PxJob 4.0

Introduction

General

- **PxJob**: command line based px-file managing application
  - meant especially for routine process automation
  - the commands may be given directly from the command line or via specific command files (e.g. `bat` or `cmd` text files)
  - may be called from other applications (the return code for a successful operation is `0`)
- PxJob uses the same source code as PxEdit
  - the default answer to all user interface requests (e.g. when opening or saving files) is **OK** (or **Yes**)
- PxJob has some functionalities that are not available in the PxEdit interface (and vice versa)

Installation

- Three files are needed for basic use:
  - `PxJob.exe` — main program (32 bit)
  - `PxEdit_40.paq` — program code (32 bit)
  - `Dyalog170rt_unicode.dll` — interpreter dll (32 bit)
- Other installation files, which may be needed are
  - `Zip.exe, Unzip.exe` — archive packer and opener
  - `PxEdit_main_40.ini` — main settings file
  - language files and other settings or control files
- PxJob may be installed simply by copying the necessary files into one directory
  - updating is usually simple: just replace the `paq` file with a newer one
  - Excel functionality needs Excel (or MS Office) to be installed

Statistics Finland

Batch

Command line syntax

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

job	job type (default: px)
in	source directory, file, file list or list file (mandatory)
out	output directory or file (default: source files)
err	directory for erroneous files
copy	directory for source files
meta	metadata file or directory
set	settings file
path	common directory path
log	log file
-	options
!	switches

Command line

- The command line consists of the program name `[path\]PxJob[.exe]` and space separated parameter definitions (arguments)
  - the parameter ordering is free
  - the parameter codes are separated from the definition part by an equal sign (for example `job=csv`)
  - if the first parameter definition does not have the equal sign, it is interpreted as a `job` parameter
  - `in` parameter is always compulsory
- The command line may also consist of the program name and a macro file name
  - the macro file contains one or more command lines
  - the macro file extension is `.pxjob`

`job`: batch type

`px`	standardised px-file (default)
`csv`	semicolon separated structural text file
`exp`	*eXplorer* file
`htm`	html file or table
`sql`	PxSQL macro
`txt`	tab separated structural text file
`xml`	CoSSI/XML table
`xls`	Excel file
`report`	database report
`split`	partial table
`translate`	multilingualiser

- If `job` parameter is missing, and the first argument does not have the equal sign, it is interpreted as the batch type
- If `job` cannot be deducted, it is interpreted as `job=px`

File types

- For some file types the default output (type, format and character coding) may be changed with switches (e.g. `-o` and `-t`)

- The default types need no extra switches
  - `px` standard `px` file
  - `csv` semicolon separated text file
  - `htm` html table with no specific colouring
  - `report` semicolon separated `csv` file
  - `split` `px` table
  - `sql` all *INPUT* macros
  - `txt` WinANSI coded, tab separated text file
  - `xls` Excel file (`xls` normally, `xlsx` for big tables)
  - `xml` `CoSSI/XML` table: *XDF* format

Parameters

`in:` source (mandatory) /1

- Either source directory, file, list file or comma-separated file list
  - if there are spaces or commas in the definition,
    the parameter must be given in quotes (`"`)
- Source directory
  - the directory name should end with a slash (`\` or `/`)
  - as default, only `px` files are read from the directory
    (see options `-i` and `-s`)
- Source file
  - the file name may include wildcard characters `?` and `*`
  - if the file extension is `list` (or `lst`), the file is regarded as a *list file*
  - direct net addresses start with `\\`
  - web addresses start with `http:`

`in:` source (mandatory) /2

- List file
  - a *list file* is a simple text file, which contains file names, one per row
  - lines that start with a semicolon are skipped (i.e. the file may include comments)
  - if all the files are in the same directory, only the first name must contain the file path
  - if all the files have a common path, it will be used with `s1` option
- File list
  - the file list consists of comma-separated file names (no spaces allowed)
  - if the file names contain commas, the names must be given in quotes
  - if all the files are in the same directory, only the first name must contain the file path

`in:` source (mandatory) /3

`in=D:\path\source\`	source directory
`in=D:/path/source/`	ditto
`in=D:\path\Source.xls`	source file
`in=D:\path\*.xls`	read all `xls` files
`in=D:\path\Allfiles.list`	read all list file files
`in="D:\pa th\Source.csv"`	space in the name
`in=D:\path1\*.px,D:\path2\*.csv`	file list (with wildcards)
`in=D:\path\A.px,B.px,C.px`	file list (in the same directory)
`in=D:\path\"A,1.px","B,2.px"`	file list (commas in names)
`in=\\ad.sta.fi\path\File.px`	direct net address
`in=http:\\px.sta.fi\path\File.px`	web address

`out:` result (recommended)

- Output directory or file
  - the directory name should end with a backslash
  - any non-existent directory will be created (see options `-o` and `-s`)
  - if there are spaces included, the parameter must be given in quotes

out=D:\path\Result.px	output file
out=D:\path\target\	output directory
out=D:/path/target/	slash is also permitted

- If `out` is omitted, the output files are written in the source directories possibly overwriting existing source files

