Apache™ FOP: Complex Scripts

Table of contents

1 Overview

This page describes the complex scripts features of Apache™ FOP, which include:

- Support for languages written with right-to-left scripts, such as Arabic and Hebrew scripts.
- Support for languages written with South Asian and Southeast Asian scripts, such as Devanagari, Khmer, Tamil, Thai, and others.
- Support for advanced substitution, reordering, and positioning of glyphs according to language and script sensitive rules.
- Support for advanced number to string formatting.

2 Disabling complex scripts

Complex script features are enabled by default. If some application of FOP does not require this support, then it can be disabled in three ways:

1. **Command line:** The command line option `-nocs` turns off complex script features: `fop -nocs -fo mydocument.fo -pdf mydocument.pdf`
2. **Embedding:** `userAgent.setComplexScriptFeaturesEnabled(false);`
3. **Optional setting in fop.xconf file:** <fop version="1.0"> <complex-scripts disabled="true"/> ... </fop>

When complex scripts features are enabled, additional information related to bidirectional level resolution, the association between characters and glyphs, and glyph position adjustments are added to the internal, parsed representation of the XSL-FO tree and its corresponding formatted area tree. This additional information will somewhat increase the memory requirements for processing documents that use these features.

> **Note:**
> A document author need not make explicit use of any complex scripts feature in order for this additional information to be created. For example, if the author makes use of a font that contains OpenType GSUB and/or GPOS tables, then those tables will be automatically used unless complex scripts features are disabled.

3 Changes to your XSL-FO input files

In most circumstances, XSL-FO content does not need to change in order to make use of complex scripts features; however, in certain contexts, fully automatic processing is not sufficient. In these cases, an author may make use of the following XSL-FO constructs:

- The `script` property.
- The `language` property.
- The `writing-mode` property.
- The number to string conversion properties: `format`, `grouping-separator`, `grouping-size`, `letter-value`, and `fox:number-conversion-features`.
- The `fo:bidi-override` element.
- Explicit bidirectional control characters: U+200E LRM, U+200F RLM, U+202A LRE, U+202B RLE, U+202C PDF, U+202D LRO, U+202E RLO.
- Explicit join control characters: U+200C ZWNJ and U+200D ZWJ.

4 Authoring Details

The complex scripts related effects of the above enumerated XSL-FO constructs are more fully described in the following sub-sections.

4.1 Script Property

In order to apply font specific complex script features, it is necessary to know the script that applies to the text undergoing layout processing. This script is determined using the following algorithm:

1. If the FO element that governs the text specifies a `script` property and its value is not the empty string or `"auto"`, then that script is used.
2. Otherwise, the dominant script of the text is determined automatically by finding the script whose constituent characters appear most frequently in the text.

In case the automatic algorithm does not produce the desired results, an author may explicitly specify a `script` property with the desired script. If specified, it must be one of the four-letter script code specified in ISO 15924 Code List or in the Extended Script Codes table. Comparison of script codes is performed in a case-insensitive manner, so it does not matter what case is used when specifying these codes in an XSL-FO document.

4.1.1 Standard Script Codes

The following table enumerates the standard ISO 15924 4-letter codes recognized by FOP.

Code	Script
`arab`	Arabic
`beng`	Bengali
`bopo`	Bopomofo
`cyrl`	Cyrillic
`deva`	Devanagari
`ethi`	Ethiopic
`geor`	Georgian
`grek`	Greek
`gujr`	Gujarati
`guru`	Gurmukhi
`hang`	Hangul
`hani`	Han
`hebr`	Hebrew
`hira`	Hiragana

Code	Script
kana	Katakana
knda	Kannada
khmr	Khmer
laoo	Lao
latn	Latin
mlym	Malayalam
mymr	Burmese
mong	Mongolian
orya	Oriya
sinh	Sinhalese
taml	Tamil
telu	Telugu
thai	Thai
tibt	Tibetan
zmth	Math
zsym	Symbol
zyyy	Undetermined
zzzz	Uncoded

4.1.2 Extended Script Codes

The following table enumerates a number of non-standard extended script codes recognized by FOP.

Code	Script	Comments
bng2	Bengali	OpenType Indic Version 2 (May 2008 and following) behavior.
dev2	Devanagari	OpenType Indic Version 2 (May 2008 and following) behavior.
gur2	Gurmukhi	OpenType Indic Version 2 (May 2008 and following) behavior.
gjr2	Gujarati	OpenType Indic Version 2 (May 2008 and following) behavior.
knd2	Kannada	OpenType Indic Version 2 (May 2008 and following) behavior.

Code	Script	Comments
mlm2	Malayalam	OpenType Indic Version 2 (May 2008 and following) behavior.
ory2	Oriya	OpenType Indic Version 2 (May 2008 and following) behavior.
tml2	Tamil	OpenType Indic Version 2 (May 2008 and following) behavior.
tel2	Telugu	OpenType Indic Version 2 (May 2008 and following) behavior.

Warning:

Explicit use of one of the above extended script codes is not portable, and should be limited to use with FOP only.

Note:

When performing automatic script determination, FOP selects the OpenType Indic Version 2 script codes by default. If the author requires Version 1 behavior, then an explicit, non-extension script code should be specified in a governing `script` property.

4.2 Language Property

Certain fonts that support complex script features can make use of language information in order for language specific processing rules to be applied. For example, a font designed for the Arabic script may support typographic variations according to whether the written language is Arabic, Farsi (Persian), Sindhi, Urdu, or another language written with the Arabic script. In order to apply these language specific features, the author may explicitly mark the text with a `language` property.

