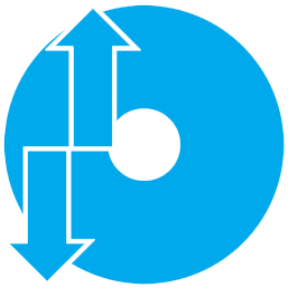
Contents

Please note that the information in this documentation always refer to the most recent version of CloneApp, which can be downloaded here.

1. Introducing CloneApp



1.1 What is CloneApp?

CloneApp is a small, **fully portable** Microsoft Windows utility that can be used to easily backup and restore application settings and Windows configurations. CloneApp can backup configurations stored in Windows directories, profile folders and the Windows registry.

1.2 What can you use it for?

As you can guess from its name, CloneApp can be used to copy and restore configuration files from applications and Windows components.

For instance, CloneApp can save Windows and program settings when Windows needs to be reinstalled or to backup applications configurations.

1.3 What makes CloneApp different from other backup software?

Other backup software make complete backups of all files of a program or even a complete Windows drive. CloneApp will **only** backup the configuration of an application, i.e. ".ini files" and information in the Windows directories and registry. This can be done quickly and normally does not require a lot of disk space.

With this mind, it should be clear that CloneApp is not a replacement for other backup software, since it serves a different purpose.

Info: In many ways, backing up ONLY the configuration files is also the more secure way: if your system becomes insecure due to a malware attack or user error, a fresh install is sometimes the only way to go. CloneApp can restore the "old" configuration when an app is freshly reinstalled.

1.4 System requirements

CloneApp can be used on any PC running **Microsoft Windows 7, Windows 8, Windows 8.1, Windows Server 2008-2012, Windows 10** (all editions, including 64-bit).

CloneApp is fully portable and stand-alone: no need to install it, just unpack and run. It uses few system resources: there are no minimum memory or hard drive requirements. It is not using any 3rd-party libraries nor has it any other dependency. CloneApp does not write anything to the registry.

1.5 About the project

- The initial release of CloneApp was on May 5th 2015 and version 2.0 came on April 5th 2018.
- It is currently not digitally signed.
- Since this is a one-man project, it relies on donations.
  If you want to contribute and support the development of CloneApp, you can send me a donation here.

If you appreciate CloneApp, please introduce it in your environment or social network, write about CloneApp on your blog's and help me develop CloneApp further.

Note: CloneApp contains no spyware or adware.

2. Using CloneApp

2.1 The user interface

The Main menu of CloneApp is the **Hamburger menu** in the top left corner, next to the CloneApp caption. This menu provides additional and advanced settings, which is discussed in Chapter 4 - CloneApp Settings.

The CloneApp interface is divided into three panes:

1. **Navigation tab:** The first lists main program functions such as backup or restore.
2. **App window:** The second lists all supported programs
3. **Status-, Log window:** The third lists log information

The Main menu of the **Navigation tab** (1.) is captioned with the name **Home.** This is also the first tab you will see, when starting CloneApp.

Next to the backup and restore buttons, you will also see a **Preview** button.

Note: The **Preview** feature is intended **for analysis ONLY.** No files are backed up during the process.

Under the **Preview** button, you will see also a **Select Installed** button. This function will check which of the installed plug-ins belongs to actual applications found on your system. Every detected application will be marked and reported in the **Status-, Log window**.

Tip: Press a letter on the keyboard to jump to the first program in the list that begins with that letter, e.g. pressing "C" jumps to "Calibre".

Under the Select Installed button, you will find the function to **Import Plug-ins.** *Please look into chapter 2.3 – Importing new applications, to learn more about this feature.*

Next to these submenus of the Home window, you will find in the Navigation tab also the menu **Options** and **Info.** *The menu Options is described in detail in Chapter 4, CloneApp – Settings.* The Info menu provides versioning information and a function to check for software updates. *See chapter 2.4 Checking for CloneApp updates to learn more about this feature.*

The **App window** (2.) provides information about the plug-in itself, e.g. plug-in name, author of the plug-in and lists all installed plug-ins.