- The output file name may be created also 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:\path\:matrix
```

`err:` error directory

- When this setting is in use, the erroneous source files will be copied in the designated directory (they will not be processed otherwise)
  - the directory name should end with a backslash
  - any non-existent directory will be created
  - if there are spaces in the definition, the parameter must be given in quotes
  - as default, PxJob is pretty sensitive to all errors with `err` parameter
    - the sensitivity level may be set with option `-e`
  - source files are not deleted automatically
    - the expunge switch `!x` is used for file deleting

`copy`: archive directory

- The source files will be copied into the designated directory
  - the directory name should end with a backslash
  - any non-existent directory will be created
  - if there are spaces in the definition, the parameter must be given in quotes
  - if the `err` parameter is in use, it will be used in erroneous cases
  - the source files are not deleted automatically
    - the expunge switch `!x` is used for file deleting

`meta:` templates and control files

- Setting of template directory or file
  - if there are spaces in the definition, the parameter must be given in quotes

`meta=D:\path\Result.pxk` template file

`meta=D:\path\templates\` template directory

- PxJob will try to find an equivalent metadata source file (`px` or `pxk`) which has the same name as the result file
  - with or without trailing or leading parts (separated by underscores)
  - name searching is not case sensitive
- The `meta` parameter is also used with multilingualising (`translate`), reporting (`report`), partial table splitting (`split`) and metadata injection (`-a`)

`set:` settings file

- Settings file for batch job
  - usually PxJob uses the main settings file in the installation directory (`PxEdit_main_40.ini`), if it is found
    - the personal ini file is not read
    - the main ini file may be skipped using the `!s` switch
  - the `set` parameter may be used for using an additional settings file (like the personal settings file in PxEdit)
    - it might be suitable e.g. for defining default keywords
  - if there are spaces in the definition, the parameter must be given in quotes

`path:` common directory path

- The common path setting for file/directory definitions in the command line
  - the file/directory settings that start with a backslash (\) will be prefixed with the `path` setting
  - if there are spaces in the definition, the parameter must be given in quotes
  - for example, the following input strings will be interpreted similarly:

```
in=D:\database\input\ out=D:\database\output\
 meta=D:\database\template\

in=\input\ out=\output\ meta=\template\ path=D:\database\

in=/input/ out=/output/ meta=/template/ path=D:/database/
```

`log`: keeping track

- The log file records the messages shown during the job (in English)
  - all error messages, confirmations, input and output file names, etc. will be recorded
  - if there are spaces in the definition, the parameter must be given in quotes
  - PxJob will also log the batch start and stop timestamp in the PxEdit log (if it is in use)
  - the log feature may be switched off with `!o`
- The log directory must exist (i.e. it will not be created)
  - the default directory is the launch directory
  - the default name is `pxjob_`*`yyyymmdd`*`.log`
- The file extension is `log`
  - if the file exists, the new log messages will be added to the file

Macro files

Command line may be replaced by a macro

- Macro files are normal text files
  - the extension is `.pxjob`
- Macros may contain several PxJob command lines, section headers, empty lines and comments
- Sections are enclosed with braces [ ]
  - section headers will be written in the log file
- Comments start with a semicolon ;
- Benefits
  - PxJob is loaded only once in memory
  - DOS character conversions are skipped

Options and switches

Options start with dash, switches with exclamation mark

- Options and switches are used to fine-tune the batch
  - the options have (usually) multiple alternatives
    - the symbols after the options in this document:
      `?/*/&` = one character/ string/ comma-separated list
  - the switches have two alternatives (states)

- The options and switches may be grouped, but

  `-g`, `-i`, `-j`, `-l`, `-n` and `-v` have to be separate

  for example

