Regular Expressions in Create Lists (Revised 2010)

1. Literal Characters and Metacharacters

Regular expressions are formed from combinations of literal characters and metacharacters. Literal characters are characters that represent themselves in a matching field. They include letters, numbers, the space, and most marks of punctuation. In Create Lists, the vertical bar ("|"), which is used as the subfield delimiter, is a literal character.

Metacharacters are those that perform some function within the regular expression. Some metacharacters work in combinations that can be termed "metasequences." Any metacharacter can be turned into a literal character by "escaping" it, that is, by preceding it with a backslash. The metacharacters and metasequences used in Create Lists are shown in the table below.

Regular expressions are invoked using the "matches" condition in Create Lists. Like all searches in Create Lists, regular expressions in Create Lists are always case insensitive.

	Character Classes
.	Period (or "dot"). Matches any single character.
[...]	User-defined character class. Matches any single character that is included in the class. Examples: `[aie]` the letter *a*, *e*, or *i* (upper or lower case) `[a-z]` matches any single letter (upper or lower case) `[a-z0-9]` matches any one letter or digit `[av-z]` matches any one of the letters *a*, *v*, *w*, *x*, *y*, or *z* `['"]` matches a single quote or a double quote `[- ,.]` matches a hyphen, space, comma or period (Note in the last example that the hyphen must come first so as not to be interpreted as a range indicator. Also note that the period character is treated as a literal within a character class.)
[^...]	Negated character class. Matches any single character that is *not* in the class. Examples: `[^ ]` any character that is not a space `[^0-9a-z]` any character that is not a digit or letter
	Quantifiers
+	Plus. Matches when the preceding character (or group of characters) occurs 1 or more times. Example: `|a0+523` matches "|a0523", "|a00523", "|a000523", etc.
*****	Asterisk (or "star"). Matches when the preceding character (or group of characters) occurs 0 or more times. Examples: `|a0*523` matches "|a523", "|a0523", "|a00523", etc. `.*` matches any number of any characters

Richard V. Jackson, Huntington Library
rjackson@huntington.org

	Quantifiers (continued)

{ *min* , *max* }
{ *num* }
	Matches when the preceding character (or group of characters) occurs at least *min* times, but no more than *max* times. When one number is given, the character (or group of characters) must occur exactly *num* times. The largest number that may be used in a quantifier is 127. Examples:
	`[a-z]{5,8}` — matches a string of 5 to 8 letters
	`[0-9]{4}` — matches a string of 4 numbers
	`.{100,125}.{100,125}` — matches a string of 200 to 250 characters
	(These examples may match fields containing strings longer than the maximum value indicated, unless literal characters or other more specific subexpressions are included both before *and* after.)

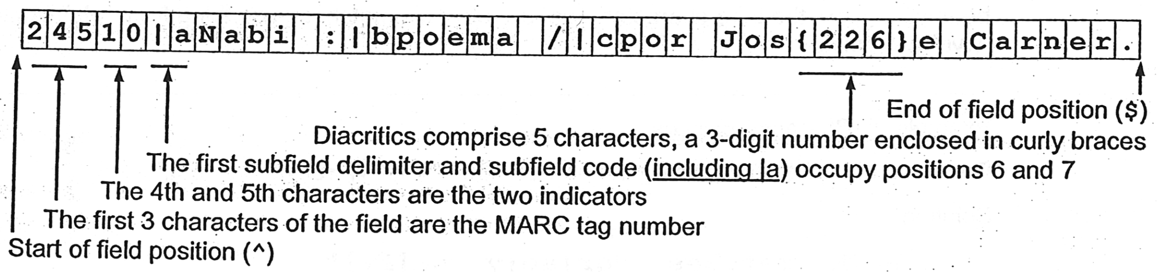
	Grouping
(...)	Allows a quantifier to apply to a group of characters. Example: `Melvil(le){0,1} Dewey` — matches "Melville Dewey" or "Melvil Dewey" (the string "le" occurs 0 or 1 times)
	Position Indicators
^	Beginning of the field position. Anchors what follows to the start of the field. (^ must be the first character in the expression.) More information on p. 3.
$	End of field position. Anchors what precedes it to the end of the field. ($ must be the last character in the expression.) Examples: `[^.]$` — matches when the last character in the field is not a period. (Note that the "^" within the brackets indicates a negated character class, not the beginning of the field.) `^245..\|a.{0,5}$` — matches a 245 tag with 5 or fewer characters between |a and the end of the field
	Backslash ("Escape")
\	Causes the character that follows to be interpreted as a literal character. Examples: `\.\.\.$` — matches 3 periods at the end of the field `\{[0-9]{3}\}` — matches any diacritic (3 digits within braces) `\$5[0-9][0-9]\.` — matches "$"–"5"–number–number–decimal point (dollar values between $500 and $599)

