IVTT – Intermediate Voynich MS Transliteration Tool – User Manual

R.Zandbergen

Issue 1.1, 10/04/2020

Contents

1 Purpose 3

1.1 General 3

1.2 Limitations 3

1.3 Version 3

1.4 References 3

2 Introduction 4

3 Options 6

3.1 Text selection options 6

3.1.1 General 6

3.1.2 Page selection 7

3.1.3 Locus selection options 7

3.2 Processing options 10

3.2.1 General 10

3.2.2 Processing annotations – comments and foliation 10

3.2.3 Options related to characters 11

3.2.4 Options related to whitespace and line wrapping 12

3.2.5 Miscellaneous options 13

4 Processing non-standard files 15

5 Diagnostics messages 16

5.1 Warnings 16

5.2 Errors 16

6 Operating system 18

1 Purpose

1.1 General

This document describes a tool (’`ivtt`’) that can read and process files in the IVTFF format. The format description of such files is given in Reference [R1]. The user will need to consult this document ([R1]) in order to be able to properly use `ivtt`.

1.2 Limitations

No more bugs are known to me and the tool has been tested rather thoroughly, and used in a routine manner.

1.3 Version

This is version 1.1 of the document, dated 10/04/2020. It describes the use of the software version 1.1, which can process IVTFF files of format versions 1.5 to 1.7.

1.4 References

[R-1] R. Zandbergen: IVTFF – Intermediate Voynich MS Transliteration File Format. Issue 1.7 of 10/04/2020. Available via: http://www.voynich.nu/software/ivtt/IVTFF_format.pdf

[R-2] IVTFF – Conventions (to be issued)

[R-4] Web page: http://www.voynich.nu/transcr.html

[R-5] Web site: http://www.voynich.nu/

2 Introduction

The “Intermediate” Voynich Transliteration Tool (`ivtt`) may be used to read and modify Voynich Manuscript transliteration files in various ways. Its two main purposes are:

- Selecting a portion of the text (for example: all text in Currier language A, or all text on herbal folios)
- Turning the selected formatted text into 'raw data' for further processing by analysis tools.

It is invoked by the command:

```
ivtt
```

followed by various arguments. Arguments starting with a plus or minus sign are treated as options. Other arguments are treated as file names.

It reads a transliteration file (or other text file) from standard input or a user-named file, writes the output (processed) file to standard output (or again a user-named file) and it will write some information to standard error output.

For example, the command:

```
ivtt ZL_ivtff_1q.txt out.txt
```

will read file `ZL_ivtff_1q.txt`, write it out to `out.txt` without modification, and it will display the following messages:

```
Intermediate Voynich Transliteration Tool (v 1.1)

Summary of options:
Leave high ascii as is
Keep hash comment lines
Keep all inline comments
Keep foliation
Keep ligature brackets
Use comma for uncertain spaces
Use dot for normal spaces
Keep white space
Keep alternate readings notation
Keep words containing ?
Keep paragraph start / end code
Keep drawing intrusion code
Maintain line wrapping

Line selection options:
Ignore transliterator ID
Keep all normal paragraph text lines

Page/tag selection options (exclude overrules include):
- Include all.
```

```
Input file: ZL_ivtff_1q.txt
Output file: out.txt

Starting...

   8510 lines read in
      0 lines de-selected
      0 hash comment lines suppressed
      0 empty lines suppressed
   8510 lines written to output
```

possibly with some warnings or error messages about the input file between the '`Starting`' and '`nnnn lines read in`' output. Operating system-dependent features of the input and output are discussed at the end of this document.

If the input file has no IVTFF file header, a warning is issued, but the tool will continue.

While the file header looks like a comment line, it is treated in a different manner. For this point, see the descriptions of options `–c` and `–f` further below.

3 Options

The user may specify two types of options: those to decide whether to include or exclude different parts of the text, and those to indicate how to process the selected (i.e. included) text. Pages or loci that are not selected are not written to the output. In order to select all text, just omit any page or locus selection options.

IVTFF-formatted files include transliterated text of the Voynich MS with different types of annotations:

- Textual comments
- Indications about which page the transliterated text is on
- Indications about which location of the page the transliterated text is on
- Indications about uncertain readings of the text

In order to do text processing of the transliterated text, the user will usually want to strip the file of these annotations, and this is what the processing options are for.