  ```
  -icsv,txt -o1 -u+2 -r -z -s !x
  ```

  may be shortened to

  ```
  -so1ru+2z -icsv,txt !x
  ```


Statistics Finland

Input

`-e?:` error sensitivity

- As default, PxJob handles all input files
  - except those that have fatal errors in them
    (e.g. the number of data cells does not equal the metadata)

- With the `err` parameter, PxJob will not tolerate invalid tables, and all warnings or errors will make the job copy the files in the `err` directory

  `-e1` normal sensitivity level (as without `err` parameter)

`-i&:` input file types

- Source file extension definition
  - the default input files are `px` files, but it is possible to give a comma-separated extension list (do not add spaces)

- Wildcard characters `?` and `*` are allowed

`-ixls`	only `xls` files will be read
`-ipx,xls`	both `px` and `xls` files
`-i*`	all files in the source directory

`-s?`: sub-directories

- As default, only the files in the source directory are handled
- The `-s` option is meant for situations, when there is need to go through the directory structure (starting from the source directory)
  - `-s1` reflect the directory structure in output, too
  - `-s2` write the output files in the output directory only (no sub-directories)

`-y*`: freshness filtering

- `y` filters input files based on age in days (“youngness”)
- The definition format is `dd.hh.min`

`-y14`	files no older than two weeks are read
`-y0.2`	files no older than two hours are read
`-y0.0.15`	files no older than fifteen minutes are read

- The *extended* format makes it possible to set the time window: `dd.hh.min+dd.hh.min`

`-y2+10`	not newer than ten minutes and older than two days
`-y2+1.0`	not newer than one hour and older than two days
`-y2+1.0.0`	not newer than one day and older than two days
`-y0+10`	not newer than ten minutes

Input switches

!a try to open all Excel worksheets

- usable e.g. for joining tables
- all sheets will have the sheet number at the end of the output file name (except the first sheet)

!b bypass the default input string conversion (DOS > Unicode)

- useful when there are national characters in the command, e.g. with the `-n` option

!d remove (delete) variables with only one value

!i show job progress information in the *Task Bar* balloon

!s skip the main settings file

!z convert BIG5 coded texts to Unicode

Statistics Finland

Metadata

`-a`: add and manage metadata

- Metadata for almost all the keywords (table, variable value or cell specific) can be added, updated or removed with a control csv file (given with the `meta` parameter) and `-a` option
  - the keywords *CODES*, *DATA*, *HIERARCHIES*, *HIERARCHYNAMES*, *KEYS*, *LANGUAGES*, *PARTITIONED* and *VALUES* are not manageable, though
  - the keyword contents are lightly checked (not as thoroughly as in px file reading)
  - the control texts are not case-sensitive
  - *TIMEVAL* value may be given as the frequency code (e.g. `M`)
  - `-a` option is not needed with the `job=px` setting
- It is also possible to pivot the table (change the variable order), and change any variable, value text or value code

`-a:` the control file

- The first row of the file contains the column headers

`<keyword>`	valid px keyword name (*STUB* and *HEADING* are special)
`languagecode`	specific language code
`variablename`	variable name for variable and value-specific keywords
`valuetext`	value text (or code) for value-specific keywords
`code`	value code for value-specific keywords
`<variable>`	variable names (for cell-specific keywords)
`replacetext`	the text string to be replaced (if needed)
`filename`	the file name or mask (if needed)

- The column order is free
- 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
- Empty cells are not handled

`-a:` keyword columns

- *<keyword>*
  - contains the new contents for the keyword
    or the text string, which replaces the `replacetext` string
  - tilde (**~**) removes the keyword (not a mandatory one, though)
  - a new keyword will be added, but replacing an existing one needs to have the `-r` option as well
  - *TIMEVAL* is treated differently: only the format of the new contents or just the frequency code is taken in account, the final keyword will be based on the target variable
  - footnote keywords can be copied to another same-level footnote (e.g. *NOTEX* to *NOTE*)
  - *VALUENOTE*s may be expanded to *CELLNOTE*s
  - *VALUES* may be copied to *CODES*

`-a:` language and variable columns

- `languagecode`
  - the language code for the row (empty ⇒ all table languages)
  - if the code equals the main language of the table, the keyword will also be set to all other table languages (if they are not set separately)

- `variablename`
  - mandatory for variable and value-specific keywords
  - the variable name may be either in the defined language (defined by `languagecode`) or in the main language

`-a`: variable, value and code columns

- *<variable>*
  - mandatory for cell-specific keywords
  - either value text in the defined or main language, value code, or `*` (= all)

- `valuetext`
  - `valuetext` or `code` is mandatory for value-specific keywords
  - either value text in the defined language or in the main language **or** value code (preferably `code` should be used for this)

- `code`
  - value code for value-specific keywords

`-a:` replace and file filtering columns

- `replacetext`
  - the text string, which is to be replaced
    (i.e. the keyword cell always contains the new contents)
  - may contain wildcard characters `*` and `?`
  - search is case-insensitive
  - will bypass the possible `-r` option
- `filename`
  - table file name
  - the `px` extension is not needed
  - wildcard characters `*` and `?` are accepted
  - may contain (part of the) database path

`-a`: changing variable order

- `STUB`
  - comma-separated variable list
    - if the variable name contains commas, it has to be enclosed in quotes
  - if there is no *HEADING* column and all the given variables exist in the table, they will be set as the row variables, and the others as the column variables
  - if there is a *HEADING* column, and the table has the same variables, the table will be pivoted according to the *STUB*, *HEADING* setting
- `HEADING`
  - works like *STUB* but for column variables

`-a:` possible column combinations /1

- `<keyword> {<keyword>, …} {replacetext} {languagecode} {filename}`
  - table-specific keyword injection
- `<keyword> {<keyword>, …} {variablename} {replacetext} {languagecode} {filename}`
  - variable-specific keyword injection
- `<keyword> {<keyword>, …} {variablename} {valuetext} {replacetext} {languagecode} {filename}`
- `<keyword> {<keyword>, …} {variablename} {code} {replacetext} {languagecode} {filename}`
  - value-specific keyword injection
- `<keyword> {<keyword>, …} <variable> {<variable>, …} {replacetext} {languagecode} {filename}`
  - cell-specific keyword injection

`-a:` possible column combinations /2

- `STUB {languagecode} {filename}`
- `HEADING {languagecode} {filename}`
  - setting row or column variables
- `STUB HEADING {languagecode} {filename}`
  - table pivoting
- `variablename replacetext {languagecode} {filename}`
  - change variable (from `replacetext`)
- `variablename valuename replacetext {languagecode} {filename}`
- `variablename valuename code {languagecode} {filename}`
  - change value text from `replacetext` or according to `code`
- `variablename code replacetext {languagecode} {filename}`
- `variablename code valuename {languagecode} {filename}`
  - change code from `replacetext` or according to `valuename`

-g&: group variables for combining

- The grouped variables will be joined as a new variable
- The variable name may be set with the -v option or it will be combined from the grouped variable
- The variable values and codes will be combined with the separator, which may be set with the -p option (the default is slash /)
  - the variable names are given in the main table language as a comma-separated list (the names can thus not contain commas)
  - the *TIMEVAL* keyword will be fixed, if possible (see !t switch)

