PxJob 4.0

0. General

PxJob is the command line based sister product of *PxEdit* which uses the same source code. It has most of the PxEdit functions available, and some functionality has been designed directly for it. The main idea is to automate and simplify routine processes.

PxJob may be used for validating `px` files (e.g. as a part of the publication process), make database reports, convert files to other formats, add, update and remove metadata, join or split tables, combine variables and handle the table languages.

PxJob reads source files and processes result files according to the command line. If the process encounters points where PxEdit asks for user input (e.g. replacing keyword or file replacing confirmation), PxJob will bypass these always answering *yes*. The actions will be written in a separate log file. If needed, the process may be shown in the Task Bar information bubble.

PxJob will always return a return code in the calling environment. A successful operation will always return `0`, values `11` and `12` mean failing to read or write a file correspondingly. Unsuccessful metadata file operation gives return codes `13` (no template file found), `14` (problem with the template file) or `15` (metadata injection `csv` file is not opened). If the return code is greater than `20`, there is an internal application error.

The command line parts are deliberately terse, the idea is to be able to put as much information in the command as possible.

0.1 Command files

PxJob may be run directly in the *Windows Command Prompt*. The recommended way is to use separate **command files** (`bat` or `cmd` files), which make it easy to run the same commands repeatedly. The command files are text files that contain command lines. They may be run either in the Windows Command Prompt or by double-clicking in the File Manager.

For example creating the following (e.g. with *Notepad*) file `D:\bat\Tablerun.bat`, which has just one command line:

```
D:\PxEdit\PxJob job=csv in=D:\dbase\Table.px out=D:\out\
```

When it is called in the Command Prompt (the file extension is not needed)

```
D:\bat\Tablerun
```

the file `D:\dbase\Table.px` is converted to a `csv` formatted file `Table.csv` in the directory `D:\out\`.

It is possible to include some programming logic in the command files.
For example the file `D:\bat\Makecsv.bat`:

```
@echo off
D:\PxEdit\PxJob job=csv in=%1 out=d:\out\
@pause
```

This command file can be used by writing the source file name after it in the command window. The command lines are not echoed on the screen (`@echo off`) and the command window stays open. If the source files are proper, the output `csv` files will be created in the directory `D:\out\`.

```
D:\bat\Makecsv D:\dbase\Status.px
D:\bat\Makecsv Q:\world\denmark\Turisme.px
```

The results for these commands are files `D:\out\Status.csv` and `D:\out\Turisme.csv`. The Command Prompt window uses DOS-ANSI character coding, which means that e.g. national characters may be problematic. PxJob makes the character conversions based on the system language or the table language setting.

0.2 Macro files

PxJob may use so called macro files as well. The macro file extension is `pxjob`.
When using macro files, the file name is the **only** input parameter.
The following command types are possible:

```
PxJob in=macrofile.pxjob
PxJob macrofile.pxjob
PxJob macrofile
```

The macro file consists of PxJob commands, each in a separate line (without the program call, of course). There may also be empty lines, comment lines (which start with a semicolon) and section headers (enclosed in square brackets). The section headers will be written in the log file.

The macro files should make it easier to use non-ascii characters in the command line. The macro files can also be slightly quicker to use because the interpreter will not be unloaded and loaded between the commands.

For example creating the following (e.g. with *Notepad*) file `D:\bat\Example.pxjob`, which consists of the following lines:

```
[csv conversion]
; the header above will be written in the log file
job=csv in=D:\dbase\Table.px out=D:\csv\
; the empty line below will be skipped
```

```
[html conversion]
job=htm in=D:\dbase\Table.px out=D:\html\
```

0.3 Installation

PxJob may be used from the PxEdit installation directory, using the common settings files. If needed, PxJob may be copied to a separate directory. When PxEdit is updated, the new `paq` file should be copied to this PxJob directory, too.

PxJob needs the following files:

`PxJob.exe`	application loader
`PxEdit_40.paq`	source code
`Dyalog170rt_unicode.dll`	*DyalogAPL* command interpreter

64 bit version needs the following files:

`PxJob32.exe`	application loader
`PxEdit64_40.paq`	source code
`Dyalog170_64rt_unicode.dll`	*DyalogAPL* command interpreter

The following files may be needed:

`PxEdit_main_40.ini`	main PxEdit settings file (e.g. for default values)
`PxEdit_lng_`***`cc`***`_40.ini`	language file for language code *cc*
`Zip.exe, Unzip.exe`	*Info-Zip* archiver
`Xdf_40.ini`	*CoSSI/XML* settings file

The language files are needed especially when making the table title for languages that are not included in PxJob. Included language codes are *da*, *de*, *en*, *es*, *fi*, *fr*, *it*, *kl*, *no*, *pt*, *ru*, *sl*, *sv* and *uk*.

Excel functionality needs an installed *Excel* application (usually *MS Office*).

1. Command line

The command consists of PxJob call and the command line:

`[`*`drive:\path\`*`]PxJob``{.exe}` *`command line`*

`drive``:`	PxJob installation drive (e.g. `D:`)
`\`*`path`*`\`	home directory path (e.g. `\PxEdit\`)
`PxJob`	command line interpreter (file extension is not needed)

NB: the PxJob call is usually left off from the commands in this document for simplicity.

The command line consists of the following parts:

{**job**}[**in**]{**out**}{**meta**}{**log**}{**err**}{**copy**}{**set**}{**path**}{-*options*}{!*switches*}

job	job type
in	source: file, file list, directory, or list file (mandatory)
out	target file or directory
meta	metadata file or directory or control file
log	log file
set	settings file
err	directory for discarded source files
copy	directory for passed source files
path	common directory path
-*options*	options with modifiers start with a dash
!*switches*	switches start with an exclamation mark

The command line parts are separated with spaces, the order of the parts is not specified.

1.1 Parameters

Parameters are short keywords followed by equal sign (=) and the modifier. If the modifier contains spaces, it has to be delimited by quotes ("). Parameters are used for job type, source and target files and possible other main settings.

job *batch type*

px	px file (default)
csv	csv file (either semicolon or tabulator separated text file)
exp	*eXplorer* file (txt)
htm	html file
report	database report (csv)
split	partial table (px, csv or xhtm)
sql	PxSQL macro file (sql)
translate	create or read translation files (translate)
txt	text table (either semicolon or tabulator-separated text file)
xls	Excel table
xml	*CoSSI/XML* file

job defines the output file type. If it is missing, the job is thought to be of type px, unless the first part of the command line is not recognised.

