Configuring and Extending the PDF-D4P Transformation Type

W. Eliot Kimber

Overview of the PDF-D4P transformation type

W. Eliot Kimber

Contents

Chapter 1

Overview of the PDF-D4P transformation type

The PDF-D4P transformation type extends the base PDF transformation type in order to make configuration of the overall publication structure easier and more flexible.

As of version 1.5.4M3 of the Open Toolkit, the PDF transformation type has a number of limitations that make it harder than it should be to do things like generate covers and customize the details of page geometry, the order of components, especially generated components like the ToC, and page breaks before specific topics or topic types. It also provides no easy way to control the numbering of topic headings.

The PDF-D4P transformation type is an experimental refactoring of the base PDF transformation type with the goal of making it easier to customize the overall look and feel of a publication.

The transformation type provides the following new features:

- Enables easy creation of front and back covers.
- Makes it easier to add new page masters and page sequence masters.
- Makes it easier to control how specific topic types create page breaks.
- Makes it possible to have multiple top-level topics in a single page sequence.
- Makes it possible to customize the details of page sequence generation.
- Enables (but doesn't implement) the ability to have a single topic result in multiple page sequences.
- Addresses a number of other convenience factors or missing features in the base PDF transform.
- Provides an extension point for numbering topics ("enumeration") and provides built-in options for "military" and "outline" style numbering.

The PDF-D4P transformation type is implemented as a separate transformation type that overrides and extends the base PDF transformation type. You can use the same customization mechanism and global extension points for customizing the PDF-D4P transformation type as you use for the base PDF transformation type. In general, the PDF-D4P transformation type retains the original mechanisms (attribute sets, named templates, modes, etc.) except where needed to implement new features. This means that many customizations of the base PDF transforms should continue to work, especially those that only modify attribute sets or add overrides for existing templates not involved with handling of top-level topics. Customizations that override templates from `root-processing.xsl`, `commons.xsl`, or `layout-masters.xsl` will probably need to be reworked (and should become simpler or unneeded in many cases).

Chapter 2

Extending the PDF-D4P transformation type

Topics:

The PDF-D4P transformation type implements a completely different processing model for constructing page masters, page sequence masters, and page sequences.

Table 1: Extension Point Quick Reference on page 3 lists the aspects of the publication whose generation you can customize and the associated XSLT modes, named templates, and attribute sets. In general, where there is a mode, you can implement templates in that mode to override or extend the default behavior. When there is a named template you can override the named template to change the default behavior. When there is an attribute set you can override the attribute set to configure specific properties of the result XSL-FO elements.

Table 1: Extension Point Quick Reference

Publication Aspect	Modes	Named Templates	Attribute Sets
Page masters	generateLayoutMasters	createDefaultLayoutMasters, createLayoutMasterSet	
Page sequence masters	createPageSequenceMasters	createDefaultPageSequenceMasters	
Page sequences	constructNavTreePageSequences, constructPageSequence, getPublicationRegion, getTopLevelTopics, setPageNumberFormat, setInitialPageNumber, constructStaticContent	constructNavTreePageSequences, insertBodyStaticContents	
Page breaks for topics	setTopicPageBreak	setTopicPageBreak	topic-first-block-topLevel. topic-first-block-*topicType*

Publication Aspect	Modes	Named Templates	Attribute Sets
Front cover	getFrontCoverTopics, getFrontCoverGraphicUri	createCoversAndInitialPages, createFrontCover	region-body.front-cover, front-cover-graphic
Back cover	getBackCoverTopics, getBackCoverGraphicUri	createLastPages, createBackCover	back-cover-graphic
Bookmark list	bookmark		
Topic title numbering	enumeration, getTitle	getTitle	

The files for the PDF-D4P plugin mirror the filenames and organization of the base PDF transform for the most part, so if you are familiar with the base PDF transform you should be able to quickly find any overrides or extensions for the PDF-D4P plugin. Where the names do not match, the PDF-D4P filenames reflect the mode or general processing concern the files implement.

A key change is an abstraction and extension of the general notion of "topicref type". The library function dita-ot-pdf:determineTopicType() takes a topic and returns its abstract topic type, e.g. "topicChapter", "topicAbstract". Most of the business logic that maps specific topics to specific output results is based on topic type, not the actual root tagname or DITA class. This abstraction allows you to use any rule at all to determine that a given topic is, for example, a chapter or an appendix. Usually this is based on the structure of the map, as in the case of Bookmap, where topicrefs of `<part>` and `<chapter>` determine the topic types of the topics they point to.

2.1. Customizing Page Layout and Page Sequences

You can add new page masters and page sequences to the base set. You can also configure how topics are organized into page sequences. You can also control how different topic types generate page breaks.

