Local Data Markup Language Tag Specifications

Collations

Summary ## It contains the child element special, which is affected in turn by its child element sortRulesType.

Collation special

Summary ##

special

Summary ##. Special either has an alias or a sortRulesType child element.

```
<!ELEMENT special (alias |(sortRulesType))>
```

sortRulesType

The <sortRulesType> element is used for deciding what rules the collation will use for sorting. It can be set as either “simple”, “icu” or “other”.

```
<!ELEMENT sortRulesType EMPTY>
<!ATTLIST sortRulesType value (simple|icu|other) "simple">
```

The value attribute decides what rules will be used for sorting. This will, in turn, change the special element.

Collation other

Summary ##This is what happens when you put “other” into the sortRulesType field.

special

Summary ##. It is the special that occurs when the value of sortRulesType is “other”. It has the child element base.

```
<!ELEMENT special sortRulesType value = “other” (base)>
```

base

The <base> element contains an alias element that points to another data source that defines a base collation.

```
<!ELEMENT base (alias)>
```

alias

The <alias> element points to another data source that defines a base collation. If present, it indicates that the settings and rules in the collation are modifications applied on *top* of the respective elements in the base collation. That is, any successive settings, where present, override what is in the base by default. Any successive rules are concatenated to the end of the rules in the base.

```
<!ELEMENT alias EMPTY>
<!ATTLIST alias source CDATA #REQUIRED>
```

The source attribute indicates another data source to define a base collation.

Collation icu

Summary ##This is what happens when you put “icu” into the sortRulesType field.

special

Summary ##. It is the special that occurs when the value of sortRulesType is “icu”. It has the child elements settings, suppress_contractions, optimize and rules.

```
<!ELEMENT special sortRulesType value = “icu” (settings,suppress_contractions,optimize,rules)>
```

settings

The <settings> element affects how the collation strings will be sorted. There are nine attributes to <settings>.

```
<!ELEMENT settings EMPTY>
```

```
<!ATTLIST settings alternate (non-ignorable|shifted) #REQUIRED>
```

Sets alternate handling for variable weights. Based on the setting of the variable weighting, collation elements can either be treated as ignorables or not.

```
<!ATTLIST settings normalization (on|off) “on”>
```

If *on*, then the normal UCA algorithm is used. If *off*, then all strings that are in FCD (see http://unicode.org/notes/tn5/) will sort correctly, but others will not necessarily sort correctly. So should only be set *off* if the the strings to be compared are in FCD.

```
<!ATTLIST settings caseLevel (on|off) “on”>
```

If set to *on,* a level consisting only of case characteristics will be inserted in front of tertiary level. To ignore accents but take cases into account, set strength to primary and case level to *on*.

```
<!ATTLIST settings caseFirst (upper|lower|off) “upper”>
```

If set to *upper*, causes upper case to sort before lower case. If set to *lower*, lower case will sort before upper case. Useful for locales that have already supported ordering but require different order of cases. Affects case and tertiary levels.

```
<!ATTLIST settings numeric (on|off) “on”>
```

If set to *on*, any sequence of Decimal Digits is sorted at a primary level with its numeric value. For example, "A-21" < "A-123".

```
<!ATTLIST settings strength (primary|secondary|tertiary|quaternary|identical) #REQUIRED>
```

The number of levels that are to be considered. For comparison, most of the time a three-level strength will need to be used. In some cases, a larger number of levels will be needed, while in others — especially in searching — fewer levels will be desired.

```
<!ATTLIST settings backwards (on|off) “on”>
```

Sets the comparison for the second level to be backwards, in the “French” style.

```
<!ATTLIST settings hiraganaQuaternary (on|off) “off”>
```

Controls special treatment of Hiragana code points on quaternary level. If turned *on*, Hiragana codepoints will get lower values than all the other non-variable code points. The strength must be greater or equal than quaternary if you want this attribute to take effect.

```
<!ATTLIST settings variableTop CDATA #REQUIRED>
```

The parameter value is an encoded Unicode string, with code points in hex, leading zeros removed, and 'u' inserted between successive elements.

Sets the default value for the variable top. All the code points with primary strengths less than variable top will be considered variable, and thus affected by the alternate handling.

suppress_contractions

The <suppress_contractions> element turns off any existing contractions that begin with those characters. It is typically used to turn off the Cyrillic contractions in the UCA, since they are not used in many languages and have a considerable performance penalty.

```
<!ELEMENT suppress_contractions CDATA #IMPLIED>
```

optimize

The <optimize> element is purely for performance. It indicates that those characters are sufficiently common in the target language for the tailoring that their performance should be enhanced.

```
<!ELEMENT optimize CDATA #IMPLIED>
```

rules

The <rules> element specifies collation ordering in terms of actions that cause characters to change their ordering relative to other characters. For more information about rules, see section 5.14 in the ldml database at http://unicode.org/reports/tr35. Some of the rules child elements are specified in the section Common Sorting Elements.

```
<!ELEMENT rules (alias |(reset,(reset | p | pc | s | sc | t | tc | q | qc | i | ic | x)*)) >
```

q

The <q> element changes the sort order on a quaternary (punctuation/space) level. This is the form of sorting where 'a c' could precede 'a-c', etc.

```
<!ELEMENT q (#PCDATA | cp | last_variable)*>
```

qc

This element accomplishes the same function as <q>, but can be used to organize the sort order of multiple characters. For instance, <qc>a-c a;c a,c</qc> would be the same as writing <q>a-c</q> <q>a;c</q> <q>a,c</q>.

```
<!ELEMENT qc (#PCDATA | cp | last_variable)*>
```

i

The element sets the sort order to be on an identical level. If you needed, you could set 'w' to sort on the same level as 'v'.

```
<!ELEMENT i (#PCDATA | cp | last_variable)*>
```