The **Status-, Log window** (3.) logs all actions during the backup and restoration process. *Read more about this feature in the chapter 5.1 log operations.*

2.2 Backup and restore application settings

2.2.1 Backup application settings

Note: You should be **running CloneApp always with administrator rights.** You can still use it as a normal or restricted user, but Windows may prevent you from copying certain files.

First, select all apps you want to be backed up by ticking their checkboxes. If you are unsure which apps to select, you can use the function **Select Installed**.

Select Installed will check all CloneApp plug-ins for references to files, folder or registry entries. If a reference is found, CloneApp assumes the app is installed the corresponding checkbox ticked.

To **start a backup**, click the **Backup** button in the Home menu. CloneApp starts processing and export all files to the Clone Path defined under Options *(Read more about this feature in the chapter 4 - CloneApp Settings)*. The backup process itself is straightforward. It runs the process for each application that you have selected individually and logs the progress that it makes in the Status-, Log window.

2.2.2 Restore application settings

The same procedure from the backup process is used in the restoration process, but vice versa. To start a restoration, click on the **Restore** button in the Home menu. **The basis for the restoration process is the plug-in file itself and the restoration file cloneapp2.ini**, which is being created after each backup process and where all relevant settings are saved, e.g. ClonePath, clone mode and the confirmation mode *(Read more about this feature in the chapter 4.5 - Clone modes)*.

2.2.3 Background information about backup and restoration process

CloneApp runs the backup and restoration process very careful.

Default a dialog will be displayed if CloneApp cannot perform an action during the backup or restoration process. This might occur for instance, when a file cannot be copied, because an application is still running during the process or when folders cannot be copied because the the application is not installed and the main folders are not found.

In the last case, CloneApp will ask you whether the folders should be created. This could also happen, when an environment variable does not exist and CloneApp does not know where to copy and/or restore the files.

Read more about this process in chapter 4. - CloneApp Settings > Clone modes and how to handle dialog boxes and error messages in CloneApp.

CloneApp will log every action taken in the status window.

2.2.4 How can I restore a backup to another computer?

Just copy the CloneApp program folder including the CloneApp plug-ins and of course, the backup folder including the **cloneapp2.ini** to the machine where you want to restore the configuration.

CloneApp writes the path where the files have been backed up (Clone Path) also to the **cloneapp2.ini**

The username (which is part of the Clone Path) on the target machine you want to restore the configuration, could be different from the username on the source machine where the backup was made. In this case, CloneApp will return the error message *"Restoration Settings could not be loaded. Do you want to select the Restoration file?"* If you see this message, please select the restoration file manually.

2.3 Importing new applications

New plug-ins for CloneApp can be found here.

To import and install a new downloaded plug-in use the **Import Plug-in** link in the Home menu or **Hamburger menu > Import Plug-in.** *(Read more about this feature in the chapter **CloneApp rules**)*

2.4 Checking for CloneApp updates

The **Check for updates** feature checks whether a more recent version of CloneApp is available to download. To use the feature, your computer must have a connection to the internet. If your internet connection uses a proxy server, make sure your web browser connection settings are configured correctly.

To check for updates, go to the **About this App** tab and click on the **Version information** or use the link Check for updates in the **Hamburger menu**. A message box will popup, if a newer version is available.

3. CloneApp rules

CloneApp uses ".ini" files to determine which applications and Windows configuration files to backup and/or restore and how to do that. Since ".ini" files are plain text files, you can edit them with any text editor you like even plain old Notepad.

CloneApp comes with pre-configured plug-ins for dozens of popular applications. You can also write your own plug-ins for your specific applications. Of course we like to receive the plug-ins you create and add these to our list of plugins and extend CloneApp's functionality.