All options should be placed on the command line. There may be any number of options, and they can be of any combination of selection and processing options. They are of the format:

- a plus or a minus sign; followed by:
- one character, which may be upper case alphabetical for page selection options, the @-sign for locus selection options, or lower case alphabetical for processing options; followed by:
- one alphanumerical character (with one exception, where also a pair of characters is allowed)

The plus sign is used only for the selection options (page selection, locus selection, transcriber[1] selection - see below).

There can be at most two file names on the command line. The first name, if present, will be used as the input file (instead of using standard input), and the second one, if present, as the output file (instead of using standard output).

Each argument must be separated from the others by at least one space. The arguments can be in any order, but they will be interpreted from left to right. Contradicting options are allowed; the rightmost one will take precedence. Side effects of some options (e.g. of `-q1`, `-s2`, `-h3`, `+@...`, `-x...`) may thus be overruled.

3.1 Text selection options

3.1.1 General

Text selection options can be of two types: selection / elimination of entire pages or selection / elimination of individual loci. They are all based on the presence of certain dedicated comments in the transliteration file.

[1] While the Voynich MS text has been transliterated rather than transcribed, there does not appear to be a word like 'transliterator' so transcriber will be used instead.

For page selection, these comments are included in the page header and are called page variables. Selection of individual loci can be based on three different criteria. One of these is the use of so-called ’text tags’ which are the same as page variables, but defined for individual lines/loci or groups of loci.

3.1.2 Page selection

Page selection is based on the presence of page variables on the first line for each of the pages (i.e. the page headers). These comments set a number of one-character variables to one-character values. For example the following line in a transliteration file:

```
<f12r>   <! $A=1  $B=b   $C=C >
```

sets variable A to 1, variable B to b and variable C to C, for folio 12 recto. All variable definitions set for a previous page are reset at the start of a new page. Variable names must be upper case characters, while their values can be upper case, lower case or numerical.

The command line options to select/de-select certain pages are given in Table 1.

Table 1: page selection options (generic)

Example	Meaning
+A1	Include page if it has variable A set to value 1
-Bb	Exclude page if it has variable B set to value b

Some variables are pre-defined according to the IVTFF format, for which see reference [R1].

As an example, the following command will select all herbal B pages in file trans.eva and write these to file hb.eva (without any further processing):

```
 ivtt +IH +LB  trans.eva  hb.eva
```

The cumulative effect of page selection options is as follows: a page will be included if it matches all of the include options and at the same time none of the exclude options. Thus, in the above example only those pages that are both herbal (by illustration type) and in Currier language B are selected.

If a variable is not defined for a certain page, it is the same as if its value was set to 'space'. This will not match any include option and it will not match any exclude option. Thus, if the Currier language variable is not defined for a given page, the effect of the selection option +LA is to *not include it* in the output file, but `-`LB will *include it* together with the language A-pages.

Furthermore, the user is free to add other variable settings to his own files and use `ivtt` to select pages on the basis of these.

3.1.3 Locus selection options

The locus selection option may be used to include/exclude loci based on three different criteria:

- By text tag
- By locus type
- By transcriber ID

Locus selection by text tag or locus type may have the side effect that all loci for one page are de-selected, and the output for this page is empty. In that case the page header will also not be output, as per [R-1]. This has as a second side effect that most comment lines are automatically suppressed when text tag or locus type selection is enabled.

3.1.3.1 Locus selection by text tag

The use of text tags is explained in detail in [R-1], and may be summarised as follows. The list of text tags (variables, values) is the same as for page variables. They are set/unset in the text with the dedicated comment:

```
<@A=1>
```

These comments will only be interpreted if the page header has a corresponding page variable setting:

```
<$A=@>
```

With this, the ivtt option “+A1” will select text that has either page variable or text tag A set to 1. The use of text tags is valid from IVTFF format 1.7 only, and supported by ivtt version 1.1 . It has no impact on the ivtt command, but only on how this command is interpreted. Versions of ivtt lower than 1.1 cannot process files that include text tags, and would stop with an error message if they are encountered.

3.1.3.2 Locus selection by locus type

For selection of locus types, the command line option to include or exclude them is to use the (pseudo-) page variable @. Locus types to be selected (with the +@ option) or de-selected (with the -@ option) may be specified either as single-character values or as character pairs. As explained in Table 9 of Ref. [R1], the single-character values are ‘generic types’. The single-characters values that are allowed are given in Table 2.