PxJob uses the system default decimal separator in csv, htm and txt outputs.
This may be changed with the q parameter if needed.

For example the commands

```
PxJob job=px in=..
PxJob px in=..
PxJob in=..
```

are interpreted as of type `px`, the following ones are of type `csv`:

```
PxJob job=csv...
PxJob csv
```

If the parameter contains file names or directories that have spaces or commas, the definition must be delimited with quotes. The directory names should end with a backslash for clarity, especially when the directory does not exist but it should be created.

```
out="D:\my files\Table.px"
in="D:\dbase\Population,2018.xls"
out=D:\out\results\
```

The directory or file names should contain the whole path name, e.g. `D:\dbase\Table.px`. It is possible, but not recommended, to use paths related to the run directory. For example, if PxJob is in the directory `D:\dbase\job`, the file `D:\dbase\job\in\New.px` may be defined with `.\in\New.px` and the file `D:\dbase\work\Old.px` with `..\work\Old.px`.

The directory path separator is backslash (`\`), but the slash character (`/`) is allowed as well:

```
D:\dbase\Population.px
D:/dbase/Population.px
```

in *source description (mandatory)*

The source files may be defined in many ways. A single file name may be given directly:

```
in=D:\dbase\Population.px
```

If the file name contains spaces or commas, it has to be given in quotes:

```
in="D:\dbase\Population, 2018.px"
```

A few files may be given as a comma-separated input file list. There should not be spaces around the comma. If all the files reside in the same directory, only the first file needs to have the full file path:

```
in=D:\dbase\Population.px,D:\dbase\Industry1.px
in=D:\dbase\Population.px,Industry1.px
```

Wildcards are allowed in the file names:

```
in=D:\dbase\*2019.px,Industry?.px
```

Usually it is convenient to put all the input files in one directory. In this case, the definition needs only to contain the directory name. If there is need to include all the sub-directories, the option `-s` should be used.

```
in=D:\dbase\
in=D:\dbase\ -s
```

As default, only `px` files will be read, other file types are defined with the option `-i`.

```
in=D:\dbase\ -ipx,csv
in=D:\dbase\ -i*
```

The option `-y` filters only the youngest, i.e. those files that have been recently updated. For example, selecting the files that have been updated during the last two hours:

```
in=D:\dbase\ -s -y0.2
```

Writing multiple files in the command line is tedious and prone to errors. A convenient way is to use simple text files with the extension `lst` (or `list`). The file contains one file with full path name per row. If only the first file name has the path name, it is used for each file name in the file.

```
in=D:\in\Database.lst
```

If all the input files have a common path, it will be used when the output directory is created with the option
`-s1`. All the lines that start with a semicolon are treated as comments.

The input file or directory may have the server definition, in which case the string starts with two backslashes:

```
in=\\server.stat.fi\dbase\Population.px
```

It is possible to use a separate web address, in which case the string starts with `http:`

```
in=http:\\pxnet.stat.fi\dbase\Population.px,Industry?.csv
```

out *output file or directory*

If there is no output definition, the output files will be written in the source directory, possibly replacing the original files. This may lead to unwanted results, especially when using the expunge switch `!x`.

The resulting file name will consist of the source file name and the output file extension.

A single output file definition is practical to use when there is only one source file, joining tables (`-j`) or archiving several files in a single file (`-z`). In a case when there are many input files but only one output file definition, only the last source file will be saved. The output files will not be saved if the user does not have writing permission for the output directory, though.

```
out=D:\out\New.px
out="D:\my files\Result.px"
```

The output file name may also be created from any of the keywords *CONTENTS*, *DESCRIPTION*, *MATRIX* or *TABLE-ID* by using the keyword as a file name, and a **colon** in front of it.

```
out=D:\out\:matrix
```

If the output directory does not exist, it will be created at the beginning of the batch job.

```
out=D:\out\new_data\
```

The output file will be saved with the file extension according to the job. For most of the jobs the default output format may be changed using the option `-o`. For example the default format for `xml` files is *xdf*, but it may be changed to *cals*:

```
job=xml in=D:\dbase\industry\ out=D:\out\industry\ -o2
```

As default, the output files will be written in lowercase, the spaces are replaced by underscores and national characters are replaced by their ascii equivalents. These changes may be prevented with the option `-b`.

Other options and switches allow e.g. setting the character conversions (`-c`), dot and dash code changes
(`-d`, `-f`), decimal and thousand separators (`-q`) and decimal precisions (`-p`).

log *log file*

The default name of the log file is `PxJob_`*timestamp*`.log` (e.g. `PxJob_20190404.log`). The log records are written in English at the end of the file, an existing log file is not deleted. If the log directory does not exist, PxJob will **not** try to create it.

```
log=D:\dbase\logs\Run-log.txt
```

The default file extension is `log`. Log file writing may be prevented with the switch `!o`.

meta *metadata or control file*

Metadata source can be defined in several ways. A single px or pxk file (which is similar to the px file without the data part) is handy for example in standardising keywords. The metadata file (template file) need **not** be fully validated.

```
meta=D:\dbase\meta\Template.pxk
```

Metadata parameter may define a directory, and in that case PxJob tries to find a corresponding template file, either with the same name or having the same file name beginning, and use that for the metadata source.

```
meta=D:\dbase\templates\
meta="D:\my files\templates\"
```

The recommended metadata update method is to use a special csv file for metadata injection (see 1.4).

```
meta=D:\dbase\meta\Control.csv
```

In some job types there are special control file or directory settings:

```
job=translate meta=D:\dbase\langs\
job=report meta=D:\dbase\meta\Report-control.csv
job=split meta=D:\dbase\meta\Partial.csv
job=csv meta=D:\dbase\meta\Keywords.csv -o3
meta=D:\statfin\meta\Filelist.csv !h
```

set *settings file*

PxJob tries to use the PxEdit main setup file (PxEdit_main_40.ini) in the startup directory, if found. The personal setup file (PxEdit_40.ini) is not used, though. If needed, the additional setup file may be defined with the set parameter, e.g. for using default metadata. The setup file needs only to have the needed sections (such as [defaults]).

```
set=D:\PxJob\Jobsettings.ini
set="D:\my files\Defaults.ini"
```

err *directory for discarded files*

Usually the non-severe metadata problems are just written to the log file, but otherwise the output files will be made. With the err parameter only fully valid px files will be handled, the others will be copied untouched to the defined directory. If the directory does not exist, it will be created. The source files are not removed as default, that can be done with the switch !x.

```
err=D:\dbase\problems\
```

```
err="D:\my files\problems\"
```