3.1 Structure of a plug-in file

A plug-in file consists of an Info section, a Variables section (optional) and a Files section.

The information in the Info section is used to show details of the plug-in to the user.

Info section syntax:

```
[Info]
Title=Name of the plug-in. Title is showed in the UI > Name field
Version=Shows with which Version a plug-in is compatible (Optional)
Description=Shows a short Description of what is being backed up
Author=Mirinsoft or http://www.mirinsoft.com
AuthorURL=Optional in URL format: http://www.mirinsoft.com
Warning=Display Warnings (Optional)
```

To illustrate the Info section of the plug-in to backup 7-zip:

```
[Info]
Title=7-Zip
Description=Backup 7-Zip Configuration from Registry.
Author=Mirinsoft
AuthorURL=http://www.mirinsoft.com
```

The entries in the Files section comprise the steps CloneApp takes to back up a particular app. Each step executes a command, like copying a file or folder or run a particular command

Files section syntax:

```
[Files]
File1=BackupFile|%AppData%\Example App\Configuration\config.ini
File2=BackupFolder|%AppData%\Example App\Configuration\
File3=BackupRegKey|
File4=RunFile|
File5=RunCommand|
File6=TaskKill|
File7=Detect|
```


The file entries must start at File1 and incremented for each subsequent entry. So the first line of the section should start with "File1=" and the second with "File2=" and so forth.
After the "Filex=" string comes the CloneApp command to be executed, e.g. **RunFile** followed by the path to the file/folder/registry Keys or command.

The CloneApp commands, like RunFile, are explained below.

Info: *The path entries are case-sensitive.*

To illustrate the Files section of the plug-in to backup Adguard:

```
[Files]
File1=BackupFile|%ProgramData%\Adguard\dbase.s3db
File2=BackupFile|%ProgramData%\Adguard\sfbr.s3db
File3=BackupRegKey|HKEY_LOCAL_MACHINE\SOFTWARE\Adguard
```

3.2 Supported variables

CloneApp works with the environment variables in Windows. The most common variables like *%ProgramFiles%, %AppData%* etc. are defined on every Windows version.

To see all environment variables configured on your system do the following:

Start > Run > cmd > Type the command ***set***

The **"%" (percent sign)** defines **Windows environment variables.**
Here is a list of the most common variables on Windows Vista and later systems.

Variable	Value
%ALLUSERSPROFILE%	C:\ProgramData
%APPDATA%	C:\Users\{username}\AppData\Roaming
%LOCALAPPDATA%	C:\Users\{username}\AppData\Local
%PROGRAMDATA%	C:\ProgramData
%PROGRAMFILES%	C:\Program Files
%PROGRAMFILES(X86)%	C:\Program Files (x86) (only in 64-bit version)
%ProgramW6432%	C:\Program Files (only in 64-bit version)
%PUBLIC%	C:\Users\Public
%SystemDrive%	C:
%SystemRoot%	C:\Windows
%USERPROFILE%	C:\Users\{username}
%WINDIR%	C:\Windows

The **"$" (dollar sign)** defines **custom/internal variables of CloneApp** based on Windows API.

Variable	Place
$AppClonePath$	Clone/Backup/Output folder of CloneApp
$Date$	Date stamp e.g. in Clone Path
$Documents$	C:\Users\{username}Documents
$AppData$	C:\Users\{username}\AppData\Roaming
$LocalAppData$	C:\Users\{username}\AppData\Local\Roaming
$ProgramFiles$ (only 32-bit version)	C:\Program Files (x86)
$FirefoxProfile$	Default Firefox Profile in AppData\Roaming

You can also define your own variables in a CloneApp plug-in file. These variables can then be used in different File commands.

```
[Variables]
MyDocuments=%UserProfile%\Documents
```

The variable **MyDocuments** can now be used in the [Files] section.

```
[Files]
File1=BackupFile|%MyDocuments%\config.ini
File2=BackupFile|%MyDocuments%\data.txt
```

3.3 Commands in the Files section

CloneApp's core is the customizable plug-in engine. This supports several commands, which will be explained in this chapter.