2. How Millennium Stores MARC Variable Fields:

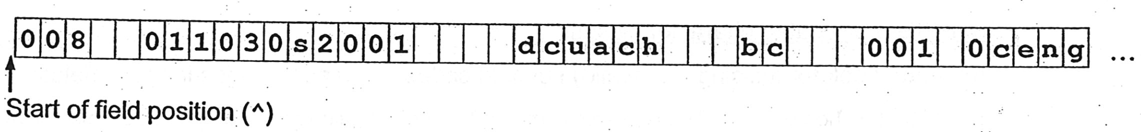
Successful use of regular expressions, particularly the position indicators ("^" and "$"), requires understanding how Millennium stores MARC format data. Some examples will illustrate:

- A MARC variable field includes a tag number, indicators, subfield delimiters, subfield codes, and data. The vertical bar ("|") is used as the subfield delimiter.

```
245   10   Nabi :|bpoema /|cpor José Carner.
```

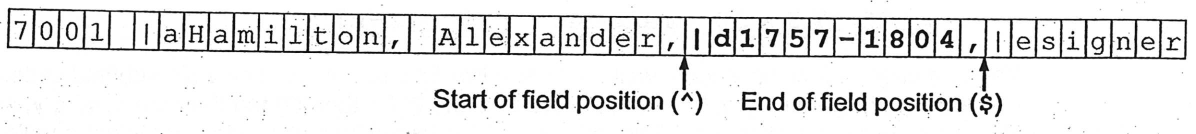

24510|aNabi :|bpoema /|cpor Jos{226}e Carner.
End of field position ($)
Diacritics comprise 5 characters, a 3-digit number enclosed in curly braces
The first subfield delimiter and subfield code (including |a) occupy positions 6 and 7
The 4th and 5th characters are the two indicators
The first 3 characters of the field are the MARC tag number
Start of field position (^)

- 00X control fields do not use subfield delimiters or codes, but do include 2 spaces for indicators.

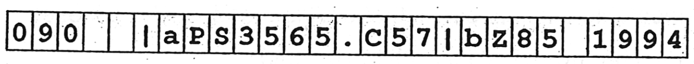

008 011030s2001   dcuach bc 001 0ceng ...
Start of field position (^)

- When a particular MARC tag subfield is specified in a search statement, the "field" begins with the delimiter for that subfield and ends just before the next delimiter, or at the end of the entire field.

Example: **MARC TAG 700<u>|d</u> matches "^|d175"**

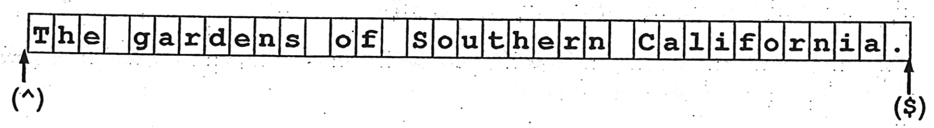

7001 |aHamilton, Alexander,|d1757-1804,|esigner
Start of field position (^)
End of field position ($)

- The normalization rules that may apply to call numbers affect indexing only, not the way the call numbers appear to Create Lists.



