ZePrA 12 Command Line Interface

This document describes the command line interface of ZePrA.

Preliminary notes

In the following text, parameter names are written in capital letters. These should not be used literally. Square brackets are used to indicate optional parts and are not part of the command line. The term "zcmd" stands for the executable of the ZePrA CLI.

In Unix like shells, e.g., MacOS terminal, file and configuration names and paths with spaces (or other special signs) in the name should be enclosed in quotation marks.

The CLI uses the settings of a ZePrA application installed on the same computer. If multiple major versions of ZePrA are installed the settings belonging to the same major version as the CLI are used, that means, the ZePrA 10 CLI uses settings from ZePrA 10, the ZePrA 11 CLI uses settings from ZePrA 11, etc.

To use the CLI on the same computer with a different user than the ZePrA GUI, make sure to enable the checkbox "Use ZePrA settings from the current user account in CLI" in the ZePrA "Preferences".

CLI executable

Under MacOS, the executable is:
ZePrA.app/Contents/MacOS/zcmd

Under Windows it is:
zcmd.exe
Located in the program folder beside the ZePrA.exe GUI executable.

Command line description

zcmd [OPTIONS] -c CONFIGURATION INPUT [-o OUTPUT] ...

Convert a file with a configuration.

CONFIGURATION – The name of the configuration to be used for the conversion. It must appear in the list obtained with listconfigurations.

OPTIONS – See section "Options".

INPUT – The file to be converted.

OUTPUT – The output file. If this is not specified, zcmd appends an underscore "_" and the configuration name to the original file name.

Example: zcmd –c 'SaveMaximum – ISOCoated' C:/MyFiles/test.pdf -o C:/MyFiles/test_converted.pdf

Note: If you want to convert multiple files in one call, append them to the command line, each file optionally followed by an "-o OUTPUT".

zcmd JOB_CONTROL_FILE
Processes a file with options defined in a **Job Control File**. The job control file should specify the file names including suffix for the input file and output file.

Note: This is one of two ways of using job control files with the CLI. The other way is specifying a job control file with the -j option (see below under "Options"). Job control files can be JSON or XML files.

zcmd listconfigurations [--filter=FILTER] [--bytags] [--untagged] [--sort=SORTMODE]
Display a list of available configurations on stdout. This list matches the configuration list displayed in the ZePrA GUI. If zcmd listconfigurations is used without options, all configurations are listed.
The following options are supported for this command:

--filter=FILTER - List only configuration matching the filter. The format of the filter string is described in section "Filter Format".

--bytags - Produce a list separated by tags, like in the ZePrA GUI. For each tag, a header line consisting of the tag name surrounded by 4 dashes is produced, followed by the list of configurations tagged with this tag.

--untagged - only valid if --bytags is specified. Produces the line "---- untagged ----" followed by configurations which have no tags. This makes sure that all configurations are listed at least once.

--sort=SORTMODE – produce a sorted list. SORTMODE can be "a-z" or "date".

Example: zcmd listconfigurations --bytags --untagged --sort=a-z

zcmd listdevicelinkprofiles
Display a list of available device link profiles on stdout.

zcmd listiccprofiles [--name=NAME] [--class=CLASS] [--dcs=DCS] [--pcs=PCS] [--format=json]
(ZePrA 11.0.1 and later) Display a list of available ICC profiles on stdout.
The following options are supported for this command:

--name=NAME – List only profiles with matching file names. NAME can contain wildcards (remember to use quotes in Unix like shells).

Example: zcmd listiccprofiles '--name=DEMO*'

--class=CLASS – List only profiles with the specified profile class.

Example: zcmd listiccprofiles --class=prtr

--dcs=DCS – List only profiles with the specified color space.

Example: zcmd listiccprofiles --dcs=CMYK

--pcs=PCS – List only profiles with the specified profile connection space.