Note that the commands are case-insensitive.

In the description below the clone path refers to the ***Clone Path defined under the Options tab*** in CloneApp. Read more about this feature in the chapter **CloneApp Settings**.

3.3.1 Backup files

The BackupFile and BackupFile64 commands are used to backup a specific file to the clone path. Using wildcards multiple files can be copied in one command.

Syntax:
File<x>=BackupFile|<source file(s)>[attributes]
Info: *This command supports the attributes write-only and hidden.*

Examples:

Copy the file config.ini to the clone path
```
File1=BackupFile|%AppData%\Example App\Configuration\config.ini
```

Using wildcards copy all files in the Configuration folder to the clone path
```
File2=BackupFile|%AppData%\Example App\Configuration\*.*
```

Using wildcards copy only ".ini" files in the Configuration folder to the clone path
```
File3=BackupFile|%AppData%\Example App\Configuration\*.ini
```

Copy all files with the name "config" in the Configuration folder to the clone path
```
File4=BackupFile|%AppData%\Example App\Configuration\config.*
```

> ***Note:*** The x64-bit variant of the BackupFile command, has to be used when working with x64-bit environment variables e.g. %ProgramW6432%.

3.3.2 Backup folders

The BackupFolder and BackupFolder64 commands are used to backup **complete** directories to multiple new parent directories in the specified path.

Due to Windows file system restrictions folder names with special characters, e.g. * " ? are not allowed. CloneApp would try to generate a folder called Configuration2015*, but this is not possible. Therefore, we have to create our own custom folder Configuration2015.

Syntax:
File<x>=BackupFolder|<source folder>[|<destination folder>][<attributes>]

You can define custom output folders of your choice, when working with this command.
Info: *This command supports the attributes write-only and hidden.*

Examples:

Copy the folder %AppData%\Example App\Configuration to the clone path
```
File1=BackupFolder|%AppData%\ExampleApp\Configuration
```

Copy the folder to a custom output folder Configuration2015
```
File2=BackupFolder|%AppData%\ExampleApp\Configuration2015|Configuration2015
```

Copy using wildcards: in this example all folders whose name begin with "Configuration" will be copied. Instead of searching for the whole folder *Configuration2015*, we can also search for *Configuration**
```
File3=BackupFolder|%AppData%\Example App\Configuration*|Configuration2015
```

Copy using wildcards:
```
File4=BackupFolder|%AppData%\Example App\*Configuration*|Configuration2015
```

In example **File4** you can locate your search for phrases e.g. *Configuration*. This would backup every folder containing *Configuration* in its folder name and also all files in these folders.

In the last two cases you can combine this command also with file types and phrases.

When backing up an application which has different locations but the same folder name, this command splits the absolute path of each backed up folder. To illustrate:

Windows Themes

%LocalAppData%\Microsoft\Windows\Themes

The created output folder **Windows Themes** creates five other subfolders beginning with **%LocalAppData%\Microsoft\Windows\Themes\Themes**

Here the same, where instead of %LocalAppData%, the folder **%AppData%** will be created.
%AppData% \ Microsoft \ Windows \ Themes \ Themes

> ***Note:*** The x64-bit variant of the BackupFolder command, has to be used when working with x64-bit Environment variables e.g. %ProgramW6432%.

3.3.3 Backup registry keys

The **BackupRegKey** command is used to back up a complete registry key.