- Non-MARC fields are stored without MARC tags, indicators, or subfield codes.

```
TITLE    The gardens of Southern California.
```



3. Examples of Regular Expressions in Create Lists

Example 1: Use of the "dot" meta character

Problem: You are asked to limit a search to titles published in the United States. Unfortunately, the fixed-length field COUNTRY uses separate codes for each of the 50 states, D.C., etc.

Solution:

```
COUNTRY  matches  "..u"
```

All U.S. codes have "u" as the third character. With this expression, the first two characters can be anything, but the third character must be "u". This matches everything from "alu" (Alabama) to "wyu" (Wyoming).

Example 2: Character classes

Problem: Find notes containing the phrase "*gray* [or *grey*] *wolf* [or *wolves*]."

Solution:

```
NOTE  matches  "gr[ae]y wol[fv]"
```

Note that a character class (without a quantifier) must match one and only one character.

Example 3: Negated character classes

Problem: Look for missing (or invalid) subfield codes, such as the ones in these fields:

```
245 10 Love for love|[microform] :|ba comedy /|cby William Congreve.
                     ^
650  0 United States|xHistory|Civil War, 1861-1865|vAnecdotes.
                             ^
```

Solution:

```
MARC TAG 245  matches  "|[^abcfghknps6]"

MARC TAG 650  matches  "|[^avx-z]"
```

Negated character classes work well for finding invalid characters. The valid subfield codes for the 245 tag are generally a, b, c, f, g, h, k, n, p, s and 6; for the 650 tag they are a, v, x, y, and z. These expressions match a subfield delimiter ("|") followed by a character that is not a valid code.

Example 4: Use of "dot-star"

Problem: Look for non-repeatable subfield codes that are repeated, such as these:

```
245 10 Deception :|ba novel /|by Philip Roth.
245 10 California :|ca history /|cAndrew F. Rolle.
```

Solution:

```
   MARC TAG 245  matches  "|b.*|b"
OR MARC TAG 245  matches  "|c.*|c"
```

The construction "dot-star" (`.*`) is often used as a placeholder for "any number of any characters" between two more specific sub-expressions.

Example 5: More with "dot-star"

Problem: Create a bibliography of anything relating to 18th century France.

Solution:

```
     SUBJECT  matches  "france.*18th cent"
  OR SUBJECT  matches  "france.*17[0-9][0-9]"
```

This search will match all these headings and more:

```
651  0 France|xHistory|yRevolution, 1789-1799|xArt and the revolution
600 00 Louis|bXIV,|cKing of France,|d1638-1715|xArt collections
651  0 France|xIntellectual life|y18th century
650  0 Books and reading|zFrance|xHistory|y18th century
650  0 Architecture|zFrance|zParis|y18th century
651  0 France|xPolitics and government|y1789-1815
650  0 Printing|zFrance|xHistory|y18th century
650  0 Republicanism|zFrance|xHistory|y18th century
650  0 Individualism|xSocial aspects|zFrance|xHistory|y18th century
650  0 Ethnopsychology|zFrance|xHistory|y18th century
651  0 Paris (France)|xHistory|y1799-1815
651  0 Lyon (France)|xHistory|xSiege, 1793
651  0 Paris (France)|xSocial life and customs|y18th century
600 10 Douglas, Frances,|cLady,|d1750-1817          [false drop]
```

Example 6: "Escaping" a metacharacter

Problem: The diacritics grave ("`") and acute ("´"), stored as "{225}" and "{226}" respectively, have occasionally been used incorrectly as single quotes or apostrophes in note fields:

```
500   Originally written for {225}The Youth's companion{226}.
500   Includes publisher{226}s advertisements on covers.
```

Solution:

```
     NOTE  matches  "\{22[56]\}[^aeiou]"
  OR NOTE  matches  "\{22[56]\}$"
```

To search for diacritics, the curly braces (which are normally metacharacters) must be preceded by a backslash. This search matches a grave or acute that is followed by any character that is not a vowel, or a grave or acute that is the last character in the field. This search may also find other kinds of errors (e.g., Nez Perce{226}).

