Authority file comparison rules
Revised 2007-01-11

Introduction

When a new authority record is added to a file of authority records, each access field[1] in the new record must be compared to access fields already present in records in the file to determine whether the new access fields are adequately differentiated from, or adequately similar to, existing access fields. (Under certain conditions, two access fields *must not* be found to be identical during this comparison; under other conditions, two access fields *must* be found to be identical.) Similarly, when a new access field is added to an existing authority record, the new field must be compared to existing access fields to ensure that the new access field is adequately differentiated from, or adequately similar to, existing access fields.

When determining whether access fields are sufficiently similar or dissimilar, access fields are not compared in their native forms as found in authority records, but are transformed according to a set of rules; and it is the transformed versions that are compared. The transformation process[2] described in this document removes distinctions felt to be irrelevant to the purposes of this comparison, and retains distinctions felt to be critical.[3]

The comparison rules defined in this document are designed to achieve a single goal: determining whether or not two access fields are sufficiently different, or sufficiently similar, for the purpose of adding access fields to authority records. These comparison rules are most emphatically not designed for or intended to be used in the many other comparison tasks performed by library workers and library systems.[4] Each of these other tasks has, or should have, a comparison form uniquely tailored for its special ends; a single comparison form can *never* serve all such purposes. A library system that uses the rules defined in this document only for the limited purpose described in this document may be said to conform to this document; no library system may make any claim that it implements the rules defined in this document for any other purpose.

The use of these rules to compare two headings is part of an operation that involves other sets of rules and standards. For example, the application of these rules to an authority 1XX and an authority 4XX field can show that the fields should be considered identical and that therefore a change of some kind to one or the other is needed, but these comparison rules do not prescribe the various options available to resolve the conflict; suggestions for such resolution are contained in

---

[1] *Access fields* are MARC21 authority fields in the 1XX, 4XX and 5XX groups.
[2] Although the term *normalization* is commonly used for this process within the library community, this term is also used in that community and elsewhere for other operations in character handling. In order to avoid confusion, the term *normalization* is avoided in this document.
[3] For example, lower-case alphabetic characters are transformed into upper-case alphabetic characters because a difference in casing is considered irrelevant for the purpose of comparing access fields; but subfield codes are retained because a difference in subfield coding is considered important for the purpose of comparing access fields.
[4] Among the tasks for which the operation described in this document is *not* designed are these: the arrangement of access fields in a sorted list; the comparison of user-input search terms to access fields; matching access fields in authority records to variable fields in bibliographic records.

other bodies of rules (cataloging rules, the *NACO users' manual*, etc.). Other rules may impose constraints on the content of access fields beyond those imposed by these comparison rules.

These comparison rules are designed only for application to complete access fields. A separate scheme may be devised for comparing parts of access fields.[5]

The transformation described in this document is expressed in terms of the set of code points ("characters," approximately) defined in the current edition of *The Unicode standard*[6] and associated documentation. Although these rules may be adapted for use with other character sets (such as the MARC-8 character set), no reference is made in this document to such other character sets. These rules are expressed in terms of MARC 21 coding, but they may be applied to authority data conveyed with other labels.

Rules for Comparison

1. Compare only records that may be compared

These rules are designed to be applied to any given set of authority records, whether those records reside in one or more physical files. Although these records allow for the comparison of all records from all sources, there are two exceptions:

- access fields in authority records with differing subject heading system codes (MARC 21 008/11) are not compared[7]
- access fields in LC/NACO name authority records are not compared to access fields in LCSH authority records even though they have the same subject heading system code

2. Selection of subfields

Select all subfields from candidate variable fields except the following:

- subfields with numeric codes ($0-$9)
- subfield $e; but retain subfield $e in conference headings (X11 fields)
- subfield $i (used in 4XX and 5XX fields)
- control subfield $w

3. Preparation of subfield text for comparison

For each subfield in the field of interest to be retained for the purposes of comparison:

---

[5] For example, a scheme might be devised for comparing the name portion of a name/title heading to the heading in the authority record that authorizes the name alone.

[6] *Unicode*™ is a trademark of the Unicode consortium.

[7] For example, access fields in authority records for Medical Subject Headings (008/11='c') are not compared to access fields in authority records for Library of Congress Subject Headings (008/11='a') even if they reside in the same physical file.

1) If the subfield contains text marked with U+0098/U+009C pairs, remove[8] those characters and everything between them. If the subfield does not contain U+0098/U+009C pairs and if the subfield code of the subfield is $a and if the field contains a nonfiling characters indicator whose value is higher than zero, remove from the beginning of the subfield the number of code points indicated by the nonfiling characters indicator.

2) Remove leading and trailing spaces.

3) If the subfield text is now a null string, omit the subfield from the comparison heading.

4) If any code point in the subfield is listed in the *Unconditional mappings* section of the Unicode special casing file, replace the code point with its uppercase equivalent.