Syntax:
File<x>=**BackupRegKey**|<registry key>[|<destination folder>]

Examples:

Backup the registry key HKEY_CURRENT_USER\Software\ExampleApp
```
File1=BackupRegKey| HKEY_CURRENT_USER\Software\ExampleApp
```

A short registry key format is also supported, e.g. **HKCU** i.s.o. **HKEY_CURRENT_USER**
```
File2=BackupRegKey| HKCU\Software\ExampleApp
```

This command supports also the **creation of custom output folders**
```
File1=BackupRegKey|HKEY_CURRENT_USER\Software\ExampleApp|CustomFolder
```

This command creates a separate subfolder for each part of the registry key to handle duplicate registry keys with identical filenames. To illustrate:

7-Zip

HKEY_CURRENT_USER\Software\7-Zip
The created output folder **7-Zip** creates three other subfolders beginning with **HKEY_CURRENT_USER \ Software \ 7-Zip**

Likewise, the folder **HKEY_LOCAL_MACHINE** will be created, instead of HKEY_CURRENT_USER.
HKEY_LOCAL_MACHINE \ Software \ 7-Zip

3.3.4 Run files

1
2

The RunFile command is used to execute a specific program.

Syntax:
File<x>=RunFile|<program to run>?
Don't forget the "**?**" character, which executes the RunFile command.

Example:
```
File1=RunFile|%AppData%\Example App\Configuration\My Software.exe?
```

3.3.5 Run commands

The RunCommand is used to execute a specific command.

Syntax:
File<x>=RunCommand|<command and parameters>

Example:

Execute the "netsh advfirewall" command to export the current firewall settings to the backup directory of CloneApp.

```
File1=RunCommand|netsh advfirewall export $AppClonePath$\FirewallSettings.wfw"
```

The RunCommand supports the internal variable $AppClonePath$. If enabled, it uses the default clone path directory of CloneApp.

3.3.6 Kill tasks

The TaskKill command is used to terminate tasks by process id (PID) or image name.
Optional a hint can be showed by registering the **WARNING** command.

Syntax:
File<x>=TaskKill|<image name>[|WARNING]

Example:

Kill the FireFox browser instance and give a warning that the instance was terminated
```
File1=TaskKill|firefox.exe|WARNING
```

3.3.7 Detect

```
File1=Detect|HKEY_CURRENT_USER\Software\Microsoft\Office\14.0\Word
```
Or
```
File2=Detect|%AppData%\Microsoft\Word
```

Note: ONLY one Detect *command* is allowed

By default, the detection routine checks every reference to file, folder or registry key in a plug-in file and checks if the respective file, folder or registry key actually exists.

Optional you can use the **Detect** command. CloneApp will check only the specified file, folder or registry key exists. The result of the Detect command takes precedence over the detect function in CloneApp itself.

3.3.8 Delete files

The DeleteFile command is used to delete a specific file after or before a backup. Using wildcards multiple files can be deleted in one command. The syntax is the same as in the BackupFile command.

```
File1=DeleteFile|%AppData%\Example App\Configuration\config.ini
```

3.3.9 Delete folders

The DeleteFolder command is used to delete a complete folder after or before a back up.

```
File1=DeleteFolder|%AppData%\Example App\Configuration
```

3.4.0 Delete registry keys

The DeleteRegKey command is used to delete a specific registry key after or before a back up.

```
File1=DeleteRegKey|HKEY_CURRENT_USER\Software\ExampleApp
```

3.4 External plug-ins

CloneApp supports command-line options for external applications.

External plug-ins are based on the default plug-in file and the external application itself, which has both to be imported to the plug-in directory of CloneApp.

3.4.1 Example of an External plug-in

Info: This command has been updated with v2.00 to ***PlugEx.*** In versions below, the command works under the name ***ExternalPlug***

PlugEx

File1= **PlugEx|**Plug-ins\ProduKey\ProduKey.exe /stext **$AppClonePath$**\Product keys.txt

The command used is **PlugEx**, which executes ***NirSoft ProduKey*** and exports the product keys in a text file. As destination path, the internal **$AppClonePath$** variable is used.