The error sensitivity may be changed back to normal with the option `-e`.

copy *directory for passed files*

All the source files will be copied to the defined directory untouched. If the `err` parameter is in use, only those source files that pass the metadata checking will be copied. If the directory does not exist, it will be created. The source files are not removed as default, that can be done with the switch `!x`.

```
copy=D:\dbase\ok\
copy="D:\my files\ok\"
```

path *mutual path setting for the command line*

This setting is meant just to simplify the command line. The parameter contains the common beginning of the file path. The file or directory definitions, which start with a backslash, will get the common path.
For example, the following settings are equivalent:

```
in=D:\dbase\in\File.px out=D:\dbase\out\ meta=D:\dbase\meta\
in=\in\File.px out=\out\ meta=\meta\ path=D:\dbase\
```

1.2 *Options*

Options are used for fine-tuning the commands. Options start with dash followed by one character and there may be a modifier.

Options that may be given in one group:

-a add and modify keywords and reorder variables
used with the control file given with the `meta` parameter
N.B. with `px` output, the `csv` file will always be handled as control file

-b bypass the standard file name conversions

`-b_`	leave spaces in the names [default: convert spaces to undescores]
`-b=`	leave character cases [default: convert to lower case]
`-b~`	leave national characters [default: convert to corresponding ASCII characters]
`-b/`	convert path separators to slashes in reports

modifiers may be combined, e.g. `-b~_`

-c character coding for the output file [default: `-c0`]

`-c0` WinANSI
`-c1` Unicode (UTF-8) (`-c`)
`-c2` ISO-8859
`-c10` WinANSI (DOS coding when reading, if *CHARSET* keyword is missing)
`-c11` Unicode (- " -)
`-c12` ISO-8859 (- " -)
`-c20` WinANSI (ignore the *CODEPAGE* setting when reading)
`-c21` Unicode (- " -)
`-c22` ISO-8859 (- " -)

`-d` dash code conversion when reading structural tables [default: fill item]
`-d0` convert to zero
`-d..` convert to two dots

dashes may be converted to any dot code or zero

`-e` error sensitivity with the `err` parameter [default: `-e0`]
`-e0` only fully valid files will pass
`-e1` normal level: metadata warnings are tolerated (`-e`)

`-f` fill item setting (missing or invalid data) [default: `..`]
`-f0` zero
`-f..` two dots

fill item may be any dot code or zero

`-h` service batches (tailored mainly for Statistics Finland's internal use)
`6` key figures setting (variable order: *Region*, *Information*, *Year*)
`7` variable-value listings for each input table
`8` create the *eXist* update file (`csv`)
`9` metadata listings of each directory (`json`)
`10` publishing pipeline (`!h`) without file name changing
`11` *StatFin* quality report
`12` publishing pipeline (`!h`), file name checking also for PxPro tables
`25` search interesting data values (default value is `25`, may be set with option `-v`)

`-k` create missing codes
`-k1` use the corresponding value texts of the main language (`-k`)
if there are unique space separated prefixes for all texts, use them
`-k2` use the corresponding value texts of the main language only
`-k3` create sequential zero-padded numeric codes
if all the corresponding value texts are numeric, use them
`-k4` create sequential zero-padded numeric codes only

only empty code lists will be created, i.e. if only some codes are empty, they will not be

replaced

`-k11..14` all empty codes for ragged lists will be replaced
`-k21..24` ragged code lists will be replaced wholly

-m add default metadata from the settings file
`-m1` add missing values only (`-m`)
`-m2` replace existing keywords, too

default metadata will be added after all other possible metadata operations

-o output file format [default: `-o0`]

`job=px`	`-o0`	`px`
	`-o1`	`pxk` (metadata file)
	`-o2`	`px` (sparse data format)
`job=csv`	`-o0`	semicolon separated
	`-o1`	tabulator separated
	`-o2`	comma separated
	`-o3`	metadata-`csv`, verbose output
	`-o4`	metadata-`csv`, repeated values included
	`-o5`	semicolon separated, file extension is `xls` (see also `!q`)
	`-o6`	semicolon separated csv, all variables in rows
`job=htm`	`-o0`	grey background colours for cells
	`-o1`	PxEdit background colours for cells
	`-o2`	no cell colouring
	`-o3`	html table
`job=report`	`-o0`	semicolon separated
	`-o1`	tabulator separated
	`-o2`	comma separated
`job=split`	`-o0`	`px` (see also `!n`)
	`-o1`	`csv`, semicolon separated
	`-o2`	`xls`
	`-o3`	`htm`, grey background colours
	`-o4`	`htm`, PxEdit background colours
	`-o5`	`htm`, no cell colouring
`job=sql`	`-o0`	make all INSERT macros
	`-o1`	make meta INSERT macros only
	`-o2`	make data INSERT macros only
	`-o3`	data part in `csv` format
	`-o4`	DROP, CREATE and all INSERT macros
	`-o5`	DROP, CREATE and meta INSERT macros
`job=translate`	`-o0`	all languages in one file
	`-o1`	languages in separate files (with the language code)
`job=xls`	`-o0`	`xls` (`xlsx`, if the table is too big for `xls` format)

	-o1	xlsx
job=xml	-o0	XDF formatted (_xdf.xml)
	-o1	CALS formatted (_cals.xml)
	-o2	KEYS formatted (_keys.xml)

the footnotes will be set as cell comments in xls output, if the !n switch is not in use

the footnotes for csv, htm, split and xls output may be limited to contain only *NOTE* keywords by adding 10 in the modifier, e.g.:

job=htm	-o**1**1	PxEdit background, only *NOTE* keywords with the !n switch
job=split	-o**1**2 xls	output, only *NOTE* keywords with the !n switch

-p the separator used in variable combining [default: /]

-p-	dash
-p::	two colons
-p" - "	string containing spaces

-q data formatting for xml/Cals output [default: -q.]