The PDF-D4F transformation type defines a base set of page masters and corresponding page sequence masters. You can add additional page masters and page sequence masters by implementing templates in the modes generateLayoutMasters and createPageSequenceMasters, normally by matching on "/". If you need to override the details of the default page masters or page sequence masters you can override the templates createDefaultLayoutMasters or createDefaultPageSequenceMasters respectively.

The details of the built-in page masters are defined by the attributes sets in `cfg/fo/attrs/layout-masters-attr.xsl`, in both the PDF-D4P plugin and the base PDF plugin.

2.1.1. Overview of XSL-FO Page Masters, Page Sequence Masters, and Page Sequences

An XSL-FO document consists of two main parts: the layout master set and one or more page sequences. The layout master set defines all the page masters and the page sequence masters that use the page masters. The page sequences contain the content to be paginated.

The page masters determine the geometry and regions of each page (headers, footers, side regions, and body regions). The page sequence masters combine page masters into sequences based on page position within the sequence: first, not first, even, odd, last, etc. You control the details of page layout through the definition of page masters and page sequences. Overall page geometry is controlled through attribute sets that set the page size and margins. However, you may need to create more sophisticate page masters in order get specific effects, such as overlapping regions or something. In most cases the default page sequence masters are sufficient. Any new page masters you create must have a corresponding page sequence master in order to be useable from page sequences.

Page sequences contain the actual content to be flowed. A page sequence is associated with a single page sequence master. The page sequence also defines how the pages within it will be numbered and whether or not it must start on an odd or even page. Each page sequence must start a new page.

If you want pages with different page masters from preceding pages you must start a new page sequence—there is no way in XSL-FO to have different page masters assigned dynamically to arbitrary parts of the flow, so there's no way to say something like "if topic type X occurs within the flow, use page master Y". So you will need to create new page sequences for topic types that need a different page geometry than the pages that precede it (other than first/not-first pages, which is handled by the page sequence master).

If you want different page numbering in different parts of the publication, for example, folio-by-chapter, then you must start new page sequences in order to reset the page numbering at 1 (and, with XSL-FO 1.1, define the folio prefix or suffix if that feature is supported by your FO engine).

Note that you **do not** need distinct page masters or page sequences to get different effects within the body region of a page. For example, you can create unique chapter opener effects using fo:block or fo:block-container without the need for a specific page master for the first page as long as the page master otherwise provides an appropriately-sized body region. You also do not need new page sequences simply to cause page breaks: any fo:block can break to a new page, a new even page, or a new odd page.

2.1.2. PDF-D4P page sequence construction process

The construction of page sequences from the merged topic document involves the following main steps:

1. The named template createCoversAndInitialPages constructs any pages that should be generated before the first content topic, e.g., front cover, inside front cover, title page, copyright page, etc.
2. The named template constructNavTreePageSequences manages the generation of page sequences from the map's navigation tree (meaning the topicrefs that are not resource-only topicrefs).

3. The named template createIndex manages the generation of any index pages (if the index was not generated by an explicit topicref in the navigation map).
4. The named template createLastPages manages the generation of any pages that follow the navigation tree pages, such as a canned colophon or back cover.

The named template constructNavTreePageSequences applies templates in the mode constructNavTreePageSequences. The default template for "/" in that mode then does the actual work of constructing the page sequences. This lets you override the default processing by providing a template that matches "/", although that shouldn't normally be necessary unless you want to change things in a radical way.

The default page sequence construction process proceeds as follows:

1. The mode getTopLevelTopics is used to construct a list of the topics or generated list placeholders that are top-level topics, meaning they are candidates for starting new page sequences. Note that being a top-level topic does not *require* a topic or topicref to start a new page sequence, it only gives it the opportunity to do so. You can have a single page sequence contain multiple top-level topics. By default, any topic whose topicref has no parent topicref or topichead is a top-level topic. Any topics identified as front or back cover topics are excluded from this set (see the discussion of cover generation).
2. The top-level topics are grouped adjacently by publication region, using the dita-ot-pdf:getPublicationRegion() function.

   Publication regions correspond to things like frontmatter, body, appendices, backmatter, etc. You can define your own publication regions and associate them by implementing templates in the mode getPublicationRegion. The function dita-ot-pdf:getPublicationRegion() returns the string name of the publication region the topic is assigned to. All the adjacent elements with the same publication region will go in a single page sequence.