3.4.2 Supported External plug-ins

- ✓ Nirsoft ProduKey, which save a list of product keys for applications and Windows e.g. Windows, Office, Adobe products etc. into a text file.
- ✓ Nirsoft MyUninstaller, which save a list of Installed applications into a text file.

4. CloneApp Settings

In the **Control Panel** tab, you can edit settings for CloneApp.

4.1 Clone Path

The most important setting is the backup destination path, called **Clone Path.** This is where the backed up files are going to be copied to and restored from. **CloneApp automatically creates the output/backup directories.**

You can also use an variable**/Dynamic Clone Path** e.g. %UserProfile%\Desktop\CloneApp (Backup)-**%Date%**\

As you see, the clone path supports also the variable **%Date%**, in the Format YYYY-MM-DD.

4.2 Log path

In the Control Panel tab, you will be also able to set a **Log Path** for CloneApp. All actions performed during a backup or restoration process are saved to the Log Path. These logs can be saved into a text file **right-click Status window** > **Save** or **Save as ...**

Note: You can also use a variable**/Dynamic Log Path** e.g. **%UserProfile%\Desktop**

4.3 Editor Path

The Editor Path option allows you to define a custom plug-in editor e.g. Notepad or Notepad++.
By default, the internal CloneApp editor **$AppEditor$** *(read more about this feature in the next chapter 4.3.1 – Internal Editor)* is preconfigured.

Note: You can also use a variable**/dynamic editor path** e.g.
%ProgramFiles%\Notepad++\notepad++.exe

4.3.1 Internal editor

CloneApp comes with a small and fast built-in plug-in editor, which is enabled by default.

To make sure that it is enabled, please check for the variable **$AppEditor$** entered in the editor path. The internal AppEditor allows you to easily edit available plug-in files and to save new plug-in files based on your editing.

4.4 7z Compression

7-Zip is an effective compression program. The 7z.exe program is used to compress, extract and update files through the command line. It provides superior compression. It is open-source. This makes it easy to obtain and use it in CloneApp.

If installed CloneApp will automatically detect the installation path and recommended command line options by clicking the **Browse** button, otherwise a dialog will be displayed for entering a custom path.

Use the **Compress** and **Decompress** textboxes to edit the command line options for compression and decompression.

4.5 Clone modes

4.5.1 Clone mode: clone apps in same or separate folder:

This option is set by default and let CloneApp **export all backed up files into one and same directory** which is defined under the clone path.

If you make a second click on this option, you will enable the second mode **clone apps in separate folder**. When using this option, CloneApp will create for each backed up application a separate directory. The naming of the folders will be on basis of each plug-in file.

Note: This option is with newer versions of CloneApp enabled by default.

If the plug-in file is called **Windows Favorites** but the application being backed up carries the name **Favorites,** then CloneApp will create a folder with the name "Windows Favorites" and export the Favorites folder itself as a sub folder of "Windows Favorites".

4.5.2 Confirmation mode: show or hide clone conflicts

The option **Confirmation Mode** shows information dialogs if a file or folder already exists and would be overwritten, e.g. if a folder does not exist and should be created or if the application being backed up is running and the files cannot be copied. This and such other "normal" Windows file conflicts will always be displayed if an error occurs.

If you make a second click on this option, you will enable the second mode, **Respond silent to all Clone conflicts.** While this option is enabled, CloneApp will respond to all clone and Windows conflicts automatically with "Yes" and hide all user interface and dialog boxes if an error occurs. In this case, files and folders will be automatically overwritten, folders will be created if they do not exist and running applications and processes will be ignored.

Note: This option is with newer versions of CloneApp enabled by default.

Note: By enabling this option, you will have to try NOT to evocate Windows conflicts e.g. a running application such as Mozilla Firefox will not be backed up or restored completely while it is opened, because CloneApp *(and also every other application or process)* cannot copy or replace files in usage.