When specifying the `language` property, the value of the property must be either an ISO639-2 3-letter code or an ISO639-1 2-letter code. Comparison of language codes is performed in a case-insensitive manner, so it does not matter what case is used when specifying these codes in an XSL-FO document.

4.3 Writing Mode Property

The `writing-mode` property is used to determine the axes and direction of the inline progression direction, the block progression direction, the column progression direction (in tables and flows), the shift direction, region placement, the resolution of writing-mode relative property values (such as start, end, before, after), and the default block (paragraph) bidirectionality level.

The `writing-mode` property is inherited, so it can appear on any XSL-FO element type; however, it applies (semantically) only to the following element types:

- `fo:page-sequence`
- `fo:simple-page-master`
- `fo:region-*`
- `fo:block-container`
- `fo:inline-container`
- `fo:table`

If it is not specified on one of these element types, but is specified on an ancestor element, then the value specified on that ancestor element (the inherited value) is used; otherwise, the initial value `lr-tb` is used.

At present, only the following values of the `writing-mode` property are supported:

- `lr-tb`
- `rl-tb`
- `lr`
- `rl`

Writing modes that employ a vertical inline progression direction are not yet supported.

4.4 Bidi Override Element

The `fo:bidi-override` element may be used to override default bidirectional processing behavior, including default embedding levels and default character directionality. In the absence of either this element or use of explicit Bidi Control Characters, the default behavior prescribed by the Unicode Bidirectional Algorithm applies.

4.5 Bidi Control Characters

In addition to the use of the Bidi Override Element, an author may make use of the following explicit Unicode Bidi Control Characters:

- U+200E - LEFT-TO-RIGHT MARK (LRM)
- U+200F - RIGHT-TO-LEFT MARK (RLM)
- U+202A - LEFT-TO-RIGHT EMBEDDING (LRE)
- U+202B - RIGHT-TO-LEFT EMBEDDING (RLE)
- U+202C - POP DIRECTIONAL FORMATTING (PDF)
- U+202D - LEFT-TO-RIGHT OVERRIDE (LRO)
- U+202E - RIGHT-TO-LEFT OVERRIDE (RLO)

If an embedding or override is not terminated (using U+202C PDF) prior to the end of a delimited text range, then it is automatically terminated by FOP.

4.6 Join Control Characters

In order to prevent joining behavior in contexts where joining occurs by default, for example, between U+0628 ARABIC LETTER BEH and U+0646 ARABIC LETTER NOON, an author may used a U+200C ZERO WIDTH NON-JOINER (ZWNJ).

Conversely, in order to force joining behavior in contexts where joining does not occur by default, for example, between U+0628 ARABIC LETTER BEH and U+0020 SPACE, an author may used a U+200D ZERO WIDTH JOINER (ZWJ).

The behavior of ZWNJ and ZWJ is script specific. See The Unicode Standard, Chapter 8, Middle Eastern Scripts for information on the use of these control characters with the Arabic script. See The Unicode Standard, Chapter 9, South Asian Scripts - I for information on the use of these control characters with common Indic scripts.

5 Supported Scripts

Support for specific complex scripts is enumerated in the following table. Support for those marked as not being supported is expected to be added in future revisions.

Script	Support	Tested	Comments
Arabic	full	full	
Bengali	none	none	
Burmese	none	none	
Devanagari	partial	partial	join controls (ZWJ, ZWNJ) not yet supported
Khmer	none	none	
Gujarati	partial	none	pre-alpha
Gurmukhi	partial	none	pre-alpha
Hebrew	full	partial	
Kannada	none	none	
Lao	none	none	
Malayalam	none	none	
Mongolian	none	none	
Oriya	none	none	
Tamil	none	none	
Telugu	none	none	
Tibetan	none	none	
Thai	none	none	

6 Supported Fonts

Support for specific fonts is enumerated in the following sub-sections. If a given font is not listed, then it has not been tested with these complex scripts features.

6.1 Arabic Fonts

Font	Version	Glyphs	Comments
Arial Unicode MS	1.01	50377	limited GPOS support
Lateef	1.0	1147	language features for Kurdish (KUR), Sindhi (SND), Urdu (URD)

Font	Version	Glyphs	Comments
Scheherazade	1.0	1197	language features for Kurdish (KUR), Sindhi (SND), Urdu (URD)
Simplified Arabic	1.01		contains invalid, out of order coverage table entries
Simplified Arabic	5.00	414	lacks GPOS support
Simplified Arabic	5.92	473	includes GPOS for advanced position adjustment
Traditional Arabic	1.01	530	lacks GPOS support
Traditional Arabic	5.00	530	lacks GPOS support
Traditional Arabic	5.92	589	includes GPOS for advanced position adjustment

6.2 Devanagari Fonts

Font	Version	Glyphs	Comments
Aparajita	1.00	706	
Kokila	1.00	706	
Mangal	5.01	885	designed for use in user interfaces
Utsaah	1.00	706	

7 Other Limitations

Complex scripts support in Apache FOP is relatively new, so there are certain limitations. Please help us identify and close any gaps.

- Only the PDF output format fully supports complex scripts features at the present time.
- Shaping context does not extend across an element boundary. This limitation prevents the use of `fo:character`, `fo:inline` or `fo:wrapper` in order to colorize individual Arabic letters without affecting shaping behavior across the element boundary.

8 Related Links

In addition to the XSL-FO specification, a number of external resources provide guidance about authoring documents that employ complex scripts and the features described above:

- The Unicode Standard
- Unicode Bidirectional Algorithm
- OpenType Advanced Typographic Extensions