Table 2: Locus types (single character)

Value	**Meaning**
P	Normal running text
L	Labels and dislocated words or characters
C	Circular writing
R	Lines along radii of a circle

By specifying, for example, the option: +@L , the user selects all loci of generic type L, i.e. all possible label types.

By specifying, for example, the option: `+@Lz` , the user selects only loci which are zodiac labels.

For possible values of 2-character locus types, see the IVTFF format definition, [R1].

As another example, an output file including only all labels in the biological section may be obtained with the following command:

```
  ivtt +IB +@L  trans.eva  blabel.eva
```

3.1.3.3 *Locus selection by transcriber ID*

The transcriber selection option is only useful for interlinear files, i.e. files in which several versions of each transliteration may be present. The option looks like a processing option (for which see below), since it uses a lower-case t. However, it may be preceded both by a plus sign or a minus sign. It is used to select certain lines from transliterations files, but in a different way than for page or locus selection as described above.

The meaning of the transcriber selection options is as follows:

Table 3: transcriber selection options

Option	**Meaning**	**Comment**
`-tX`	Select only lines from transcriber 'X'. This also selects lines that have no transcriber ID. If this option is omitted, all lines are kept in the file, regardless of the transcriber ID	Whitespace that may have been used to align the various transliterations will be removed.
`+tX`	Similar to `-tX`. The difference is that in this case the transcriber identification part in the locus (`;X`) is preserved, while for `-tX` this is deleted	Whitespace that may have been used to align the various transliterations will be removed.

3.2 Processing options

3.2.1 General

Processing options are of the general format: minus sign followed by one (lower case) alphabetical character followed by one digit. For any processing option that is not specified there is a default choice. The default is always the same as when specifying a zero for the digit.

Just as an example, the following two commands will do the same.

```
ivtt -c3 -f0 -u2 -w0 -c1

ivtt -u2 -c1
```

The meaning of the processing options is described in the following sections / given in Table 5.

3.2.2 Processing annotations – comments and foliation

To remove annotations (comments, foliation) from the file, the three options `-c` , `-f` , and –p are available. Their function is given in the following table. The file header looks like a comment line, but since it has decodable information, it is not treated like a comment. It can be removed together with the foliation information using the `-f` option. Also text tags are removed by using the `-f` option.

Table 4: annotation-related options

Option	Meaning	Comment
`-c0`	Keep all comments	
`-c1`	Remove hash (#) comment lines	This does **not** affect the file header.
`-c2`	Remove inline comments, but keep page header comments	Dedicated comments are not affected. For these see the –p option
`-c3`	Remove both inline comments and page header comments. This also remove text tags.	Dedicated comments are not affected. For these see the –p option
`-c4`	Remove hash (#) comment lines and inline comments, but keep page header comments.	Dedicated comments and the file header are not affected
`-c5`	Remove hash (#) comment lines, inline comments and page header comments. This also remove text tags.	Dedicated comments and the file header are not affected
`-f0`	Keep all foliation information in the file	
`-f1`	Remove all foliation information. This will also remove spaces before the transliterated text.	This will also remove the file header and all text tags.
`-f9`	Ignore locus layouts. This can be used to process old files or files not properly conforming to the IVTFF format. It will still interpret page headers	More information is provided in Section 4. Most old files cannot be processed by `ivtt` for other reasons.

`-p0`	Keep paragraph start/end codes and illustration intrusion codes as they are	
`-p1`	Remove illustration intrusion codes and strip the paragraph start/end codes.	
`-p2`	Replace illustration intrusion code by additional space and the paragraph end code by an additional newline.	The symbol used for space is the 'certain space' as selected with the `-s` option (see below)

Notes:

- The `–f9` option can be used in combination with the `–f1` option

3.2.3 Options related to characters

This includes options related to swapping between different notations for rare symbols and ligatures, and for processing alternative readings, which are presented in the table below.