Example: zcmd listiccprofiles --pcs=Lab

--format=json – If this option is specified, output is generated in JSON format. The output is a JSON array of dictionaries, one per profile. Each dictionary is a "Profile Info Dictionary" as described in the JobProperties.pdf. If this option is not specified, output is a list of profile file names.

zcmd listsmartlinkpresets [--name=NAME] [--srctype=SRCTYPE] [--dsttype=DSTTYPE] [--saveink] [--format=json]
(ZePrA 11.0.1 and later) Display a list of available SmartLink presets on stdout.
The following options are supported for this command:

--name=NAME – List only presets with matching names. NAME can contain wildcards (remember to use quotes in Unix like shells).

Example: zcmd listsmartlinkpresets '--name=*Exception*'

--srctype=SRCTYPE – List only presets for the specified source color space.

Example: zcmd listsmartlinkpresets –srctype=CMYK

--dcstype=DCS – List only presets for the specified target color space.

Example: zcmd listsmartlinkpresets --dsttype=CMYK

--saveink – List presets for ink saving. By default, presets for conversion are listed.

Example: zcmd listsmartlinkpresets --saveink

--format=json – If this option is specified, output is generated in JSON format. The output is a JSON array of dictionaries, one per preset. Each dictionary contains the following entries:

Name – The name of the preset.
SrcType – The source color space of the preset.
DstType – The target color space of the preset.

If this option is not specified, the output is a list of profile file names.

zcmd listspotcolorlibraries [--name=NAME] [--format=json]
(ZePrA 11.0.1 and later) Display a list of available spot color libraries on stdout.
The following options are supported for this command:

--name=NAME – List only libraries with matching names. NAME can contain wildcards (remember to use quotes in Unix like shells).

Example: zcmd listspotcolorlibraries '--name=PANTONE*'

--format=json – If this option is specified, output is generated in JSON format. The output is a JSON array of dictionaries, one per library. Each dictionary contains the following entries:

Name – The name of the library.
Count – The number of spot colors in the library.

If this option is not specified, output is a list of spot color library names.

zcmd listspotcolors [--name=NAME] [--lab=LAB] [--dE00=DELTA] [**--libraries**=LIBRARIES**] [--format=json]**
(ZePrA 11.0.1 and later) Display a list of available spot colors.
The following options are supported for this command:

--name=NAME – List only spot colors with matching names. NAME can contain wildcards (remember to use quotes in Unix like shells).

Example: zcmd listspotcolors --name='PANTONE 801 C'

--lab=LAB – List only spot colors with matching Lab values. "Matching" means that the Lab value is near the specified value within a dE 2000 tolerance of 1.0. The tolerance can be changed using the --dE00 option.

Example: zcmd listspotcolors --lab='60 70 70'

--dE00=DELTA – Specifies the dE 2000 tolerance for the –Lab option. The default is 1.0.

Example: zcmd listspotcolors --lab='60 70 70' --dE00=2

--libraries=LIBRARIES – List only spot colors from libraries with matching name.

Example: zcmd listspotcolors --libraries=MyLibrary

--format=json – If this option is specified, output is generated in JSON format. The output is a JSON array of dictionaries, one per spot color. Each dictionary contains the following entries:
Name – The name of the spot color.
Library – The name of the library where the spot color has been found.
Lab – The Lab value of the spot color.

If this option is not specified, output is a list where each line consists of the spot color name, the library name, and the Lab value, separated by comma.

Note: This command is primarily intended for a spot color search, with search criteria defined by the options. When used without options, it will display all available spot colors, which can be a very long list.

zcmd status
Display license status.

zcmd version
Display software version and build number.

zcmd export [--exportall] [--exportprofiles] [--exportprofileassignments] [--exportspotcolors] [CONFIGURATION ...] -o <OUTPUT>
Export one or more configurations into a user specified file name.

CONFIGURATION – The name of the configuration to export. Multiple configurations can be specified, each divided with a blank space.

OUTPUT – The output file. Use the suffix CCF, JSON or XML to define what type of export is desired. CCF is the internal ZePrA configuration file format while the two others are Job Control File Formats.