```
-gYear,Month                          variables to be combined
-g"First name","Old name"   quotes needed for the spaces
-gHEADING                             combine all column variables
-gSTUB                                combine all row variables
```

`-h?:` Statistics Finland specific

- Most of these are reserved for Statistic Finland's internal use, relying upon the database structure, table file names and some specific metadata settings

  `-h6` key figures setting (variable order: *Alue*, *Tiedot*, *Vuosi*)

  `-h7` variable-value listings

  `-h8` *eXist* update csv

  `-h11` *StatFin* quality report

  `-h12` *StatFin* pipeline (`!h`): file name checking also for PxPro tables

  `-h25` search interesting data values in the database (default value is `25`, may be changed with `-v` option), the result is a `csv` file

`-m?` : default metadata

- The default keywords are read from the `[Default]` section in the main settings file
  - default metadata will be added after any other metadata operations

`-m1` copy the missing metadata

`-m2` replace all possible metadata

-n&: new variable

- The syntax: `-nVariable;value`
  - the single value may be either a given value or keyword contents
  - if `value` is missing, the file name will be used instead
  - the path and extension are removed from the file name and underscores are converted to spaces
  - the names may be given for multilingual tables as a comma-separated list (the language order must be standardised)
  - the names cannot have commas or semicolons

`-nInformation`	new variable name `Information`
`-n"New variable"`	quotes needed for the spaces
`-nNew;CONTENTS`	the value text comes from the given keyword
`-nTieto,Information;arvo,value`	bilingual settings

`-p*`: partition character or string

- The separator used in variable combining (see the `-g` option)
- The default separator is a slash (`/`)

`-p-`	use dash
`-p::`	use two colons
`-p" = "`	quotes needed for the spaces

`-r?`: replace metadata

- Used only with the `meta` parameter
  - `-r0` add all possible metadata (default)
  - `-r1` add and replace all possible metadata
  - `-r2` add all possible metadata and replace variable names (if there are the same number of names)
  - `-r3` add and replace all possible metadata and replace variable names, values and codes (use with caution!)

`-u*`: update timestamp

- Add the *LAST-UPDATED* keyword

`-u0`	use the current date (i.e. when the batch is run)
`-u20130713`	set the date (use the default time)
`-u20130713_13:30`	set the full timestamp
`-u+2`	set the date two days later

  - if the date would be on a weekend, it will be changed to the next weekday
  - as default, the weekend is regarded as Saturday and Sunday (see `-w`)
  - the default time is usually defined in the main settings file

`-u+2_09:00`	set the date and time in future

- Valueless option (`-u`) equals to option `-u0`

`-v*` : variable names

- As default, the combined variable name will be made from the grouped variable names using the combine separator (see the `-g` and `-p` options)

`-v`Time	set the new variable name as *Time*
`-v`Time,Tid,Aika	set the new names for multilingual table (in the language order)
`-v`"New name"	quotes needed for the spaces

`-w*` : weekend skipping/first column width

- Defining the non-Western weekend (along with option `-u`)
  - the default weekend is regarded as Saturday and Sunday
  - needed, if there is need to change the weekend days or just bypass the default weekend skipping with the option `-u`
  - weekday numbering: `0` (Monday) … `6` (Sunday)

`-w45`	set weekend as Friday and Saturday
`-w7`	no weekend skipping

- For `html` output, the option may be used for defining the first column width in pixels

`-w200`	200 pixel wide column

`-x?` : title fine-tuning

- For `px` job, the *TITLE* will always be set according to the px rules
- For other output types, the *TITLE* may be created from *CONTENTS* or *DESCRIPTION* without the variable names

`-x1`	set *TITLE* as *CONTENTS*
`-x2`	set *TITLE* as *DESCRIPTION* (or *CONTENTS*, if *DESCRIPTION* is not found)

Database report

`job=report:` database reports /1

- The database report result is a csv file

- The following column headers are used:

`<keyword>`	contents of the specified keyword
`filecreate`	file creation timestamp
`filename`	file name
`filepath`	the directory path of the file
`filesize`	file size in bytes
`fileupdate`	file change timestamp
`languagecode`	the language codes in the table
`pathname`	file path and name combined
`tablesize`	table size `(rows)x(columns)=figures`

`job=report:` database reports /2

- The following headers cause the report process to read the data part, which may slow down the process remarkably

`datacells`	total number of data cells
`datanumbers`	total number of figures
`datazeroes`	total number of zeroes
`datadashes`	total number of dash codes
`datadotcodes`	total number of dot codes
`datadots1..7`	total number of corresponding dot codes
`datamin`	the minimum data value
`datamax`	the maximum data value
`datamean`	the average data value

- If the header has the percent sign at the end (e.g. `datadotcodes%`), its value will be the percentage calculated with the total number of cells

`report:` general

- If there is no control csv file, the report includes the columns `filepath`, `filename`, `filesize`, `fileupdate`, `tablesize`, `languagecode`, `VARIABLES` and all mandatory keywords

- The `VARIABLES` column contains all the variables in the table (from *STUB* and *HEADING* keywords)
- The keywords *CODES*, *HEADING*, *KEYS*, *STUB* and *VALUES* cannot be included in the report

- The `!f` switch will print all the requested file information in the report even though there is no filtering information found (see the control csv structure)
- The controls are not case sensitive

`report:` the control file /1

- The report contents may be tuned by a control csv file, which is given with `meta` parameter; the file may be created in PxEdit (*Edit*|*Database*|*Report*)
- The **first** column is a control code, which is either `0` (general) or `1..4` (the keyword level: table, variable, value or cell specific keyword accordingly)
- The **second** column contains the control words or keywords
- The general controls are either for information (`filename`, `filecreate`, …) or filtering (`languagecode`, `variable`, `value` or `content`)
- `languagecode` may be alone (all the table languages) or it may be used for filtering the needed languages (language codes in separate columns)
  - each language will add a new line in the report

`0;languagecode` all languages

`0;languagecode;en;sv` only English and Swedish metadata

`report:` the control file /2

- `variable` may be alone (equals to all table variables) or it may be used for filtering the needed variables (variable names in separate columns)
  - each variable will add a new line in the report

- The variable names can be either in specified or main language

`0;variable`	all variables (each in different lines)
`0;variable;alue;region`	filtering of the listed variables
`1;VARIABLES`	all variable names in a single cell

- `value` defines the filtering value texts or codes
  - the value names can be either in specified or main language