   Examples:

UTF-16 value	**Substitution**
U+00DF Latin small letter sharp s	U+0053 Latin capital letter S plus U+0053 Latin capital letter S
U+FB00 Latin small ligature ff	U+0046 Latin capital letter F plus U+0046 Latin capital letter F

5) Apply Unicode compatibility decomposition[9] to the subfield text. Perform this action recursively, until no further changes are possible.[10]

[8] Here and elsewhere, forms of the verb *remove* are used to indicate that one or more code points drop out of the text and the code points adjacent to the removed code points are brought together. This action is not the same as replacement of a code point by the space character.

[9] Unicode normalization form KD: replace each code point with the code point or code points indicated in position 6 of each code point's entry in the Unicode character database, minus any qualifier given within angle brackets. If position 6 of a code point's entry in the Unicode character database is null, make no change in this step.

[10] To save processing steps, an implementation may choose to apply the rules given in the remaining steps to the substitution performed in this step. For example, because U+031B Combining horn will later be removed, an implementation may choose not to insert that character here as part of the compatibility decomposition; similarly, because lowercase characters will eventually be replaced by uppercase characters, an implementation may choose to use uppercase characters where lowercase characters are indicated.

Examples:

UTF-16 value	**Substitution**[10]
U+01A0 Uppercase O-hook	U+004F Latin capital letter O
U+01A1 Lowercase o-hook	U+004F Latin capital letter O
U+01AF Uppercase U-hook	U+0055 Latin capital letter U
U+01B0 Lowercase u-hook	U+0055 Latin capital letter U
U+0133 Latin small ligature ij	U+0049 Latin capital letter I plus U+004A Latin capital letter J

Examples:

UTF-16 value	Substitution
U+01A0 Uppercase O-hook	U+004F Latin capital letter O plus U+031B Combining horn
U+01A1 Lowercase o-hook	U+006F Latin small letter o plus U+031B Combining horn
U+01AF Uppercase U-hook	U+0055 Latin capital letter U plus U+031B Combining horn
U+01B0 Lowercase u-hook	U+0075 Latin small letter u plus U+031B Combining horn
U+00B2 Superscript two	U+0032 Digit two[11]
U+0132 Latin capital ligature IJ	U+0049 Latin capital letter I plus U+004A Latin capital letter J
U+0133 Latin small ligature ij	U+0069 Latin small letter I plus U+006A Latin small letter j

6) Perform additional substitutions for those code points listed in the following table.[12]

UTF-16 value	Substitution
U+00C6 Uppercase digraph AE	U+0041 Latin capital letter A plus U+0045 Latin capital letter E
U+00D8 Uppercase Scandinavian O	U+004F Latin capital letter O
U+00DE Uppercase Icelandic thorn	U+0054 Latin capital letter T plus U+0048 Latin capital letter H
U+00E6 Lowercase digraph ae	U+0061 Latin small letter a plus U+0065 Latin small letter e
U+00F0 Lowercase eth	U+0064 Latin small letter d
U+00F8 Lowercase Scandinavian O	U+006F Latin small letter o
U+00FE Lowercase Icelandic thorn	U+0074 Latin small letter t plus U+0068 Latin small letter h
U+0110 Uppercase D with crossbar	U+0044 Latin capital letter D

[11] Ignoring the "<super>" designation that precedes the substitution value in the Unicode character database.
[12] To save processing steps, an implementation may choose to apply the rules given in the remaining steps to the substitution performed in this step. For example, because lowercase characters will eventually be replaced by uppercase characters, an implementation may choose to use uppercase characters where lowercase characters are indicated.

Examples:

UTF-16 value	Substitution
U+0111 Lowercase d with crossbar	U+0044 Latin capital letter D
U+0131 Lowercase Turkish i	U+0049 Latin capital letter I

U+0111 Lowercase d with crossbar	U+0064 Latin small letter d
U+0131 Lowercase Turkish i	U+0069 Latin small letter i
U+0141 Uppercase Polish L	U+004C Latin capital letter L
U+0142 Lowercase Polish l	U+006C Latin small letter l
U+0152 Uppercase digraph OE	U+004F Latin capital letter O plus U+0045 Latin capital letter E
U+0153 Lowercase digraph oe	U+006F Latin small letter o plus U+0065 Latin small letter e
U+02BB Ayn	Remove
U+02BC Alif	Remove
U+2113 Script small l	U+006C Latin small letter l

7) Apply additional transformations based on the Unicode character category code:

- Class Cc (control characters): remove
- Class Cf (formatting characters): remove
- Class Co (private-use characters): remove
- Class Cs (surrogate characters): remove
- Class Ll (lowercase alphabetic character): Replace the code point with the code point identified in position 13 of the code point 's entry in the Unicode character database if position 13 is not null; if position 13 in the Unicode database is null, retain the original code point
- Class Lm (modifier letters): remove
- Class Lo (other letters): retain
- Class Lt (title-case letters): code points in this category are replaced in step 6
- Class Lu (uppercase alphabetic characters): retain as given
- Class Mc (spacing combining marks): remove
- Class Me (enclosing marks): remove
- Class Mn (non-spacing combining marks): remove
- Class Nd (decimal numeral): Replace with the value indicated by position 8 of the code point 's entry in the Unicode database if that position is not null and if the value in that position is less than 10; if that position in the Unicode database is null or has some other value, retain the code point
- Class Nl (number letters): retain
- Class No (other numbers): retain
- Class Pc (connector punctuation): replace with space[13]
- Class Pd (dash punctuation): replace with space
- Class Pe (close punctuation): Take action as indicated in the following table.