Table 5: character processing options

Option	Meaning	Comment
`-a0`	Leave high ascii and `@nnn;` codes unchanged	
`-a1`	Convert high ascii values (>127) to `@nnn;`	
`-a2`	Convert `@nnn;` codes to a single high ascii byte	
`-l0`	Keep ligature brackets `{` and `}`	
`-l1`	Remove ligature brackets (but keep text contained between them)	
`-l2`	Obsolete option	If selected a warning will be issued and the option is ignored.
`-l3`	Turn ligature brackets into capitalisation rule, i.e. turn `{cto}` into `CTo`, etc.	The opposite option seems to be missing. It needs to be (re-) implemented
`-l4`	Same as `-l3` but additionally convert EVA: `sh` to `Sh`, `cth` to `cTh`, `ckh` to `cKh`, `cph` to `cPh` and `cfh` to `cFh`	
`-u0`	Keep all alternate readings as they are (that is, keep the square brackets and everything contained between them)	
`-u1`	Pick the first of alternate readings, e.g. replace `k[o:a]iin` by `koiin`, and `chee[?:y]` by `chee?`	

-u2	Replace alternate reading by a ?. k[o:a]iin will now become k?iin.	
-u3	Replace entire words containing alternate reading (k[o:a]iin) or uncertain reading (k?iin) by ???	If the word had only one or two characters, it will be replaced by ? or ?? respectively
-u4	Remove entire words containing alternate reading or uncertain reading from the line	Keeps both spaces surrounding the word
-u5	Remove entire line if it contains alternate or uncertain readings	

Notes:

- The syntax for 'alternative readings' according to IVTFF version 1.4 is: [a:b] , but older files will have [a|b]. This will no longer be recognised by ivtt version 1.1 (and higher).

3.2.4 Options related to whitespace and line wrapping

How to process the certain and uncertain spaces in the MS, and whitespace in the transliteration file, is controlled using the options in Table 6. This also includes an option to produce an output file that has one word per line.

Table 6: space-related options

Option	Meaning	Comment
-b0	Keep blanks (whitespace) in file	
-b1	Remove blanks. It also causes that empty lines will not be written to the output file.	If combined with the -s1 option, the spaces resulting from dots in the input file are still written to the output file.
-h0	Keep comma for uncertain spaces (those marked with a comma).	
-h1	Treat uncertain spaces as normal (certain) spaces (applying -s option)	The way certain spaces are treated is taken from the -s option
-h2	Remove uncertain spaces	
-h3	Turn uncertain spaces into ? and at the same time invoke the option to have words containing a ? translated into ???	This does not affect the option settings for processing uncertain reading brackets [ ]. This option has not yet been tested.
-s0	Keep dot for certain spaces (those marked with a dot).	
-s1	Convert certain spaces (those marked with a dot) to blanks	

`-s2`	Remove certain spaces	This automatically invokes the `-h2` option as well
`-s3`	Convert certain spaces (those marked with a dot) to newlines, i.e. the output file will have one word per line.	This automatically invokes option `–f1` to remove foliation, and `–w1` to unwrap any wrapped lines. The second cannot be undone by a later –w option.
`-w0`	Keep line wrapping as is	
`-w1`	Unwrap all lines. Continuation lines are concatenated	
`-w2`	Wrap lines at a maximum of 40 characters, always wrapping after a hard space (blank or dot). Continuation lines are started with / followed by one blank.	If spaces are removed from the file with the `-s2` option, this has the result that line wrapping may take place anywhere.
`-w`*n*	With *n* in the range of 2-9: same as `-w2` but set the maximum width to 20**n* (i.e. 40, 60, ... , 180).	Same comment as above

3.2.5 Miscellaneous options

The final group has all remaining options.

Table 7: processing options

Option	**Meaning**	**Comment**
`-m0`	Normal output to `<stderr>`.	
`-m1`	Suppress output to `<stderr>`, except for warning and error messages.	
`-m2`	Suppress all output to `<stderr>`.	If an error occurs before the option has been parsed, the error message will be printed
`-q0`	Keep all lines of locus family P in the file, regardless whether they are paragraph-initial or not	
`-q1`	Of all lines of locus family P, keep only paragraph-initial lines.	This does not affect lines of other locus types. Side effect: <%> and <$> codes are removed from the file.
`-q2`	Of all lines of locus family P, keep all lines except paragraph-initial lines.	Same comment as for –q1.
`-x`*n*	With *n* in the range of 0-8: shortcut option.	See Table 8.

Notes:

- The options –q1 and –q2 will include all other locus types in the output file. The desired effect for most cases is obtained by adding the +@P option, i.e. leaving all types of paragraph loci. There are a few blocks of 'dislocated' paragraph-like text (usually short lines). These have no paragraph markers. They would all be included in the output of a –q2 option, and none would be included in the output of a –q1 option.

For convenience, there are also a few 'shortcut' options, which can be combined with anything else (including other shortcut options). Table 8 shows the equivalent list of options for each of them. Note that the option –x0 is not the same as specifying no shortcut option.