--exportall – Exports all configurations including ICC profiles.

--exportprofiles – Includes ICC profiles in the export.

--exportprofileassignments – Includes Profile assignments in the export.

--exportspotcolor – Includes spot color libraries in the export.

Note: The CCF file format is undocumented and should not be used for anything else than importing via the import command of the ZePrA GUI or zcmd – see next point.

zcmd import [--preview] INPUT
Configuration import from a CCF file.

During import, zcmd writes a log of taken actions on stdout. If the "\-\-preview" option is specified, only the log is displayed, and nothing will be done. The import procedure and logging are analogous to the configuration import dialog in the ZePrA GUI.

Note: Unlike the configuration import in the ZePrA GUI, if spot color libraries are imported, existing libraries with the same name will be overwritten.

zcmd spotcolorreport [-c CONFIGURATION...] [-j JOB_CONTROL_FILE...] FILE [FILE...] -o OUTPUT
zcmd spotcolorreport [-c CONFIGURATION...] [-j JOB_CONTROL_FILE...] -l LIBRARY [LIBRARY..] -o OUTPUT

Saves a spot color report. This is equivalent to the spot color report function in the ZePrA UI. Likewise, multiple configurations and multiple files/libraries can be specified to get a cumulative report.

Configurations are either specified by name using the -c option or by a job configuration file using the -j option. Any number of configurations or job configuration files may occur in the command line. Unlike as in the command line to convert a file, a job configuration file does not modify a configuration specified with the -c option. A job control file should define a configuration in a standalone way, that means, it should contain the "BaseConfiguration" entry.

When the output format is JSON, the resulting JSON file contains an array with one entry for each configuration and file/library. Each entry is a subset of the job properties format described in JobProperties.pdf containing the following entries:

_Values/File
_Values/Configuration
_Values/DstProfile
SpotColorProcessed
MinimizeChannels
DstProfileInfo

When a job control file is used, the configuration name listed in the report is a combination of the base configuration name and the job configuration file name, enclosed by square brackets. This is unlike as in the job report, where the job configuration file is stored under the "JobOptionsFile" key.

Example: zcmd spotcolorreport -c CMYK-7C test.pdf -o report.json

Creates a spot color report in JSON format for configuration "CMYK-7C" and the file "test.pdf".

Example: zcmd spotcolorreport -j jobconfiguration.json test.pdf -o report.json

Creates a spot color report in JSON format for the configuration defined in "jobconfiguration.json" and the file "test.pdf".

Example: zcmd spotcolorreport -c CMYK-7C -j jobconfiguration.json test.pdf another.pdf
-o report.json

Creates a combined spot color report in JSON format for the configuration "CMYK-7C" and the configuration defined in "jobconfiguration.json" and the files "test.pdf" and "another.pdf".

Options

--writereport[=yes]
Write a report into a PDF file of the same name as the output file with "_REPORT" appended. The report is the same as that obtained using the "Save Job Properties..." function in the ZePrA GUI.

--reportfile=REPORTFILE
Change the output file for the report. Depending on the suffix of REPORTFILE the report is written in PDF, HTML, TXT, XML or JSON format.

Beginning with ZePrA 11, this option may occur multiple times, so that it is possible to write the job report in different formats at the same time, e.g.PDF and JSON.

Example: --writereport=yes --reportfile=example1.pdf --reportfile=example2.json

Note: XML is written in the "legacy format" by default. To generate the recommended "new XML format" (ZePrA 9 and later), the report type should be set to "XML2".

--reporttype=TYPE
Further specifies the report format in the XML case. If TYPE is "XML2", the XML report is written in the recommended new XML format (ZePrA 9 and later). For backward compatibility, the default is the legacy XML format.

--continueaftererror=yes/no
Specifies if execution shall continue after a job error so that further files listed on the command line shall be processed. This is the default. If you disable this option, execution stops after a job error and "1" is returned to indicate that an error has occurred. This option is only relevant when converting multiple files in one call.

