwise, it will just say "Thanks".

Clicking the Thanks button will show the Start Menu folder and/or launch the installed application if configured on the Finished page and then close the installer. The user can skip the finishing step by closing the installer by clicking the X button in the caption bar.

Guided Tour: Abort

If the Abort button is clicked any time during the installation, the setup program quits immediately.

2. Installing from The Command Line

When running an installer generated by DeployMaster, several parameters can be passed on the command line that affect how the installer is run.

Silent Installations

A setup package built with DeployMaster can be installed silently. That is, DeployMaster will do its job without asking any questions and without displaying anything on the screen. It also won't run the application after installing, if you specified that on the Finished tab. This is useful to install software in an automated fashion over a network.

To do this, simply specify the `/s` or `/silent` parameter on the command line. E.g.: `setup.exe /silent`.

Note that DeployMaster cannot suppress questions and messages from 3rd party installers. You'll have to configure those to run silently if you want your users to be able to install your software silently.

The uninstaller also recognizes the `/s` and `/silent` parameters to perform a silent uninstallation: `undeploy.exe /silent`.

If you turned on "install for all users" or "require admin rights" on the Project page then the installer and uninstaller will trigger a black screen security prompt on Windows Vista and later if you run them without administrator privileges. To avoid this (and thus make the process truly silent) you need to launch a silent installation from a command prompt or other process that is already running as administrator. Tools designed for automated software distribution generally do this since most installers require admin rights.

Suppress Desktop Icons

When software is deployed automatically users sometimes wish to suppress creating desktop icons. This can be done with the `/nodesktop` command line parameter. This parameter tells the installer not to put anything on the user's desktop. `/nodesktop` can be used with or without the `/silent` parameter.

Force 32-bit Installation

If an installer was built with "32-bit and 64-bit application for 32-bit and 64-bit Windows" selected on the Platform page, you can use the `/32` command line parameter to force the 32-bit version of the installer to run on 64-bit Windows. This can be useful to create a 32-bit portable installation for use on other computers that are running 32-bit Windows while your own PC is running 64-bit Windows.

Change The Temporary Files Folder

The setup.exe that DeployMaster generates is a stub with the installer and the files to be installed as its payload. When the stub runs, it extracts the actual installer into the temporary files folder and executes it. Only then does the installer appear on the screen. This will not work if a Software Restriction Policy blocks executables in the temporary files folder. To allow the installer to run, pass the `/temp` command line parameter followed by the path to a folder that can be written to and that software can be run from, e.g.: `setup.exe /temp "C:\My Folder\"`.

Change The Installation Folder

The `/appfolder`, `/appcommonfolder`, `/appmenu`, and `/userdata` command line parameters can be used to change the actual folders used for files placed under %APPFOLDER%, %APPCOMMONFOLDER%, %APPMENU%, and %USERDATA% on the Files page. If your installer uses a support DLL to define additional installation folders, then those folders can also be changed by using the name of the folder as a command line parameter. The purpose of these parameters is to allow the installation folder to be changed when using the `/silent` command line parameter. If the software was previously installed, using these command line parameters changes the installation folder, just like the Advanced Installation button can be used to change the folders of an existing installation.

These command line parameters can be used without `/silent`. In that case, when using the Advanced Installation button, the default folders will be those specified on the command line. Selecting different folders via Advanced Installation overrides the command line parameters.

3. Deployment Log

During the installation, DeployMaster saves its actions into a log file. This information is needed when the user wants to upgrade or uninstall your software.

However, your application can also use this log to see which components and files have actually been installed, so it can adjust its interface appropriately. It is a plain text file, which makes it easy for both the user and your application to inspect what DeployMaster did to the system.

The log is saved under %APPFOLDER%\Deploy.log by default. If the file already exists, a number is appended and increased until a non-existing filename is found (e.g.:Deploy3.log). This can happen if the user installs several programs into the same folder. In any case, your application can locate the log by opening the HKEY_LOCAL_MACHINE\Software\JGsoft\DeployIT registry key and reading the value that carries the name of your application, as you have specified in the Project section. It is a string value indicating the filename the log was written to.