[13] The space character is code point U+0020. Note that here and elsewhere, spaces created at the beginning and ending of a subfield by replacing a code point with a space should be omitted.

UTF-16 value	Substitution
U+005D Right square bracket	Remove
All other characters in this category	Replace with space

- Class Pf (final quote punctuation): replace with space
- Class Pi (initial quote punctuation): replace with space
- Class Po (other punctuation): Take action as indicated in the following table.

UTF-16 value	Substitution
U+0023 Number sign	Retain
U+0026 Ampersand	Retain
U+002C Comma	Retain the first comma internal to subfield $a; replace all other commas with spaces
U+0040 Commercial at	Retain
All other characters in this category	Replace with space

- Class Ps (open punctuation): replace with space

UTF-16 value	Substitution
U+005B Left square bracket	Remove
All other characters in this category	Replace with space

- Class Sc (currency symbol): retain
- Class Sk (modifier symbol): replace with space
- Class Sm (miscellaneous symbols): Take action as indicated in the following table.

UTF-16 value	Substitution
U+002B Plus	Retain
U+266F Music sharp sign	Retain
All other characters in this category	Replace with space

- Class So (other symbol): Take action as indicated in the following table.

UTF-16 value	Substitution
U+266D Music flat sign	Retain
All other characters in this category	Replace with space

- Class Zl (line separator): replace with space
- Class Zp (paragraph separator): replace with space

- Class Zs (space separator): replace with space

8) If through the application of rules 5-7 the subfield has been reduced to a null string, restore the text of the subfield as it existed at the end of step 4

9) Reduce multiple adjacent internal spaces to a single space

10) Precede the text of each subfield with the delimiter character (U+001F) and its subfield code

4. Forbidden and required combinations of access points

These rules apply to access fields in authority records that have been subjected to the transformation described in earlier sections. Except as noted, the tags of the fields do not form part of this comparison.[14]

4.1 An established heading (1XX field[15]) **must not** compare the same as any other established heading. There is one exception:

- an established topical heading (150 field) in one authority record may compare the same as an established form/genre heading (155 field) in another authority record

4.2 A see reference tracing (4XX field[16]) **must not** compare the same as any established heading (1XX field) in any record.

4.3 A see reference tracing (4XX field) **must not** compare the same as any see also reference tracing (5XX field) in any record.

4.4 A see reference tracing (4XX field) **must not** compare the same as another see reference tracing in the same record.

4.5 A see reference tracing (4XX field) in one authority **may** compare the same as a see reference tracing in another authority record.

4.6 A see also reference tracing (5XX field) **must** have the same second and third tag characters[17] and compare the same as an established heading in another authority record. Every 5XX field must have a matching 1XX field; but not all 1XX fields will have matching 5XX fields.

---

[14] For example, this means that a personal name heading (100 field) cannot compare the same as a title see also reference tracing (430 field).

[15] Here and throughout this document, reference to *established* authority headings should be taken to mean *1XX fields found in authority records with values a, d, f or g in 008/09 (kind of record code).* 1XX fields found in authority records with values b, c and e in 008/09 should be regarded for the purposes of this document as 4XX fields.

[16] As explained in another footnote, the term *4XX field* includes 1XX fields in authority records with codes c, d and e in 008/09 (Kind of record).

Examples

The following examples illustrate the derivation of a form used for the purposes of comparison from an access field in an authority record.

The double-dagger symbol (‡) represents the delimiter character (U+001F).

Original:	‡aBrades, Susan Ferleger.
Comparison form:	‡aBRADES, SUSAN FERLEGER

Original:	‡aCamacam, Altamirando,‡d1930-
Comparison form:	‡aCAMACAM, ALTAMIRANDO‡d1930

Original:	‡aLabor, Obstetric‡xdrug effects
Comparison form:	‡aLABOR, OBSTETRIC‡xDRUG EFFECTS

Original:	‡aLabor (Obstetrics)‡xComplications
Comparison form:	‡aLABOR OBSTETRICS‡xCOMPLICATIONS

[17] Example: 500 field in one authority record **must** match a 100 field in another authority record. Note that the consideration of tags in this case is based on the tag itself, not the derived tag group.