-q.	dot for decimals, no thousand separator (csv, htm and txt, too)
-q,	comma for decimals, no thousand separator (csv, htm and txt, too)
-g.,	dot for decimals, comma for thousands
-g,.	comma for decimals, dot for thousands
-q~	non-breaking space for thousands (-q.~)
-q,~	comma for decimals, non-breaking space for thousands
-q_	space for thousands (-q._)
-q,_	comma for decimals, space for thousands
-q'	apostrophe for thousands (-q.')
-q,'	comma for decimals, apostrophe for thousands

-r replace metadata (given by meta parameter) [default: -r0]

-r0	add only missing and suitable keywords
-r1	replace all possible keyword values (-r)
-r2	add missing keywords and replace variable names from template file (if the number of variables matches)
-r3	replace all possible keyword values and variable names

-s read sub-directories (if there is only one input directory)

-s1	reflect the input directory structure in output (-s)
-s2	read the sub-directories but write to the output directory only

-t title type and hierarchy (csv, htm, txt and xls) [default: -t0]

-t0	hierarchical value texts
-t3	hierarchical value codes and texts

`-t10` all value texts
`-t13` all value codes and texts

used only in `htm` and `txt` output:
`-t1` hierarchical value codes
`-t2` hierarchical value codes and texts combined
`-t11` all value codes
`-t12` all value codes and texts

used only in `htm` output:
`-t20` hierarchical value texts, one row column like in PxWin
variables ordered as in the table
`-t21` hierarchical value codes (- " -)
`-t22` hierarchical value codes and texts (- " -)
`-t30` hierarchical value texts, one row column like in PxWin
the last variable will be in columns only
`-t31` hierarchical value codes (- " -)
`-t32` hierarchical value codes and texts (- " -)

`-u` set the *LAST-UPDATED* keyword
`-u` use the current date
`-uvvvvkkpp_hh:mm` set the defined date and time (NB: the undescore)
`-u+n` set the date *n* days later than current date
`-u+n_hh:mm` set the date and time *n* days later than the current date

if the calculated date would be a weekend, it will be changed to the next Monday (see `-w`)

`-w` define weekend days (for `-u+` option) [default: `-w56`]
`-w56` Saturday and Sunday
`-w45` Friday and Saturday
`-w7` no weekend skipping

weekdays are numbered from Monday (`0`) to Sunday (`6`)

used only in `htm` output:
`-wnnn` set the default width for the first column in at *nnn* pixels

`-x` fine-tune the table title in other than `px` output [default: `-x0`]
`-x0` `px` style (*CONTENTS* and variables or *DESCRIPTION*)
`-x1` *CONTENTS* (without variable names) (`-x`)
`-x2` *DESCRIPTION*, or if it is missing, *CONTENTS*

`-y` source file freshness (*ddd{.hh{.mm}{+{{dd.}hh.}mm}*)
`-y7` files updated during last week

`-y1+5` files updated during the day but not during the last five minutes
`-y+2.0` files not updated during the last two hours
`-y14+2.0.0` files updated during the last two weeks but not during the last two days

if the `-y` option is being used with reporting and the output file exists, the new report records will be added to the existing report file instead of deleting it first

`-z` zip output files to archives
`-z1` all files will be zipped separately with the original file name (`-z`)
`-z2` all the files in the same directory will be zipped in the same archive
the archive file will be named after the directory name
`-z3` all the files in the same directory will be zipped in the same archive
the archive file will be named after the directory path name
(the path separators will be changed to underscores)

archiving uses the separate *Info-Zip* archiver (`Zip.exe`) which is included in the package
if the `out` parameter is used, all `-z` options work as `-z1`
if the `out` parameter defines a single output file and the option `-s`, is in use, the source directory structure will be copied in the archive file
if the `out` parameter defines a single output directory, all the files will be archived separately
if there is no `out` parameter, the files will be archived in the source directories

Options that must be given separately:

`-g` comma-separated list of variables to be grouped (combined)
`-gYear,Month` combine variables `Year` and `Month` (if they exist)
`-g"Pop,2018","Ad hoc"` quotes needed with spaces or commas
`-gSTUB` combine all row variables
`-gHEADING` combine all column variables

the variable names are given in the main language (names are not case sensitive)
the new variable name may be given with option `-v`
the combination character for value texts and codes may be set with option `-p` [default: `/`]
when combining two variables the first one has an existing *TIMEVAL* setting,
the new *TIMEVAL* is set if possible (see the switch `!t`)

`-i` comma-separated list of input file extensions [default: `-ipx`]
`-icsv,txt` read only `csv` and `txt` files
`-i*` read all files

`-j` joining options (correspond quite well with the PxEdit joining window options)
`-ja` replace all metadata (surpasses `-r`)
`-jb` merge new values (between) the original ones after joining (if possible)
`-jc` do not use codes for variable matching
`-je` exact text matching (case sensitive, use leading zeroes)

`-jf` bypass fill items
`-jl` do not create multilingual table if possible
`-jm` do not try to match the variable names
`-jn` group the files to be joined without the last underscore separated part (overrided by options `-j1..-j4`)
`-jo` use only original values (i.e. do not add new ones)
`-jr` do not replace metadata (overrided by option `-ja`)
`-js` do not sort the variable values (bypasses options `-jc` and `-je`)
`-jt` replace value texts
`-ju` join unique *SOURCE* keywords to a #-separated list
`-j1` group the files to be joined without the last character
`-j2` - " - without the last two characters
`-j3` - " - without the last three characters
`-j4` - " - without the last four characters

the single `-j` option joins tables with default settings
the joining options may be given in a single group (`-jcore`)

`-l` comma-separated language code list for desired output languages, base language as first
`-len,fi` output in English (as the base language) and Finnish

the languages should be available in the source table
with the structural tables, the system language will be set as the base language, if not separately set
if there is an underscore character after the option (e.g. `-len,fi_`), the table main language will be set from the file name, if possible (e.g. `table_en.xlsx`)
if there is a plus character after the option (e.g. `-l+`), the language code will be written next to the table title in the structural file (`csv`, `htm` or `xls` output)
PxJob already contains the title strings for the languages da, de, en, es, fi, fr, it, kl, no, pt, ru, sl, sv and uk

`-n` add a new variable in the table
`-nNew` add a new variable *New*
`-nNew,Uusi,Ny` comma-separated list for a multilingual table in language order
`-n"New name"` quotes needed with spaces or commas

the new variable value text will be the file name without path or extension
spaces will be converted to underscores

the new value text may also be given separated by a semicolon:
`-nNew;value` add a new variable *New* with the value text *value*
`-nNew,Uusi,Ny;value,arvo,väder`
comma-separated lists for multilingual tables in language order
`-nNew;CONTENTS` add a new variable *New* with the value text from the keyword contents

-v new variable name in variable combining (see option `-g`)

`-vTime`	set combined variable name as *Time*
`-vTime,Aika,Tid`	comma-separated list for a multilingual table in language order
`-v"New time"`	quotes needed with spaces or commas

1.3 *Switches*

Switches are options with two values, they start with an exclamation mark and there are no modifiers. All the switches may be grouped (`!esx`).