Example 7: The {min,max} quantifier

Problem: Find words that may be spelled differently, or phrases with optional words.

Solution:

```
   ORDER NOTE  matches  "cancel{1,2}ed"

   ORDER NOTE  matches  "acknowledge{0,1}ment"

        TITLE  matches  "Thomas (Alva ){0,1}Edison"
```

Richard V. Jackson, Huntington Library
rjackson@huntington.org

Example 8: Position indicators

Problem: Find subject headings with second indicators that are not 0 (Lib. of Congress), 5 (Natl. Lib. of Canada), or 7. Also find those with 2nd indicator 7, where the last subfield is not |2.

Solution:

```
     SUBJECT  matches   "^6...[^057]"
  OR SUBJECT  matches   "^6...7.*|[^2][^|]+$"
```

The first search statement matches a 6 at the start of the field, followed by any 3 characters (for the rest of the tag number and the first indicator), followed by a character that is not 0, 5, or 7. The second search statement matches a 6 at the beginning, a 7 as the second indicator, any number of any characters, then a subfield delimiter with a subfield code other than 2, followed by one or more characters that are not another subfield delimiter, followed by the end of the field. The last part ensures that the subfield code that is not |2 is the last subfield in the field.

Example 9

Problem: Find system control numbers that are exactly 4 digits long in order to use Global Update to insert 2 leading zeroes.

Solution:

```
     MARC TAG 035  matches   "|a[0-9]{4}$"
  OR MARC TAG 035  matches   "|a[0-9]{4}[^0-9]"
```

This search matches |a, followed by exactly 4 digits, followed by either the end of the field or a character that is not a digit. Global Update can then be used to insert "00" at the start of the field.

Example 10

Problem: Find records where the title proper (MARC tag 245 |a) is longer than 256 characters.

Solution:

```
     MARC TAG 245|a  matches   "|a.{100}.{100}.{57}"
```

The maximum value for any quantifier is 127, so the expression "`.{257}`" won't work. Use instead any combination of multiple quantified subexpressions to achieve the desired maximum. This expression finds subfields that *contain* 257 characters; they may be longer than that.

Example 11

Problem: Some records have "empty" tags – fields with no data or perhaps with only a "|a".

Solution:

```
     MARC TAG ???  matches   "^.{1,7}$"
```

Note that you can search *all* MARC tags at once by specifying a MARC tag (using "!") and entering 3 question marks (or 3 spaces) for the tag number. (You can also specify partial tag number matches, for example, "`5??`", "`76?`", "`?10`", etc.) This expression must include both the beginning and end of field indicators in order to match tags with 7 or fewer characters. The minimum value cannot be 0, or all non-MARC fields will match.

This search may match some valid 007 tags, e.g., "007 ta". You can avoid including 00X tags by searching: `MARC TAG ??? matches "^.[^0].{0,5}$"`

Richard V. Jackson, Huntington Library
rjackson@huntington.org

Example 12: Using the fixed-length fields *(newly revised)*

Problem: Limit a search to titles published *outside* the United States. (This is the complement of Example 1.)

Solution:

```
COUNTRY  matches  "[^u]$"
COUNTRY  matches  "^..[^u]"      [Alternate version]
```

These match records where the last character of the fixed-length field COUNTRY is not "u".

This search could not be done in earlier releases (prior to Release 2005?), and it still will not work in the character-based system. In earlier releases, a regular expression used with a fixed-length field could not contain more characters than the field itself (3 in the case of COUNTRY). It was necessary to use 008 bytes 15-17 to do this search.

That restriction for the fixed-length fields no longer applies. Here is another example:

```
COUNTRY  matches  "[^oq].c"
```

This will match Country codes for any place in Canada outside of Ontario and Québec.