   The result of this phase is a sequence of `<dita-ot:pageSequence>` elements, one for each distinct publication region group. The `<dita-ot:pageSequence>` elements have a `@pubRegion` attribute that specifies the publication region for that page sequence. The elements are "page sequence generators" as they directly drive the creation of the actual XSL-FO page sequence elements in the FO output.
3. The sequence of `<dita-ot:pageSequence>` elements is processed in the constructPageSequence and the corresponding set of top-level topics is passed as a parameter for use by the handling templates. Templates in this mode use templates of the form:

```
  <xsl:template mode="constructPageSequence" match="dita-
ot:pageSequence[@pubRegion = 'frontmatter']" priority="10">
    <xsl:call-template name="doPageSequenceConstruction">
      <xsl:with-param name="pageSequenceMasterName" select="'front-matter-
sequence'"/>
    </xsl:call-template>
  </xsl:template>
```

   By default, otherwise unhandled publication regions use the body-sequence page sequence master. Any publication region name starting with "body" will be matched by the default "body" page sequence constructor.

   To map custom publication regions to XSL-FO page sequences or to override the default behavior, implement templates in the constructPageSequence mode.

   The named template doPageSequenceConstruction processes the `<dita-ot:pageSequence>` element and takes as a parameter the name of the page sequence master to use.

   The doPageSequenceConstruction then calls apply-templates on the `<dita-ot:pageSequence>` element in the following modes:

setInitialPageNumber	Handles generation of the `@initial-page-number` attribute for the page sequence. By default, the first page sequence of the body sets the initial page number to "1", resetting page numbering for the body. You can override this mode if you need to change how page numbering is handled.

constructStaticContent	Handles generation of the static content for the page sequence. You can override this mode to change the set of static regions generated for a given page sequence. See *Defining Headers and Footers for Page Sequences (Static Content)* on page 7 for more information.

The doPageSequenceConstruction template then applies templates to all the top-level topics within the page sequence constructor in the mode processTopLevelTopic.

If you map multiple topics to single page sequences, you may need to adjust the page break behavior for the topics.

Given the above process, your configuration task depends on what you need to do:

- If you are using Bookmap and just want to control page break behavior, you can do that by overriding the code that sets the break-before property for the first block in a topic. See [XREF to topic on controlling topic page breaks].
- If you are implementing a custom map type or a generic map that needs to handle specialized topic types, but you want to use the built-in page sequences, then you only need to extend the topic-to-topic-type mapping by implementing templates in the mapTopicsToType mode and map your custom elements to the built-in topic types.
- If you need to implement new or different publication regions then you need to implement templates in the mode getPublicationRegion. If those publication regions need to map to specific page sequence masters or otherwise have customized page sequences, you must also implement templates in the mode constructPageSequence.
- If you want to modify the page numbering behavior, override templates in the mode setInitialPageNumber.
- If you want to modify the set of static content containers defined for a page sequence, override templates in the mode constructStaticContent.
- If you want to modify the static content itself, override the named templates defined in the base file `static-content.xsl` (see *Defining Headers and Footers for Page Sequences (Static Content)* on page 7).

2.1.3. Defining Headers and Footers for Page Sequences (Static Content)

Headers and footers (and side regions if you need them) are constructed by using a combination of static text, page number references, and marker references within `<fo:static-content>` elements within `<fo:page-sequence>`. Within the body flow, you then set markers as needed, typically with the title of each chapter or first-level heading or whatever.

The base file `static-content.xsl` defines a set of named templates that construct the static content for different publication regions (body, frontmatter, etc.). You can modify the header and footer details by overriding whichever of these templates constructs the part you need. You may also need to override the templates that set markers within the FO for each topic that needs to contribute to the static content.

The `<fo:static-content>` elements for each page sequence are generated by templates applied to the `<dita-ot:pageSequence>` elements in the mode constructStaticContent. You can implement templates in this mode in order to customize the static content details for any page sequence.

2.2. Controlling Topic Page Breaks

When a topic is not the first topic in a page sequence it can start a new page by using the `@break-before` property on its first block. The PDF-D4P transform provides several ways to control this behavior.

Overall topic handling is managed by the named template processTopLevelTopic, which is called for topic within a page sequence.

2.3. Controlling Topic Numbering

The mode enumeration manages generating numbers for use in topic titles.

The named template getTitle is used to get the title text for a topic. The base template in this mode then calls templates in the mode enumeration, which may generate numbers for the topic titles. All contexts in which you need a topic title should go through the getTitle template so that the titles are constructed consistently. This is true for topics in the main flow as well as for bookmap list construction and Contents page construction.

The base PDF processing automatically numbers topics of type "topicPart", "topicChapter", and "topicAppendix" and that processing is not modified.

Processing of topics within parts, chapters, and appendixes is handled by the default enumeration templates for topics that are not within frontmatter or backmatter publication regions when numbering is turned on.

You can turn numbering of non-top-level topics on by setting the global XSLT variable *d4pTopicEnumerationStyle* to one of the values "military" or "outline". Military numbering uses the format pattern 1.1.1.1 and outline numbering uses the pattern I.A.1.a.i.1.

You can implement your own template in the enumeration mode to implement different numbering behavior. See the templates in `enumeration.xsl` for sample code.