```
0;value;helsinki;091
```

`report:` the control file /3

- `content` defines the keyword content filtering (it does not need to be the whole text) for all keywords

```
0;content;Tilastokesk;Statist
```

- It is possible to define the filtering values for each keyword separately

```
1;DESCRIPTION;kala;fisk;fish
```

- The report creating will be permissive, i.e. if an individual keyword is filtered (and the corresponding cell left empty), other keyword values may still add a new line in the report

Table joining

`-j&:` table joining

- The joining option has to be separate
- When this option is in use, all the input files will be opened and grouped, and each group is joined to the first table (the only criteria: the same number of variables)
  - corresponds to *Edit*|*Join* function with multiple tables in PxEdit
- The plain `-j` option joins the tables with default values
- The `-jn` option tries to join single language tables to multilingual ones
  - the file names should end with underscore and the language code, such as `Example_en.px` and `Example_fi.px`, but the main language file may lack the postfix
  - with `-j1..4` options the file names may have 1 to 4 denoting characters at the end, like `ExampleEN.px` and `ExampleFI.px` for `-j2`

`-j&:` join options

- These joining options are closely related to the PxEdit *Join* window

`-ja`	replace all metadata (default: only missing ones)
`-jb`	merge new values with old values **after** joining
`-jc`	do not use codes when matching values
`-je`	exact value text or code match with values
`-jf`	bypass fill items
`-jl`	do not create multilingual files, if possible
`-jm`	do not try to match the variable names
`-jn`	group the tables by file name without the last name part
`-jo`	use only original values
`-jr`	do not replace any metadata
`-js`	do not try to match values
`-jt`	replace value texts
`-ju`	combine the unique *SOURCE* keywords as a `#`-separated list
`-j1..4`	group the file names without the last 1 to 4 characters

- There may be several settings in use at the same time (`-jmo`)

Table splitting

`split:` the control file /1

- The splitting functionality needs a controlling csv file, which is given with `meta` parameter

- The first row contains the column headers

`STUB`	row variable name (mandatory)
`HEADING`	column variable name (mandatory)
`languagecode`	specific language code (default: base language)
`takevalues`	number of values taken from the variable list (if negative, from the end)
`skipvalue`	value texts or codes which will not be in the result
`withvalue`	value text (if not given, all the values are selected)

`split:` the control file /2

- The variable and value names are case insensitive without ending blanks
- Variable order will be the same as in the control file, extra variables will be put in row variables
- The value texts may contain value codes as well
- Empty column code is interpreted as `withvalue`

File saving

-b* : bypass naming standards

- Fine tuning of file names
  - **-b_** spaces will not be converted to underscores
  - **-b=** uppercase characters will not be converted to lowercase
  - **-b~** accented characters will not be converted to ascii
- Multiple settings may be defined, if needed (**-b**~_=)
- For reporting
  - **-b/** sets the file path separator as slash

-c?: character conversion

- Set the character conversion for text and `px` files
  - `-c0` WinANSI (default)
  - `-c1` Unicode (*UTF-8*)
  - `-c2` ISO-8859
  - `-c10` WinANSI, missing *CHARSET* is interpreted as DOS coding
  - `-c11` Unicode (- “ -)
  - `-c12` ISO-8859 (- “ -)
  - `-c20` WinANSI, the *CODEPAGE* setting is ignored
  - `-c21` Unicode (- “ -)
  - `-c22` ISO-8859 (- “ -)
- It is still possible to read the old DOS-ANSI formatted px files, but DOS saving format is not supported
- Valueless option (`-c`) equals to option `-c1`

`-d*` : dash conversion

- The dash characters (**not** for `px` files) may be converted to dot codes or zero
  - `-d0` change dashes to zeroes

`-f*`: fill item

- The fill item may be any dot code or zero
- The default value will be read from the main settings file (default: `..`)
  - `-f...` set fill item as three dots

-k? : create new codes

- Empty code lists will be replaced
  - -k1 use the corresponding value texts of the main language
    - if there are unique space separated prefixes in all the value texts, they will be used
  - -k2 use only the corresponding value texts
  - -k3 create sequential zero-padded numeric codes
    - if all the corresponding values are numeric, they will be used
  - -k4 create only sequential zero-padded numeric codes
- Code lists containing empty codes will be patched with new codes using options -k11 to -k14
- Code lists containing empty codes will be replaced with new codes using options -k21 to -k24

-l&: languages /1

- The language option is used either for defining the languages for the output tables or the languages to be added when using the `meta` parameter
  - the first language in the list becomes the main language, which will e.g. define the possible character conversions
  - the languages will be added, if there are suitable metadata in the source
  - the option makes PxJob read all the available language files

`-len`	use only English in the output file (if found), if there is no *LANGUAGE* keyword, it will be set as *LANGUAGE*="en";
`-lfi,sv,en`	the output tables will have Finnish (main language), Swedish and English (if possible)

`-l&:` languages /2