This change means you can now use regular expressions even with single-character fixed-length fields (even though the User Manual still implies otherwise). See Example 18 for an example.

Example 13

Problem: Some bad data, such as Line Feed characters (ASCII 10), have somehow gotten into some records.

Solution:

```
MARC TAG ???  matches  "[^ -~]"
```

This matches any character outside of the range <space> (ASCII 32) to "~" (tilde, ASCII 126).

Example 14

Problem: Some titles have incorrect filing indicators, such as:

```
245 04 Grand Tour :|bthe lure of Italy in the eighteenth century
245 04 A letter from a Gentleman of the City of New-York to another
```

(The first title is indexed as "d tour the lure of italy...", the second as "tter from a gentleman...")

Solution:

```
    MARC TAG 245  matches  "^245.2|a.[^ '"]"
OR  MARC TAG 245  matches  "^245.3|a..[^ '"]"
OR  MARC TAG 245  matches  "^245.4|a...[^ '"]"
[etc.]
```

These expressions depend on the fact that for a filing indicator *n*, the *n*th character following "|a" should normally be a space, apostrophe/single quote, or double quote. The above expressions match when the *n*th character is not one of these characters. (This may not work if the initial article contains a diacritic; diacritics count as one filing character, but are stored as 5 characters.)

Example 15

Problem: Find subjects headings that have a second indicator 4, including those in MARC tags 600, 610, 611, 630, 650, and 651, but not including 655 or 690.

Solution:

```
    Subject  matches  "^6[0135][01].4"
```

Use "^" to anchor the expression to the beginning of the field. In this expression, 655s are excluded because the 3rd digit can only be 0 or 1. 690s are excluded because the 2nd digit cannot be 9. The first indicator can be any character, but the second indicator must be 4.

Example 16

Problem: The height of books is usually given in centimeters in MARC 300 |c. However, some records give both height and width (23 x 30 cm), some use millimeters, and some older records use inches or designations such as "folio," "8vo," etc. For the purpose of simplifying a size analysis of a collection, extract a list of records with "clean" size designations.

Solution:

```
     300  matches  "|c[1-9][0-9] cm[. ]{0,1}$"
  OR 300  matches  "|c[1-9][0-9] cm[. ]{0,1}.{0,2}|"
```

This matches 2-digit centimeter statements, with or without a period, either at the end of the tag or followed after no more than 2 characters by another subfield delimiter.

Example 17

Problem: In MARC tag 856 (Electronic location and access), a clickable "linking" text may be generated in the OPAC from |z (Public note) or |3 (Materials specified). Find records with 856s that contain neither |3 nor |z.

Solution:

```
    MARC TAG 856  matches  "^856..(|[^3z][^|]*)+$"
```

Use both the start and end of field position indicators, and account for the tag number and indicators at the beginning. The parentheses group together the characters of a single subfield, here specified as a subfield delimiter ("|"), followed by a subfield code that is not "3" or "z", followed by any number of characters that are not "|". The "+" indicates one or more such subfields occurring until the end of the field is reached.

It might seem simpler to use:

```
    MARC TAG 856  at least one field does not have  "|z"
AND MARC TAG 856  at least one field does not have  "|3"
```

However, a single record may have multiple 856s. The above search may find records where one 856 lacks "|z" and another lacks "|3." Only the regular expression guarantees finding instances of a single field that lack both subfields.

Example 18

Problem: Find records that contain a bad code in the fixed-length field MAT TYPE (aka Bcode2).

Solution:

```
        MAT TYPE  matches   "[^ac-gijkmoprt]"
```

Earlier restrictions on the use of regular expressions with fixed-length fields no longer apply (see Example 12), and it is now possible to use them even with single-character codes. This can greatly simplify finding invalid codes. The above search finds characters that are not one of the valid codes as defined in MARC21; any locally-defined codes should also be accounted for.