--threads=N
Specifies the maximum number of threads used for processing a single file. The default is 1.

This option should be used with care when multiple instances of zcmd are running at the same time. Typically, the overall number of threads should not exceed a machine-dependent limit to avoid performance loss.

-j JOB_CONTROL_FILE
Specifies a Job Control File to be used for processing the job. When using the -j option the job control file should not specify the input file and output file, because these must be part of the command line.

Note: This is one of two ways of using job control files with the CLI. The other way is specifying the job control file solely on the command line (see section “Command Line Description” above).

--progress
The progress status will be sent continuously to stdout with the prefix "Progress:" before each line. The calling program must scan stdout for lines with that prefix and display the content of the most recent line following the prefix in its GUI. Since these messages are not meant as a log, but only intended to display the status, lines before the most recent one can be ignored. The progress messages are the same as in the ZePrA GUI during job processing.

Filter Format

This section describes the format of the filter string specified in the --filter option for the listconfigurations command. The filter string is a comma separated list of expressions, each consisting of a key and a value, separated by "=":

KEY=VALUE

If KEY is the name of a configuration option, as described in ConfigurationOptions.pdf, the filter checks if the value of this option is identical to VALUE. Wildcards are allowed in VALUE. For options specifying profile paths, only the file name parts are compared.

Examples:

List configurations with target profile "ISOcoated_v2_eci.icc":

```
listconfigurations --filter=DstProfile=ISOcoated_v2_eci.icc
```

List configurations with target profile beginning with "ISO":

```
listconfigurations '--filter=DstProfile=ISO*'
```

(The quoting is necessary in Unix like shells to prevent wildcard interpretation).

In addition to configuration options, the following special KEYs are supported:

DstCS - The color space signature of the target profile is compared against VALUE.
DocumentCS - The color space signature of the document profile is compared against VALUE.
Tag – This checks if the configuration is tagged with VALUE.

Examples:

List configurations with target color space CMYK:

```
listconfigurations --filter=DstCS=CMYK
```

List configurations with target color space 7CLR:

```
listconfigurations --filter=DstCS=7CLR
```

List configurations tagged with "Proofing":

```
listconfigurations --filter=Tag=Proofing
```

Combining filters is possible. List configurations with target profile beginning with "ISO" and target color space CMYK:

```
listconfigurations '--filter=DstProfile=ISO*,DstCS=CMYK’
```

Error handling

If an error occurs, an informal line beginning with the word "Error:" is displayed on stderr. This line contains an error code. The following list contains the most common error codes:

9 (NotFound) - Something (a configuration, a profile...) could not be found
12 (FileNotOpen) - A file could not be opened for reading or writing
17 (FileReadFailed) - A read operation on a file failed
18 (FileWriteFailed) - A write operation on a file failed
19 (FormatErr) - File format error (indicates a corrupted file)
30 (InvalidLicense) - A function is not licensed
31 (InvalidICCTag) - indicates a corrupted ICC profile
32 (MissingICCTag) - indicates a corrupted ICC profile
33 (MissingICCTagData) - indicates a corrupted ICC profile
62 (PDFError) - indicates a corrupted PDF file
79 (SpotColorNotFound) - A spot color could not be resolved, and the configuration setting is "treat undefined spot colors as errors"
80 (SpotColorLibNotFound) - A spot color library could not be found

Other error codes may appear but are only meaningful for the support.

The return code of the zcmd call is "0" if no error occurs. The return code is "1" if a fatal error occurs. Job errors (like those listed above) are by default not treated as fatal errors, unless the option --continueaftererror is switched off in the command line, in which case "1" is returned.

Deprecated commands

The following commands are supported for backward compatibility but should not be used in new applications.

zcmd -a [options] <devicelinkprofile> input -o output
Convert a file with the given device link profile using the Auto-Setup "Normalize and convert colors to new output condition". The device link profile must appear in the list obtained with listdevicelinkprofiles.
zcmd --listconfigurations
zcmd --listdevicelinkprofiles
zcmd --status
zcmd --version