- Valueless option (`-l`) makes PxJob read the language files
  - the txt strings needed for *TITLE* creation are included in PxJob for languages `da`, `en`, `es`, `fi`, `fr`, `it`, `kl`, `no` , `pt`, `ru`, `sl`, `sv` and `uk`
- The underscore character in the language list (e.g. `-l_fi,en`) makes PxJob read the language codes from the file names where the code is the last part of the name and separated by the underscore character (e.g. `table_en.xlsx`)
- The plus character in the language list (e.g. `-l+`) makes PxJob save the language code in the next cell after the header for the structural files (`csv`, `txt` and `xls`)

-o?: output type /1

- px
  - -o1 metadata part of the table (pxk)
  - -o2 sparse matrix format
- csv
  - -o1 tab as field separator
  - -o2 comma as field separator
  - -o3 create metadata-csv (verbose)
  - -o4 create metadata-csv (all metadata)
  - -o5 tab as field separator, file extension is xls (see !q)
  - -o6 all variables in rows (semicolon separated)
- htm
  - -o1 coloured cell backgrounds
  - -o2 no cell colouring
  - -o3 html table

-o?: output type /2

- `report`
  - `-o1` tab as field separator
  - `-o2` comma as field separator
- `translate`
  - `-o1` separate files
    language code at the end of each file name
- `split`
  - `-o1` `csv`
  - `-o2` `xls`
  - `-o3` `htm`
  - `-o4` `htm`, coloured cell backgrounds
  - `-o5` `htm`, no cell colouring

`-o?:` output type /3

- `sql`
  - `-o1` only metadata INSERT macros
  - `-o2` only data INSERT macros
  - `-o3` bulk saving (data part in `csv` format)
  - `-o4` DROP, CREATE and all INSERT macros
  - `-o5` DROP, CREATE and metadata INSERT macros
- `xls`
  - `-o1` tables in `xlsx` format
- `xml`
  - `-o1` tables in `XML/Cals` format
  - `-o2` tables in `XML/Keys` format

-q* : decimal and thousand separators

- Formatting is used mainly in `XML/Cals` saving

`-q,`	comma for decimals, no thousand separator may be used in `txt` and `csv` outputs as well
`-q,.`	comma for decimals, dot for thousands
`-q.,`	dot for decimals, comma for thousands
`-q,_`	comma for decimals, space for thousands
`-q_`	dot for decimals, space for thousands
`-q,~`	comma for decimals, non-breaking space for thousands
`-q~`	dot for decimals, non-breaking space for thousands
`-q,'`	comma for decimals, apostrophe for thousands
`-q'`	dot for decimals, apostrophe for thousands

`-t*`: variable titles /1

- Effects only `csv`, `htm`, `txt` and `xls` output
  - hierarchically arranged:

    `-t0` texts
    `-t1` codes (`htm`, `txt`)
    `-t2` codes and texts combined (`htm`, `txt`)
    `-t3` codes and texts

  - all values:

    `-t10` texts
    `-t11` codes (`htm`, `txt`)
    `-t12` codes and texts combined (`htm`, `txt`)
    `-t13` codes and texts

`-t*`: variable titles /2

- Effects only `htm` output
  - hierarchically arranged, px-style title column:
    - `-t20` texts only
    - `-t21` codes only
    - `-t22` codes and texts combined
  - hierarchically arranged, px-style title column, last variable in columns:
    - `-t30` texts only
    - `-t31` codes only
    - `-t32` codes and texts combined

-z?: zip files to archives

- The separate `zip.exe` program is needed (included in the setup package)

- The archiving type depends on the `out` parameter
  - file: all files will be zipped in it
    - `-s` option copies the directory structure within the output file
  - directory: all output will be copied in individual archives
    - `-s` option copies the directory structure within the output directory

- If there is no `out` parameter, the files will be packed in the source directory
  - **`-z`**`2` option packs files in each directory in files (name = the directory)
  - **`-z`**`3` option works as `z2`, but the file names contain the directory paths (backslashes are replaced by underscores)

- Valueless option (`-z`) equals to option **`-z`**`1`

Saving switches /1

!c combine codes to value texts (`csv` and `xls`)

!f always print the file information in the `report` (bypass filtering)

- forces handling all the files in metadata injection
- changes dot codes to zeroes in text file output (`csv`, `htm` and `txt`)

!g add a single language code at the end of the file name

!h database publishing pipeline settings in Statistics Finland
add hierarchy codes to the structural table (`csv` and `xls`)

!k keep the file change date

!l use system language for character conversion

Saving switches /2

!m add metadata (the keyword block) to the structural table

!n add footnotes in the output table (`csv`, `htm`, `xls`)
prevent metadata copying from base language (`px`)

!o do not write the log file

!p save using the shown decimal precision (**not** for px files)

!q quick copying of data part from source file (in `px` job)
save the file with both `csv` and `xls` extensions (`csv` job & `-o5`)

!t try to set the *TIMEVAL* when combining variables

!u set the file timestamp from *LAST-UPDATED*

Saving switches /3

!v file replace validation

- only with `px` job and `out` directory parameter, no join or zipping defined
- checks that there is no newer output file present in the output directory

!w write other than `px` files to the output directory

- only with different `out` and `in` directories

!x delete (expunge) the source file(s) (use with caution!)

!y save changed tables only

Metadata editing

There are many ways to manage metadata

- Keyword fetching from template files
  - prepared px or pxk file is needed
  - the file does not have to be perfect, PxEdit has to be able to open it