!a read all Excel sheets
!A skip non-structural sheets without error message
!b bypass the default string input conversion from DOS to Unicode
!c combine codes to value texts (`csv` and `xls`)
!d delete variables with only one value
!f file reading, fill items as zeroes

`job=report`	always write the file name in the report
`job=px`	read all source files even though there is a file filter in the control file
`job=csv`	convert all fill items to zeroes
`job=htm`	-"-
`job=txt`	-"-

!g add the language code at the end of the monolingual file name
if the `-j1` option is being used the first character of the main language code will be added in capitals at the end of the file name
!h database publishing pipeline settings in Statistics Finland, e.g.:
MATRIX keyword
file name checking (the allowed name list is given with the `meta` parameter)
`csv` and `xls`: add hierarchy codes to the structural table
!i show job progress information in the Task Bar balloon
!k keep the old file timestamp if possible
!l use system language for character conversion
!m add metadata (the keyword block) to the structural table (`csv` and `xls`)
!n add footnotes to the output (`csv`, `htm` and `xls`)
do not copy the missing keywords from the base language (`px`)
!o do not write to the log file
!p save using the screen decimal precisions (not for `px` files)
uses *SHOWDECIMALS* and *PRECISION*s instead of *DECIMALS*
!q `px` files: quick file copying
first the metadata is manipulated and written to the output file
then the data part will be copied from the source file untouched (without checking)
`csv` files: save the file with both `csv` and `xls` extensions (with `-o5` option)
!s skip the PxEdit main settings file (does not affect the `set` parameter)
!t try to set the *TIMEVAL* keyword when combining variables
!u set the file timestamp from *LAST-UPDATED*
!v output file validation (only for `px` files, needs the `out` parameter)

the output file will not be replaced, if it is newer than the source file
not with joining or archiving (`-j` or `-z`)

!w write other than source files to the output directory
only with different source and target directories

!x expunge (remove) the source files (use with caution)

!y save changed tables only

!z convert BIG5 coded texts to Unicode

1.4 *Control files*

The control files are semicolon- or tabulator-separated fields containing text files (`csv` files).
The leading and trailing spaces are removed from the fields. The text comparisons are not case sensitive.

metadata injection

The first row contains the column headers and the following rows contain the corresponding control fields. Each row will be handled separately, empty fields will be skipped and will thus not be taken into account.
The column order is free.

The following columns are possible:

<keyword>	the new contents for the specified px keyword
languagecode	specific language code
replacetext	the text string to be replaced
filename	the file name (may include part of the path, no `px` extension needed)
variablename	variable name for variable and value-specific keywords
valuetext	value text (or code) for value-specific keywords
code	value code
<variable>	value text, code, or * for cell-specific keywords
STUB	row variables (comma-separated list, quoted if necessary)
HEADING	column variables (- " -)

Table-specific keyword injection or changing:
<keyword> {<keyword>,...} {replacecode} {languagecode} {filename}

Variable-specific keyword injection or changing:
<keyword> {<keyword>,...} variablename {replacecode} {languagecode} {filename}

Value-specific keyword injection or changing:
<keyword> {<keyword>,...} variablename valuetext {replacecode} {languagecode} {filename}

Cell-specific keyword injection or changing:
<variable> {<variable>,...} {replacecode} {languagecode} {filename}

Table pivoting (reordering variables):

STUB HEADING {languagecode} {filename}
the table should contain these, and only these variables

Setting row or column variables:

STUB {languagecode} {filename}
HEADING {languagecode} {filename}

Changing the variable name:

variablename replacetext {languagecode} {filename}

Changing the value text for a variable:

variablename valuetext replacetext {languagecode} {filename}
variablename valuetext code {languagecode} {filename}

Changing or setting the value code for a variable:

variablename code replacetext {languagecode} {filename}
variablename code valuetext {languagecode} {filename}

Every row is regarded as a separate update instruction (either add, change or delete command) and there may be several keywords of the same level. The input fields are checked according to the keyword type (not as strictly as when reading the `px` table). The existing keyword contents are not replaced as default, this can be changed with option `-r`. Variable name, value text or value codes are replaced, though. Keyword contents are removed with the update value **~**.

The keywords *CODES*, *DATA*, *HIERARCHIES*, *HIERARCHYNAMES*, *HIERARCHYLEVELS*, *HIERARCHYLEVELSOPEN*, *KEYS*, *LANGUAGES*, *PARTITIONED* and *VALUES* are not handled.

The control file may be created from the table metadata either with *File*|*Save to*|*PxJob-csv* selection in PxEdit or by using the `csv` job type in PxJob with output parametres `-o3` (verbose output) or `-o4` (repeated values included). The `csv` job may use another `csv` file with the meta parameter to filter the output, the filtering `csv` file uses the same format as the report control file

The contents for the *TIMEVAL* keyword may be either a valid *TLIST* sentence or just the used frequency code `A`, `Q`, `M` or `H`. PxJob always checks if the *TIMEVAL* setting may be used for the corresponding variable.

All the values for the value-specific keyword may be denoted by the special character *.

It is possible to copy some keyword values to another. The following alternatives are possible:

column header	content text
CODES	*VALUES*
NOTEX	*NOTE*
NOTE	*NOTEX*
VALUENOTEX	*VALUENOTE*

VALUENOTE	*VALUENOTEX*
CELLNOTEX	*CELLNOTE*, *VALUENOTE* or *VALUENOTEX*
CELLNOTE	*CELLNOTEX*, *VALUENOTE* or *VALUENOTEX*

When the valuenote is copied to a cellnote, the first found VALUENOTE/X value is expanded to all given variable values.

The control table may be pivoted, then the headers will be in the first column and their control fields are in the corresponding rows.

creating partial tables with `split` job

The first row of the control file contains the column headers and the columns contain the corresponding control fields. Every row is handled separately, empty fields will be skipped. The column order is free. The variable order will be the same as in the definition, all other found variables will be placed at the end of the row variable list.