4.6 Configuration files

CloneApp is a fully portable application, it saves its own settings in .ini files, which will be stored in the CloneApp installation or program **data** folder.

Next to .ini files, some other configuration files are created in the program folder of CloneApp. These are:

cloneapp.ini
This is the main configuration file, where all settings of CloneApp are stored. These settings can be modified via the **Options** tab.

cloneapp2.ini
This file is not present, until you run a backup process with CloneApp. The backup process creates this file, storing all settings used by the backup process. A copy of this file is also available in the folder where the files have been backed up.
When executing a restoration process, CloneApp will copy the **cloneapp2.ini** from the Clone Path to the **Data** folder of CloneApp. The restoration process will use the settings in this file as the basis for the restoration process.

~~**plug-ins.ini**~~
~~This file stores the status of the apps/plug-ins in CloneApp. The boolean values tells whether a plug-in is enabled (1=true) or disabled (0=false).~~

update.ini
This file stores versioning information and provides website news in the status bar of CloneApp.

theme.ini (customization file and NOT shipped with CloneApp)
This file overrides the UI and provides advanced tweaks for CloneApp.
Custom theme files can be found here.

4.7 Advanced settings

The hamburger menu provides some **Advanced Settings**, like themes, localizations and plug-in import functions. It can also show the available command-line options.

5. Advanced usage

5.1 Log operations

CloneApp is logging every file, folder, registry and command operation in the status- and log window. The log output is quite brief: its only marks the operation with an **<OK>** or **<Error>**

Successful operations are marked with <OK>

Faulty operations are marked with <Error>

There are **three scenarios** where you can get an **<Error>**

1. File, folder or registry key does not exist (for backup and restoration purposes)
2. You do not have administrator rights to access the file, folder or registry key
3. The application or an associated process being backed up is running.

5.2 Executing CloneApp from the command-line

You can use command-line parameters to tailor CloneApp to your requirements.

You can use CloneApp in batch files, scripting, shortcuts or the Windows Task Scheduler to automate CloneApp operations.

There are a number of parameters you can use when running CloneApp from the command-line.

Parameter	Explanation
/AUTO	Detect installed apps automatically and run backup
/B	Run automatically backup of selected applications while showing CloneApp UI
/SB	Run silently backup of selected applications without showing CloneApp UI and ONLY an status window on the bottom right corner of Windows

/CB "path_to_backup_folder"	Run silently backup of selected applications to custom folder e.g. /CB "%UserProfile%\Dropbox\CloneApp-Backup\"
/R	Run automatically restoration of selected applications while showing CloneApp UI
/SR	Run silently restoration of selected applications without showing CloneApp UI and only a status window on the bottom right corner of Windows

/CR "path_to_backup_folder"	Run silently restoration of selected applications from custom folder e.g. /CB "%UserProfile%\Dropbox\CloneApp-Backup\"
/P	Run preview of selected applications for backup and export them as a text-file to the directory defined under the Log Path > Options menu

5.3 Working with custom/personal plug-ins folder

CloneApp offers plug-in files for numerous Windows applications. You will certainly not need all of these plug-in files, since it is unlikely that you have installed every application supported by CloneApp.

CloneApp brings the function to **Select Installed** applications. This means that only installed applications will be shown in the **apps** window, when enabled.

To get a better overview and to show only certain or your own installed applications, you can do the following.

1. Goto **Hamburger/Main menu**
2. Select **Export**
3. Enable option **Apps to Personal folder**
4. Restart CloneApp

CloneApp will copy now all your selected applications to the custom **My Apps** directory.

Note: CloneApp will always load the **My Apps** folder first.
To load the default plug-ins directory, you just have to remove the **My Apps** folder.

6. Troubleshooting

Currently there are no known issues with running CloneApp.
If you have problems and need support, please visit the CloneApp Support Section of our website. You can also follow us on Facebook or Twitter.

7. Documentation Info

Last Update on 19/12/2018