```
PxJob in=… out=… meta=Template.pxk …
```

- Adding default keyword values from the settings file
  - prepared settings file is needed (with `[Defaults]` section)
  - useful in quick patches (for example setting *CONTACT* keyword)

```
PxJob in=… out=… set=Myset.ini –m …
```

- Metadata injection with control csv files (versatile)
  - separate csv files needed (creating is easy, e.g. with PxJob or Excel)

```
PxJob in=… out=… meta=Control.csv …
```

Simple multilingualising

`translate`: multilingualising

- Creating the translation files (no `meta` parameter):

  `PxJob translate in=D:\dbase\ out=D:\lang\ -s2`

  creates `translate` files from each px file in the database

- The translation files contain (most of the) multilingual keywords
  - every keyword defines its own block (in brackets)
  - the texts inside the sections should be translated, and the language code set accordingly
  - the file may contain several language blocks

- Multilingualising
  - the `meta` parameter defines the translation files/directory

  `PxJob translate in=D:\dbase\ meta=D:\lang\ out=D:\outp\ -s`

  multilingualises those px files in the database, for which the corresponding translation file is found

Examples

Database actions

```
PxJob in=d:\dbase\ -s log=d:\log\Check.log
```

- read all px files in the database d:\dbase\ (sub-directories, too), validate them and save back, write all actions in the log file

```
PxJob in=d:/dbase/ -sy7 log=d:/log/Check
```

- as before, but read only files, which are no older than one week

```
PxJob csv in=d:\dbase\ out=d:\ctab\ -sc1 log=d:\log\Csv
```

- convert all px files in the database d:\dbase as Unicode (UTF-8) csv tables (according to the main language) to the directory d:\ctab

```
PxJob csv in=d:\dbase\ out=d:\ctab\ -sc1 -lsv log=d:\log\Csv
```

- as before, but now the metadata will be in Swedish (if there is sv language setting in the table) or in the table main language

Metadata enriching

`PxJob in=d:\input\ meta=d:\template\ log=d:\log\Fetch`

- read px files from the directory `d:\input\`
- fetch possible new metadata for each file from the directory `d:\template` according to the table name
- save the result files in the source directory (no `out=` parameter)

`PxJob in=d:\input\ meta=d:\template\ -r log=d:\log\Fetch`

- as before, but now all possible keywords will be copied

`PxJob in=d:\input\ set=d:\Batch.ini -ms log=d:\log\Set`

- add to all px files in the database `d:\dbase` (with sub-directories) the new default keywords from the `[Defaults]`section in the settings file `d:\Batch.ini`

`PxJob in=d:\input\ set=d:\Batch.ini -m2s log=d:\log\Set`

- as before, but now the existing values will be replaced as well

Inject metadata: table-specific/1

`PxJob in=d:\dbase\ meta=d:\control\Ctrl_tbl1 -s …`

- read the control file and add new keywords for each language in the database `d:\dbase` (sub-directories included)

`PxJob in=d:\dbase\ meta=d:\control\Ctrl_tbl1 -sr …`

- as before, but now the keywords will be added or replaced

- The file structure is now as follows (`Ctrl_tbl1.csv`):

	A	B	C
1	**NOTE**	**languagecode**	
2	taulukkoalaviite	fi	
3	table note	en	
4	samma på svenska	sv	
5			
6			

Inject metadata: table-specific /2

`PxJob in=d:\dbase\ meta=d:\control\Ctrl_tbl2 -s …`

- the command is similar to the previous one

- The control file (`Ctrl_tbl2.csv`) contains several control rows:
  - set the *NOTE* keyword for files, the name of which starts with `0?0`
  - remove existing *PRESTEXT* keywords (headers are not case sensitive)
  - replace text *area* to *municipality* in the keywords *DESCRIPTION* and *CONTENTS*

	A	B	C	D	E	F	G
1	**NOTE**	**prestext**	**DESCRIPTION**	**CONTENTS**	**filename**	**replacetext**	
2	table note				0?0*		
3		~					
4			municipality	municipality		area	
5							
6							

Statistics Finland

Inject metadata: variable-specific

```
PxJob in=d:\dbase\ meta=d:\control\Ctrl_var -as …
```

- The control file (`Ctrl_var.csv`) :
  - set the *NOTE* keyword for variables of different languages
  - set the *TIMEVAL* keyword for the variable *vuosi* (`A` equals to `TLIST(A)`)
    - works for multilingual tables, if the main language is Finnish and the name of the variable is *vuosi*, *Vuosi*, *VUOSI*, etc.

	A	B	C	D	E
1	**NOTE**	**TIMEVAL**	**variablename**	**languagecode**	
2	muuttuja-alaviite		tienkäyttäjä	fi	
3	bära bilbälte		tienkäyttäjä	sv	
4	remember safety		road user	en	
5		A	vuosi		
6					

Inject metadata: value-specific

```
PxJob in=d:\dbase\ meta=d:\control\Ctrl_val -as …
```

- The control file (`Ctrl_val.csv`) :
  - set the *VALUENOTE* keyword for values
    - value *Suomi* in variable *Valtio* (all languages, Finnish as base language)
    - value *Finland* in variable *Country* (in English)
    - value code *529 (*for Naantali town*)* in variable *Region* (in Swedish)

	A	B	C	D	E
1	**VALUENOTE**	**variablename**	**valuetext**	**languagecode**	
2	100-vuotias	Valtio	Suomi		
3	100 years	Country	Finland	en	
4	Mumindalen	529	Region	sv	
5					

Inject metadata: cell-specific