The log file is organized with a text file, with each piece of information placed on its own line. Lines are separated by CRLF.

The file begins with an arbitrary number of lines starting with a semicolon. These are comments that explain the purpose of the file to a curious user inspecting it with a text editor. These are the only comment lines allowed to appear in the file.

Then follows the log version number. DeployMaster 6.0.0 and later write log versions 3 and 4. DeployMaster 4.0.0 through 5.4.0 used versions 1 and 2. Older versions of DeployMaster only used version 1. If you encounter another number then the log was created by a more recent version of DeployMaster. In that case you cannot make any assumptions about the log's contents. Consult the documentation of the most recent version for any changes in the format.

Version 1 and version 3 files are encoded using the computer's default Ansi code page. These are only created by 32-bit installers targeting Windows 98 or ME (and possibly later versions of Windows). Version 2 and version 4 files are encoded using UTF-8, without a Unicode signature (byte order marker). These are created by 32-bit installers that do not target Windows 98 or ME and by 64-bit installers.

In version 3 and version 4 logs, the line after the version number indicates the type of installation. ALL means an installation for all users. ADMIN means an installation for the current user requiring admin rights. These two values make the uninstaller require admin rights. CURRENT means an installation for the current user without admin rights. This value allows uninstallation without admin rights.

Version 1 and version 2 logs do not have the line indicating the user type. DeployMaster 5.4.0 and prior could only install for all users. The remainder of the log is the same for all versions of the log.

The next line is the application's title in the form of "companyname appname versionnumber". Then follow year (4 digits), month and day of release, each on a separate line. This info is used to determine whether a newer, older or the same version is being installed when DeployMaster is run again. Companyname, appname and versionnumber are listed again, but each on a separate line this time. Except for appname, these may be blank (but they will always occupy a line).

The next six (versions 1 and 2) or seven (versions 3 and 4) lines contain text strings that are used by the uninstallation routine to display messages to the user. These strings can be changed on the Language page in the DeployMaster Builder.

After those six or seven lines, one line indicates the command that the uninstaller will execute before starting the uninstallation. This line may be blank.

Then DeployMaster stores the paths which the user could have changed in the Advanced Installation section of the installation program. The first line of this section indicates the number of available paths. For each path, three lines are then used: one line contains the path's identifier (e.g.: %APPFOLDER%), the second line is the description of the path the user sees in Advanced Installation, and the last line is the actual path. If the last line is preceded by an asterisk (e.g.: *C:\Windows), the folder is considered a shared one. Files placed in shared folders have their reference counts updated in `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs`.

The next line is a number indicating how many components there are in the installed package. Each component takes up two lines following this number. The first line is the component's caption, the second indicates the component's status. If the status contains an S, the component was explicitly selected by the user or by the package's default selection. If the status contains an R, the component was installed because it was required by another component that was installed too. Any other character in the status string should be ignored.

Then follows the list of files that were installed. Each file occupies one line and has its full path information. The list is terminated by three percentage signs (%%%) on a separate line.

Next are the created registry items. Dummy items (ones that were not created by the setup program but should be removed upon uninstallation) are listed too. Each value occupies a single line, starting with the full path of the registry key, then a tab character (0x09) and then the name of the value. The value itself is not saved to the log, even if it was not a dummy one. Again three percentage signs terminate the list.

File associations that were created or modified are also remembered. The first line is the file type extension, including the leading dot. Lines 2 and 3 are the old (the one present on the user's system, if any, before DeployMaster modified it) and new descriptions of that file type. Lines 4 and 5 are the old and new DefaultIcon values for this file type, being an executable and an icon index, separated by a comma. Except for the extension, all these lines may be empty. The next line will either be a single percentage sign (%) or the name of an action registered for this file type. If it is an action name, the second line is the old and the third line the new associated command line that will be executed when the user selects this action. Again, the next line will be a single % or the name of the next action. Repeat until you read a %. After a file type was terminated by a single %, the next line will either be the extension for the next file type (in that case, repeat the above), or three percentage signs (%%%) terminating the file type list.

Any data following the last three percentage signs should be ignored, which DeployMaster does too.

Index