ic

This element accomplishes the same function as , but can be used to organize the sort order of multiple characters. For instance, <ic>VwW</ic> would be the same as writing V w W.

```
<!ELEMENT ic (#PCDATA | cp | last_variable)*>
```

x

The <x> element is used for expansions and giving context to certain characters. It has ten child elements used in defining rules, and two child elements for indicating expansions and context.

```
<!ELEMENT x (context?,( p | pc | s | sc | t | tc | q | qc | i | ic )*, extend?)>
```

extend

If an <extend> element is necessary, it requires the rule to be embedded in an <x> element. The <extend> element makes certain characters sort relative to other characters at an extended distance, which can be useful for organizing contractions.

```
<!ELEMENT extend (#PCDATA)>
```

context

The context before a character can affect how it is ordered. This could be expressed with a combination of contractions and expansions, but is faster using a <context> element. If a <context> element occurs, it must be the first item in the rule, and requires an <x> element.

For example, suppose that "-" is sorted like the previous vowel. Then one could have rules that take "a-", "e-", and so on. However, that means that every time a very common character (a, e, ...) is encountered, a system will slow down as it looks for possible contractions. An alternative is to indicate that when "-" is encountered, and has the context of succeeding an 'a', it sorts like an 'a', and so on.

```
<!ELEMENT context (#PCDATA)>
```

Collation simple

Summary ##This is what happens when you put “simple” into the sortRulesType field.

special

Summary ##. This is the special that occurs when the value of sortRulesType is “simple”. It has the child element rules.

```
<!ELEMENT special sortRulesType value = “simple” (rules)>
```

rules

The <rules> element specifies collation ordering in terms of actions that cause characters to change their ordering relative to other characters. For more information about rules, see section 5.14 in the ldml database at http://unicode.org/reports/tr35. The child elements for rules are specified in the section Common Sorting Elements.

```
<!ELEMENT rules (alias |(reset,(reset | p | pc | s | sc | t | tc)*)) >
```

Common Sorting Elements

reset

The <reset> element will not change the ordering of the letter you use it on, but it will place subsequent characters relative to that letter. For example, if you wanted to set 'b' to come after 'a', you would write <reset>a</reset> b . In this example, 'a' would not change position, but 'b' would now come directly after 'a'.

Resets only need to be at the start of a sequence, to position the characters relative to a certain character. For example, the line <reset>z</reset> a b c d would set 'z' to be succeeded by 'a', 'b', 'c' and 'd'.

```
<!ELEMENT reset (#PCDATA | cp)*>
```

p

The element changes the sort order on a primary (base character) level. This is the basic form of sorting where 'a' precedes 'b', 'f' precedes 'l' and so on.

```
<!ELEMENT p (#PCDATA | cp | last_variable)*>
```

pc

This element accomplishes the same function as , but can be used to organize the sort order of multiple characters. For instance, <pc>hijk</pc> would be the same as writing h i j k .

```
<!ELEMENT pc (#PCDATA | cp | last_variable)*>
```

s

The <s> element changes the sort order on a secondary (accent) level. This is the form of sorting where you can set characters like 'ä' to come after characters like 'a' based on the accent.

```
<!ELEMENT s (#PCDATA | cp | last_variable)*>
```

sc

This element accomplishes the same function as <s>, but can be used to organize the sort order of multiple characters. For instance, <sc>àáâã</sc> would be the same as writing <s>à</s> <s>á</s> <s>â</s> <s>ã</s>.

```
<!ELEMENT sc (#PCDATA | cp | last_variable)*>
```

t

The <t> element changes the sort order on a tertiary (case/variant) level. This is the form of sorting where 'A' would precede 'a' or vice-versa.

```
<!ELEMENT t (#PCDATA | cp | last_variable)*>
```

tc

This element accomplishes the same function as <t>, but can be used to organize the sort order of multiple characters. For instance, <tc>P □</tc> would be the same as writing <t>P</t> <t>□</t> <t>□</t>.

```
<!ELEMENT tc (#PCDATA | cp | last_variable)*>
```

Special

Summary ## This is the ldml special, which contains six unique elements.

abbreviation

An <abbreviation> element is used to abbreviate writing systems. Summary ##

```
<!ELEMENT abbreviation EMPTY>
<!ATTLIST abbreviation value CDATA #IMPLIED>
```

The value attribute is where you can input the abbreviation of a writing system.

defaultFontFamily

A <defaultFontFamily> element is an element to choose a font family. Summary ##

```
<!ELEMENT defaultFontFamily EMPTY>
<!ATTLIST defaultFontFamily value CDATA #IMPLIED>
```

The value attribute is where you can input a font family.

defaultFontSize

A <defaultFontSize> element is used to set a font size for a Summary ##.

```
<!ELEMENT defaultFontSize EMPTY>
<!ATTLIST defaultFontSize value CDATA #REQUIRED>
```

The value attribute is where you can input the font size.

defaultKeyboard

A <defaultKeyboard> element does … Summary ##

```
<!ELEMENT defaultKeyboard EMPTY>
<!ATTLIST defaultKeyboard value CDATA #REQUIRED>
```

The value attribute is good for....

languageName

A <languageName> element is used for Summary ##. Input the actual name of the language, not the iso-code.

```
<!ELEMENT languageName EMPTY>
<!ATTLIST languageName value CDATA #REQUIRED>
```

The value attribute is where you input the name of the language.

spellCheckingId

A <spellCheckingId> element is the spell checker you will use for your writing system. Summary ##

```
<!ELEMENT spellCheckingId EMPTY>
<!ATTLIST spellCheckingId value ID #IMPLIED>
```

The value attribute is the ID of the spell checker you will use. Use a specific ID number.