```
PxJob in=d:\dbase\ meta=d:\control\Ctrl_cell -as …
```

- The control file (`Ctrl_cell.csv`):
  - set the *CELLNOTE* keyword
    - the table must have the variables *Industry*, *Indices*, *Month* and *Year* (or some subset of them)
    - the setting will touch all values (`*`) for the variable *Indices*, and the defined values for other variables (if found)

	A	B	C	D	E	F
1	**Industry**	**Indices**	**Month**	**Year**	**CELLNOTE**	
2	50-52	*	7	1998	Footnote for July 1998	
3						

Inject metadata: different levels

```
PxJob in=d:\dbase\ meta=d:\control\Ctrl_comb -as …
```

- The control file (`Ctrl_comb.csv`):
  - a small combination of the previous control files
  - PxJob tries to deduct the preferred injection based on non-empty cells of individual rows

	A	B	C	D	E	F	G
1	**NOTE**	**TIMEVAL**	**VALUENOTE**	**languagecode**	**variablename**	**valuetext**	
2	taulukkoalaviite			fi			
3	table note			en			
4		A			vuosi		
5			100 years	en	country	Finland	
6							

Inject metadata: replacing text

`PxJob in=d:\dbase\ meta=d:\control\Ctrl_repl -as …`

- The control file (`Ctrl_repl.csv`):
  - first line: replace text "`contry`" to "`Country`" in the *NOTE* keyword
  - second line: replace text "`comments#`" to "`Comments :`" for *CELLNOTE*s for year 2000 and France
  - third line: replace text "`comments#`" to "`Comments: `" for all *CELLNOTE*s
  - fourth line: expand the *VALUENOTE* for Finland to *CELLNOTE*s
  - fifth line: delete the *VALUENOTE* for Finland

	A	B	C	D	E	F	G
1	NOTE	replacetext	CELLNOTE	year	country	VALUENOTE	
2	Country	Contry					
3		Comments#	Comments :	2000	France		
4		Comments#	Comments:	*	*		
5			VALUENOTE	*	Finland		
6					Finland	~	
7							

Statistics Finland

Table pivoting

```
PxJob in=d:\dbase\ meta=d:\control\Ctrl_set -as …
```

- The control file (`Ctrl_set.csv`) :
  - pivot the table with variables *Year*, *Region* and *Age*
  - move the variable *Gender* to row variable (others to columns)
  - move the variables *Region* and *NACE, 2008* as column variables

	A	B	C
1	**STUB**	**HEADING**	
2	Year	Region, Age	
3	Gender		
4		Region, "NACE, 2008"	
5			

Table splitting

```
PxJob split in=d:\dbase\ meta=d:\control\Ctrl_spl -s …
```

- The control file (`Ctrl_spl.csv`) :
  - the variable *Region* will have three named values in this order in rows
  - the variable *Industry* will have all its values in rows
  - the variable *Year* will have the five last values in columns
  - the table must have at least one of these variables
  - other possible variables will be in rows

	A	B	C	D	E	F	G
1	**STUB**	**HEADING**	**takevalues**				
2	Region			Helsinki	Tampere	Turku	
3	Industry						
4		Year	-5				
5							

Table joining

```
PxJob in=d:\dbase\Tseries.px,"d:\in\Monthly upd.xlsx" -j ...
```

- add data for a new month to the time series from a structural Excel table
- the Excel file name has to be in quotes because of the space in the name
- the original table has to be the first in the list

```
PxJob in=d:\input\Provinces.xlsx out=d:\result\Package
      meta=d:\template\Mk.pxk !a -jz
```

- open all worksheets of the Excel file (they have to be structural): `!a`
- join the tables: `-j`
- add suitable metadata from the template file: `meta=`
- save the zipped (`-z`) result (`Provinces.px`) as `Package.zip`

Miscellaneous /1

```
PxJob report in=d:\dbase\ out=d:\reports\ -s
```

- make a default database report and save it in the output directory as `pxreport_timestamp.csv`

```
PxJob in=d:\dbase\deaths\ out=d:\result\Co_deaths.px
      "-nCause of death" -j
```

- add a new variable *Cause of death* in the tables (NB: quotes), the value texts will be taken from the file names: `-n`
- join the tables (with the same number of variables): `-j`
- save the result table as `Co_deaths.px`

```
PxJob in=d:\spain\ -les -s
```

- if needed, add the *LANGUAGE* keyword as Spanish, and create the *TITLE* keywords, too

Miscellaneous /2

```
PxJob in=d:\dbase\ -gYear,Period -vTime -s !t
```

- combine (group) variables *Year* and *Period* as a variable *Time* in each table in the database where such variables are found
- try to set a suitable *TIMEVAL* expression for the combined variable
- if it succeeds, the variable codes are set according to the new *TIMEVAL*

```
PxJob in=d:\olddb\ out=d:\newdb -jn -s -lfi,sv,en
```

- join monolingual tables in the database `olddb` as multilingual ones in the language order Finnish, Swedish and English to new location `newdb` with sub-directories
- the monolingual file names should have the language code as separate postfix (the main language files can lack the code)
  e.g. `Table1_fi.px`, `Table1_sv.px`, `Table1_en.px`, `Table2.px`, `Table2_sw.px`, `Table2_en.px`, ...