Example 19

Problem: Find books where the number of pages given in 300 |a is 25 or fewer.

Solution:

```
       300|a  matches  "[^0-9][1-9] p\."
    OR 300|a  matches  "[^0-9]1[0-9] p\."
    OR 300|a  matches  "[^0-9]2[0-5] p\."
```

Separate search statements are needed to find books with 1-9, 10-19, and 20-25 pages. Including "[^0-9]" at the beginning of each expression ensures that values such as "6<u>23 p.</u>" are not matched, while still permitting matches on values such as "vi, 23 p." Unfortunately, this search will also match, for example, "123 p., <u>16 p.</u> of plates", and will fail to match "[8] p."

Example 20

Problem: Find bib records that have two (or more) call number fields. The only characteristic that reliably distinguishes the call numbers is that one is always longer than the other.

Solution:

```
         (   CALL #  matches  "|a.{6}$"
   AND       CALL #  matches  "|a.{7}"     )
   OR    (   CALL #  matches  "|a.{7}$"
   AND       CALL #  matches  "|a.{8}"     )
   OR    (   CALL #  matches  "|a.{8}$"
   AND       CALL #  matches  "|a.{9}"     )
   OR    [etc.]
```

This search assumes that the shortest possible call number contains 6 characters following "|a". The first pair of search statements matches records where one call number has exactly 6 characters and another call number in the same record contains at least 7 characters. (Note the presence or absence of the end of field indicator ($).) The second pair of search statements increments these lengths by one, finding one call number with exactly 7 characters and another with 8 or more. You will need to add enough pairs of statements to reach the maximum likely length for the call number. The parentheses shown grouping each pair of statements above are not actually necessary (Create Lists always evaluates Boolean AND before OR), but are included here for the sake of clarity.

4. Metacharacters That Do *Not* Work in Create Lists

Several metacharacters and metasequences that are available in other implementations of regular expressions do not work in Create Lists. Some of these are listed here, along with some possible workarounds.

?	Used to indicate that the preceding character (or group of characters) is optional. In Create Lists, the question mark is a literal character. *Workaround:* Use the quantifier "`{0,1}`" to indicate that a character (or group of characters) occurs 0 or 1 times.
`(...|...)`	Used to indicate two or more alternative strings of characters, for example "`(donation|gift)`". In Create Lists, "`|`" is a literal (the subfield delimiter). *Workaround:* There is no workaround within a single regular expression; however, you may be able to accomplish the same thing with multiple search statements linked with Boolean OR.
`(...)...\1`	Back references. Allows a string of text to be "captured" so the same text may be matched on again later in the same expression (or may be inserted in the replacement text). Back references are not available in Create Lists. There is no workaround. However, back references are primarily useful in match-and-replace operations, and would have limited use in Create Lists.
`\w`, `\d`, `\b`, `\S`, `\<`, `\>`, etc.	Special character classes, positions, etc., such as "`\w`" (any alphanumeric word character) or "`\b`" (a word boundary). None of these work in Create Lists. The backslash is used only to change a metacharacter into a literal character. (Backslashing a literal character simply results in the same literal character.) *Workaround:* Most of these character classes are easily rendered using other metacharacters. For example, "`\d`" (any digit) can be rendered as "`[0-9]`". Using word boundary positions is problematic in MARC format anyway, because of the use of letters for subfield codes.

5. Further information

The User Manual (page # 100672 for Millennium; #101608 for Innopac) gives only a little information on the "matches" condition in Create Lists. There is a wealth of information on the Web and elsewhere on regular expressions; however, much of it is confusing, misleading, or not applicable to Create Lists. Most of what I have learned has been through a combination of reading and experimentation. I also found the following book to be very useful:

Friedl, Jeffrey E. F. *Mastering Regular Expressions*, 3rd edition. Sebastopol, CA: O'Reilly, 2006. 515 pp.

Richard V. Jackson, Huntington Library
rjackson@huntington.org