The following columns are possible:

STUB	row variables (comma-separated list, quoted if necessary)
HEADING	column variables (- " -)
languagecode	language code (if missing, the table base language will be used)
takevalues	the number of values taken from the start (or end, if negative) of the list
skipvalue	the value which is not in the result (may be several)
withvalue	values in the selection column order (default: all values)
<empty>	equals to *withvalue*

STUB and *HEADING* are mandatory.

control file for reporting

The report job may use a `csv` control file via `meta` parameter. It consists of at least two columns. The first column contains the level code:

`0`	general code
`1`	table-specific keyword
`2`	variable-specific keyword
`3`	value-specific keyword
`4`	cell-specific keyword

The second column contains either the control code or the keyword (which cannot be *CODES*, *HEADING*, *KEYS*, *STUB* or *VALUES*). *VARIABLES* 'keyword' means the combination of *STUB* and *HEADING* keywords.

The following control words are recognised:

filename	the file name for the table
filepath	the directory path for the table

pathname	the combined directory path and file name for the table
filecreate	the file creation timestamp
fileupdate	the file updating timestamp
filesize	the file size in bytes
tablesize	table size: (row variables)x(column variables)=number of cells
languagecode	language filter (each language in own row)
variable	variable filter (each variable in own row)
value	value filter
datacells	number of data cells
datanumbers	number of genuine numbers
datazeroes	number of zeroes
datadashes	number of dash codes
datadotcodes	number of dot codes
datadots1 .. 7	number of 1-7 dot codes
datamin	smallest data value
datamax	largest data value
datamean	data average

The control words that start with *data* cause the report process to read the data part, which may slow down the process remarkably. If the data value headers have the percent sign at the end (e.g. *datadotcodes%*), its value will be the percentage calculated with the total number of cells.

The filter controls and keywords may have additional elements, which will delimit the reported tables and/or information (these elements are not case-sensitive):

languagecode	only denoted languages in separate rows
variable	only the tables containing the listed variables
value	only the tables containing the listed value texts or codes
content	only those keywords containing (part of) the given contents
`0;languagecode;en`	only English metadata
`0;variable;info`	tables containing the variable *Info*
`0;value;Espoo;049`	tables containing the value text *Espoo* or value code *049*
`0;content;HREF;htm`	metadata containing links

If there is no control file used in reporting, the default report will have the columns *pathname*, *filesize*, *fileupdate*, *tablesize*, *languagecode*, *VARIABLES* and mandatory keywords.

The control file may be created with *Edit*|*Database*|*Report* in PxEdit.

The value texts may be listed with the variable-specific control code *VALUES*, which is available only with PxJob.

control files for service runs

TABLEID run (`-h4`) the first column is table name, the second is contents.

VARIABLE-ID run (`-h5`) the columns are table name, variable and contents.

In database publishing (`!h`) the control file contains the allowed file names.

2. Table validation

A simple table validation can be made with the command:

```
in=D:\in\Table.px log=D:\logs\Batch.log
```

The file `Table.px` will be opened, its metadata integrity is checked, the possible small metadata errors will be repaired and it will be saved back in standardised format. All actions will be recorded in the log file `Batch.log`.

It might be easier for the publication process to utilise a separate directory for the validation. Validation may be run frequently with the command:

```
in=D:\in\source\ out=D:\out\ err=D:\probs\ -e1
```

All the `px` files will be read from the directory `D:\in\source`, they are validated and the result files are saved in the directory `D:\out\`. The sensitivity option `-e1` makes it possible to save all valid `px` files, and only those files which cannot be processed will be copied to the directory `D:\probs\`. If the option is missing (or is set as `-e0`), all files that have metadata problems would be copied to the error directory.

When there is need to read and write the full directory structure, the option `-s` becomes handy:

```
in=D:\in\ out=D:\out\ err=D:\probs\ -se1
```

If there is need to add standard metadata to the tables (e.g. *SOURCE* and *COPYRIGHT*), the old method is to use a separate `pxk` file for it:

```
in=.. out=.. err=.. log=.. -e1 meta=D:\meta\Template.pxk
```

The recommended way is to use a `csv` control file for this purpose:

```
in=.. out=.. err=.. log=.. -e1 meta=D:\meta\Template.csv
```

When the only purpose is to check that the metadata in `px` files for publication are valid, the error sensitivity may be set high:

```
in=.. out=.. err=.. copy=D:\ok\ log=.. !qx
```

The valid files will be copied to the directory D:\ok and they will be deleted in the source directory (!x). The data parts will be moved without checking (!q), this will speed up the process remarkably.

3. File conversions

3.1 Conversion to px-format

PxEdit and PxJob applications have originally been made for px-file manipulations and creating them from different sources. In addition to px files PxJob accepts so called structural tables (csv, txt or other recognisable text file formats). If there is *Excel* installed, reading and writing of xls and xlsx files is possible. The structural files do not have much metadata apart from the table title, variable names and value texts and codes. PxJob saves the px files even without all mandatory metadata for px file format. Other px family applications may not be able to open them, though. Metadata can be enriched using template files, joining tables, using control files or manually with PxEdit.

The output format can be changed with the option -o as a pxk file (metadata template file, px file without data part) or as sparse matrix (so called *KEYS* format) for huge data parts with plenty of rows containing only zeroes.

PxJob reads different character codings (DOS-ANSI, Win-ANSI or several Unicode codings). The files will be saved either as in WinANSI or UTF-8 (Unicode) type. The preferred coding may be set with the option -c.

3.2 Conversion to structural table (csv, txt and xls)

The tables will be saved as structural files. The way in which the value texts and codes will be shown is changed with the option -t. Footnotes may be added under the table with the switch !n. The decimal precision may be set to be the same as the screen precision with the switch !p.

Csv files are simple text files, but they can be used for transferring tables to different spreadsheet programs (e.g. *LibreOffice*), the field separator may be changed with the option -o.

Saving to Excel files (xls and xlsx) is possible, if *Excel* has been installed. The output option -o may be used to set the output type, and it is possible to add the footnotes as cell comments, too.

3.3 Other conversion types (htm, xml and sql)

Htm files are mainly used with web browsers. It is not recommended to make html files from big tables. The output option -o may be used for setting the cell background colouring, either grey scaled or the same as in the PxEdit table windows. The title option -t may be used for setting all the row variables in a one intended column.