Table 8: shortcut options

Option	Equivalent with:	What it does
`-x0`	`-c1 -p1 -w1 -b1`	To obtain one line per locus
`-x1`	`-c5 -f1 -p1 -b1`	To keep only Voynich text
`-x2`	`-c5 -f1 -p1 -u1 -b1`	As -x1 but take first of alternate readings
`-x3`	`-s1 -h1`	Certain and uncertain spaces converted to blanks
`-x4`	`-s1 -h2`	Certain spaces converted to blanks and remove uncertain ones
`-x5`	`-l1 -a2`	Convert special characters for statistics analysis
`-x6`	`-l4 -a2`	Prepare special characters in a way suitable for TTF font display
`-x7`		Turn into a text-only Ascii file, preserving uncertain spaces
`-x8`		Turn into a text-only Ascii file, removing uncertain spaces

4 Processing non-standard files

In general, `ivtt` will correctly process files that are conforming to the IVTFF format.

The most important restrictions are:

1. Alternative locus identifiers as described in [R-1] Section 6.5 are not supported by `ivtt`.
2. All paired brackets that are opened must be closed on the same line. These symbols may appear freely inside comments, where they are not interpreted and don't need to be matched by a closing bracket.
3. Page headers are identified by the absence of period (.), comma (,), and semi-colon (;) in locus ID's.
4. All other locus ID's must exactly conform to the definition in Ref. [R-1].

Also, plain Ascii files that have passed through `ivtt` once (and may have lost some parts) will generally be supported in subsequent runs of `ivtt`.

There is one processing option (`-f9`) that allows the processing of files that do not conform to point 4 above. In this case, loci that are not page headers should have at least one period, comma or semi-colon among its string. If such files are not compatible with other restrictions, `ivtt` will not be able to process them.

An example of a non-standard file that can be processed using `–f9` is the original v101 transliteration file.

I have created a modified version of this file (`GC_ha.txt`), with exactly the following changes:

- A newline has been added at the end of the last line. In the original file, the 'end-of-file' marker follows without a newline after the last line, causing `ivtt` to ignore the last line.
- All high-ascii codes have been replaced by the corresponding @nnn`;` code. This was done to make the file safer to copy and edit. It was done using the following command (which can be reversed in an equivalent manner):

```
ivtt -f9 –a1 GC.txt GC_ha.txt
```

- The standard IVTFF page headers with variable definitions have been added

As a result, `ivtt` can be used to perform (a.o.) the following operations on the file:

- Select pages based on variable settings
- Remove foliation information
- Toggle high-ascii representation
- Wrap and unwrap lines

It is to be noted that the `-f0` and `-f1` options can be used in combination with `-f9`.

5 Diagnostics messages

5.1 Warnings

Warnings will allow the tool to continue. Warnings indicate the offending character and the line read up to the point where the problem is detected. Much of this is still to be double-checked in the tool, and new warnings may be added.

The following warnings may occur:

| warning: if square brackets are found without a vertical bar in between (needs to change)

$ warning: if a variable definition is detected in the first line of a folio, but the character following the $ is not in the range A-Z. This illegal sequence is then ignored.

{ warning: if a { is found inside [..] or (..) . May not be relevant anymore

} warning: To be reviewed…

5.2 Errors

Errors cause the tool to stop. Errors are generally detected by the line parser, and result in one of the following messages:

Illegal bracket inside <..>

  if one of: { [or (appears inside the foliation.

Illegal second open bracket

  if one of: { [or (is not closed before it appears again

Illegal close bracket

  if one of: }]) or > appears without a matching open bracket

Foliation field too short

  if > appears too soon after <

Illegal bar without [

  if a | is found without an opening [

These messages are followed by the lines:

```
Offending character: X
```

```
Line parsed so far: <line>
```

Here X is the offending character and `<line>` was the string of text parsed so far. If this happens while reading the line from the input file, it is followed by the message:

```
Error reading line from <...>
```

If it happens during a later stage there will be a message indicating which processing is on-going and the original line that was read in is displayed as well.

Other error messages which require no further explanation are:

- Error parsing command line
- Error opening file(s)

6 Operating system

This command is based on C code written and compiled under both Unix and Linux. In principle, it can be compiled in other operating systems as well.

In Linux (and Unix) the standard input and output may be redirected. If the shell is a Bourne shell it is also possible to redirect the diagnostics to a file using the 2> redirection symbol.