XML output formats are made according to Statistics Finland's *CoSSI-XML* definition. Output formats may be either *XDF*, which is suitable for saving huge tables, *Cals* which is used mainly in a publishing process (e.g. in the *ArborText* product used within Statistics Finland) and *Keys*, suitable for sparse matrices. It is not recommended to make *Cals* output from big tables, because the file size may grow rapidly.

The SQL output files are made according to Statistics Greenland's PxSQL model.

4. Metadata handling

4.1 Metadata injection using control files

The recommended way to manipulate metadata is to use separate csv control files.
The next examples use these command line settings:

```
in=... meta=Control.csv
```

Each control file for the examples is shown as semicolon-separated text. The control file may be used to manipulate practically all the table metadata, but a single row should have only information on keywords that share the same level (other keyword fields should be empty). The leading and trailing spaces are removed from the fields. The keyword contents are not checked thoroughly, so extra care is needed in control file editing. The metadata will be set if there is no previous value for the keyword, the existing metadata will be replaced with the option -r.

```
NOTE;SHOWDECIMALS;COPYRIGHT;PRESTEXT
footnote;2;yes;~
```

The table-specific keyword *NOTE* will have the contents footnote, *SHOWDECIMALS* will become 2 and *COPYRIGHT* will be set as YES, if those keywords do not have content before. The option -r will always replace the existing values, which may not always be desired, though.
The *PRESTEXT* keyword will be removed, when found (~).

```
DESCRIPTION;CONTENTS;replacetext;filename
province;province;county;010*
```

For all the tables the names that start with 010, the texts in the keywords *DESCRIPTION* and *CONTENTS* will have the text *county* changed to *province*.

```
NOTE;MAP;TIMEVAL;variablename
footnote;Finland_municipality; ;region
 ; ;Q;quarter
```

The contents for the keyword *NOTE* for the variable *region* (or *Region* etc.) becomes footnote, the

MAP keyword for the same variable becomes `Finland_municipality` and the TIMEVAL keyword for the variable *Quarter* will be set as quarter type (`Q`), if it is possible. The variable names will be checked in the base language of the multilingual table, if the language is not separately set.

```
NOTE;variablename;languagecode
alaviite;ikä;fi
```

Sets the Finnish footnote for the variable *Ikä* in a multilingual table.

```
PRECISION;VALUENOTE;variablename;valuetext;code
1; ;Information;percentage;
 ;preliminary data;Year; ;2020
```

Sets the decimal precision for the value *percentage* in the variable *Information* and the *VALUENOTE* keyword with the value code in the variable *Year*.

```
CELLNOTE;Year;Region;Age;Gender;Status;languagecode
wonder if any;2018;*;100;total;married;en
```

Sets the cell-specific keyword in the table having variables *Year*, *Region*, *Age*, *Gender* and *Status*, and only them (not necessarily in this order). The asterisk denotes all the variable values.

4.2 Using `px` or `pxk` files as templates

If the `meta` parameter defines a single `px` or `pxk` file (a template file), PxJob reads its metadata and copies the suitable ones to the current table. The template file need not be a fully functional px file, only the keywords *CHARSET*, *STUB* and *HEADING* have to be set.

If the `meta` parameter defines a single directory, PxJob tries to find a suitable template file within the directory. If there is no file with the same name (without the extension), the file names are compared without the possible underscore-separated ending parts. If a corresponding template file is found, it will be processed as a normal template file.

When the template file is opened, it will be checked the same way as when opening a normal `px` file.

4.3 Default values

The option `-m` makes PxJob fetch the default keyword settings from the default section in the defined settings file (using the `set` parameter), and uses the suitable ones with the current table.
The default values are set after other possible keyword and table operations.

5. Variable handling

5.1 Variable handling with control files

The control file also makes it possible to reorder the variables and change the variable names, value texts and value codes.

The examples use the following settings:

```
in=... meta=Control.csv
```

The control file is shown with the semicolon separators.

```
STUB;HEADING
Region;
 ;Industry,Municipality
Year;"Province,2015",Age,Status
```

The variables are reordered with the base language names. The *Region* variable will be moved as the only row variable in the tables where it is found. The variables *Industry* and *Municipality* will be moved as the only column variables. In the tables which have the variables *Year*, *Province,2015*, *Age* and *Status* (and only them), the variable *Year* will be moved as a row variable the others as column variables in the given order. Variable names containing commas must be given in quotes.

```
variablename;valuetext;replacetext
Information; ;Series
Country;Finland;Finnland
```

The variable name *Series* will be named to *Information*, and the value text *Finnland* in the variable *Country* will be changed to *Finland*. The value text may also be changed with value code and the `code` column.

5.2 Partial tables

The tables may be cut into smaller parts using the job type `split` and the corresponding `csv` control file given with the `meta-` parameter. For example this control file:

```
STUB;HEADING;takevalues;
Region; ; ;Helsinki;Espoo;Vantaa;Kauniainen
;Age; ;
;Year;-10;
```

defines the result table having the row variable *Region* and its value texts, the variables *Age* and *Year* will be set in columns. *Age* will have all its values and *Year* the last ten ones. These variables have to be found in the current table; if there are more variables, the others will be put at the end of the row variable list. The output format may be changed with the output option `-o` (`px`, `csv`, `xls` or `htm`).

5.3 Variable combining

It is possible to combine the variables in the tables with the option `-g`. The option modifier contains the variable names in the base language as a comma-separated list. As default, the new variable name will be the old names joined with a slash (`/`), and the variable values and codes will be combined as well. The new name can be given with the option `-v` (for a multilingual table the names have to be given as a comma-separated list in the language order), the name separator may be changed with the option `-p`.

```
-gProvince,Municipality -vRegions -p:
```

The variables *Province* and *Municipality* will be combined as a new variable *Regions* with a colon as the name separator.

```
-gSTUB
```

Combine all the row variables with default settings.

```
-gYear,Month -vTime,Aika,Tid
```

Combine the variables *Year* and *Month* as a new variable in a multilingual table. If the variable *Year* has an existing *TIMEVAL* setting, it will be converted to monthly format, if possible, with the standardised value texts and codes. The switch `!t` may be used for the same purpose .

6. Translations

The database can be translated to new languages by first giving this type of command:

```
job=translate in=D:\dbase\ out=D:\langs\ -s1
```

This creates a separate text file from each `px` file in the database `D:\dbase\` to the directory `D:\langs` (in one single directory without sub-directories) with the file extension `translate`. The file consist of sections that start with the section header lines in brackets. For example:

```
[LANGUAGE]
fi
[VARIABLES]
Vuosi
Kuukausi
Toimiala
Sarja
[VALUES("Kuukausi")]
Tammikuu
Helmikuu
...
```

```
[VALUES("Toimiala")]
45 Moottoriajoneuvojen kauppa ja korjaus
46 Tukkukauppa
...
[SUBJECT-AREA]
Palkat ja työvoimakustannukset
[DESCRIPTION]
Palkkasummakuvaajat toimialoittain 2010=100 (TOL 2008)
...
```

The first section header is `[LANGUAGE]`, which determines the input language for each of the sections after it (before the next language section). When adding new languages, you may either just replace the old language code with a new one, or copy the whole language section at the end of the file for the new language code.

The section headers having variable or value texts may be either in the new or base language. Other section headers than the language headers need not be edited, though.

If the keyword contains a list (e.g. *STUB* or *VALUES*), the list items are given in separate rows. The corresponding lists must contain the same number of items in each language, and the order of the items should be the same.

The Swedish translation of the example text could then look like this:

```
[LANGUAGE]
sv
[VARIABLES]
År
Månad
Industri
Serie
[VALUES("Månad")]
Januari
Februari
...
[VALUES("Industri")]
45 Handel; reparation av motorfordon och motorcyklar
46 Parti- och provisionshandel utom med motorfordon
...
[SUBJECT-AREA]
Löner och arbetskraftskostnader
[DESCRIPTION]
Lönesummaindex efter näringsgren 2010=100 (TOL 2008)
...
```

The files are then ready for the translators for editing with any suitable text editor. When the translations are ready, they may be transferred to the files with the command:

```
job=translate in=D:\dbase\ meta=D:\langs\ -s
```

The translation files may be created for each language separately with the `-o1` option. Each language section will be written in a separate file, and the file name will contain the language code at the end of the file name separated by an underscore. When the translations are transferred back, PxJob will look for all possible language files in the source directory.

The language option `-l` may be used for creating language templates. If the source file doesn't have the language, the template translate file will be written using the main table language.

7. Table joining

Table joining is controlled with the option `-j`. The first file in the file list is considered as the base table to which the other files will be joined. The tables must have the same number of variables, and preferably in the same order. The base table metadata will be treated as the original metadata, and only the new metadata will be added to the joined table. The `-j` option modifiers may be used for fine-tuning the operation.

The variable order will be set according to the variable names. In the well standardised environment there are no problems, but in some cases the correspondent variables have to be deduced. For example, if the base table has the variables *Region*, *Year* and *Age* and the join table has the variables *year*, *age_structure* and *municipality*, the variable ordering will be changed to *municipality*, *year* and *age_structure*. At first the clear matches are filtered out (*Year* and *year*), then the first parts of the names (*Age* and *age_stucture*), and the others will just be left in the order (by pure coincidence, *Region* and *municipality* were the proper matches in this case).

The value texts are matched according to the value codes. If there are value texts in the variables of the join table that are not found in the base table, they will be moved to the end of the variable. The variables, that have hierarchical or otherwise suitable alphabetical or numeric codes (e.g. NACE), may be merged according to the code with the option `-jb`. Merging will be done after the join operation.

```
in=D:\dbase\Timeseries.px,D:\in\Newmonth.csv out=D:\out\ -j
```

The new monthly data will be joined in the *Timeseries* table. If needed, PxJob tries to change the corresponding *TIMEVAL* setting according to the new values.

```
in=D:\dbase\Industry.px,D:\in\NACE2010.xls out=D:\out\ -jb
```

The new annual data will be joined to the *Industry* table. The possible new industry values will be merged after the joining. The *TIMEVAL* setting will be changed, if it exists.

```
in=D:\in\provinces.xls -j !a out=D:\out\
```

All the *Excel* worksheets with stuctural tables will be opened (`!a`), they are joined (`-j`) and saved to the output directory as `provinces.px`.

```
in=D:\in\files\ -n"Causes of death" -j out=D:\out\Deaths.px
```

The files in the same directory will be joined as one table (*`Deaths`*`.px`). The file names are the classifications (variable values), and the result table will have a new variable (*Causes of death*), which will have those classifications as value texts.

7.1 Creating multilingual tables by joining monolingual ones

When joining tables with different languages, the variables must be in the same order (variable or value matching is not meaningful here). PxJob will always create a multilingual table, if possible. If some languages do not have the specific metadata, it will be copied from the base language.

Joining of the two monolingual tables as one multilingual one:

```
in=D:\dbase\Pop_fi.px,D:\dbase\Pop_en.px out=D:\out\Pop.px -j
```

If there is a database which has several languages, and each language is in a different file, and the file names have been standardised, the new multilingual database can be created with a single command.

For example, if the table names end with underscore and the language code (`Table_fi.px`, `Table_sv.px`), the tables in base language may even be without the language code (`Table.px`):

```
in=D:\dbase\ out=D:\multi\ -jn -s -len,fi,sv !w
```

This command creates a new database (`D:\multi\`) with the same structure as the original one (`-s`). All the other than `px` files will be copied (`!w`). The database tables will be joined according to the file names (`-jn`) in the language order English (base language), Finnish and Swedish (`-len,fi,sv`). All the language files are not needed in the database. The new tables will be copied according to the table base language, i.e. the files may be scattered in the database (arranged by language).

If the language identifier is at the end of the file name with a fixed length, (`TableE.px`, `TableSV.px` etc.), the numeric modifier of the option `-j` may show the identifier length (`-j1`..`-j4`), and the file names will be compared without the identifier.

8. Return codes

PxJob will return an exit code when terminating. The exit code may be used in the calling environment macros (such as `%errorlevel%` values within bat files). The following return codes are in use:

`0`	ok
`1-10` *interpreter errors*, such as:	
`1`	failed to start
`3`	system error
`4`	runtime violation
`11-20` *PxJob errors*:	
`11`	input file read error
`12`	output file write error
`13`	parsing problem
`14`	no input files
`15`	inject csv read error
`16`	template file not found
`17`	template file read error
`18`	zip engine missing
`19`	list file is empty
`21-255` *coding errors*, such as:	
`21`	ws full (memory problem)
`22`	syntax error
`23`	indexing error

NB: the error codes `11` or `12` will be returned, if the problem is encountered with any files when handling multiple files. The returned error code will be the greatest, if more than one errors is found.