Transformation and formatting specification guidelines

$Date: 2004/02/25 16:04:11 $(UTC)

Table of Contents

1. Introduction

This document overviews basic requirements in formatting specifications for customers contracting the writing of XSLT and XSL-FO stylesheets from Crane Softwrights Ltd. Crane's time will be used most effectively (thus reducing chargeable project time) if customers can satisfy these requirements when writing the formatting specifications describing their expectations.

The basic principle is the need for detail. Writing a stylesheet is a straightforward and methodical, but only when all of the details desired or required from the customer are known in advance. Rework and speculation is costly and time consuming. Our objective is to understand all of your formatting requirements in order for us to supply a comprehensive stylesheet meeting your needs that you can efficiently and effectively support on your own without our subsequent intervention.

The approach taken in these guidelines is to enumerate many areas where choices are available to help the customer consider situations that need to be accommodated but might not be readily apparent in the work done by the customer to date. Obvious requirements are stated explicitly to ensure completeness, not in any way to imply that our customers are not aware of the obvious.

The choices made in terminology and the values requested of the customer are such that the customer can utilize their formatting specification for long-term support of their own stylesheets. Most of the properties requested map directly to XSL-FO concepts and constructs, thus promoting easier support of the stylesheets written to meet the specifications. In many ways these guidelines attempt to teach the reader of the level of detail required when establishing the nuances of format available when using XSL-FO.

1.1. Copyright and conventions

These guidelines are considered part of Crane's competitive advantage, and as such, are considered proprietary and private. Please respect our copyright and do not share these guidelines with others. Some of the tools used here may already be made publicly available on our web site, or may be so in the future, but those are isolated revelations of our approaches and we do not have plans to make all the material in or referenced by this document public knowledge.

The drawing conventions in these materials include using gray-tone mockups of the information on the page and dark black labeled dimensions and highlights.

This is a work in progress and can only be improved with feedback from our customers regarding the clarity of our guidelines and the efficacy they have in helping our customers understand their own formatting requirements. We look forward to any feedback you have regarding improving this living document.

Colophon: these guidelines were written using the DocBook document model and off-the-shelf DocBook XSL stylesheets. Unfortunately, it would seem the PDF stylesheets are less than ideal for this task, but I thought the convenience of having a PDF rendition would help.

2. Test and acceptance comparison files

2.1. Sample scenarios and source data

Regardless of the output requirements, the more sets of organized sample source data the customer can supply for evaluation and testing, the better. Please document the distinctions between sample source data as identifiable scenarios that can be used for development and acceptance testing.

By "organized", we mean to say that a large haphazard collection of data files we are unable to navigate or make sense of is not a useful contribution to the project. More data reflecting varying requirements is far more helpful than repetitive data not bringing any new insights into the requirements.

Stylesheets are often aggregators that pull in information from multiple sources to produce one or more final results. The customer must make some important decisions regarding the arrangement of files in their file system, and document these decisions sufficiently well for Crane to mimic required subdirectory naming conventions and hierarchies. Many stylesheets need to hardwire relative directory paths between multiple source files and the stylesheet fragments.

Consider the utility of creating subdirectories for commonly-shared but semantically distinct data files that are referenced by the unique data files being processed, and for the development and test of stylesheets:

- `subdir/` - contains all test and stylesheet files
- `subdir/common1/*.xml` - contains common shared set one of XML files

- `subdir/common2/*.xml` - contains common shared set two of XML files
- `subdir/data/*.xml` - contains unique data files in support of documented scenarios utilized for development, debug, test and acceptance, as well as possibly any required invocation files used to produce outputs
- `subdir/ss/*.xsl` - contains Crane's development and delivered works in support of the project, and may include invocation files in place of those found in the data directory
- `subdir/key/*.xsl` - contains Crane's documentary key reference stylesheets (documented below)
- `subdir/out/*.*` - contains output files created in support of the scenarios
- `subdir/work/*.*` - contains intermediate transient work files not to be preserved after invocation of the stylesheets

Mimicking the production environment as closely as possible will reduce the number of modifications after delivery, but when production files are widely dispersed, the above practice of gathering all inputs close together will promote ease of debug, testing and acceptance. Please document any differences anticipated between your expected production environment and the above development environment.

Please arrange, document and supply various stand-alone scenarios of source data files and ensure that all files for the scenarios are included for the stylesheet to successfully act on the complete information. Where different subdirectories are required, using a ZIP file format with embedded directory names will allow us to quickly arrange the data files in our mimicked environment. Any document model descriptions for source files will help us better understand the nature of the information we are working with. Please remember that even if you are intimately familiar youre your data after decades of a close and personal relationship with it, it will be new to us and we may not be aware of something that is obvious to you.

Sample data files will form the basis for evaluation and acceptance. Please identify which scenarios (if only a subset and not the complete set of scenarios) will satisfy your acceptance test criteria in order to judge any development a success.

2.2. Sample scenario results

When producing non-formatted results, a document model of the result is very helpful in order to measure that the requirements for transformation have been met. We can validate the grammar of our results using any document model you supply using XML 1.0 Document Type Definition (DTD) expressions, W3C Schema expressions, RELAX-NG expressions and we can use Schematron assertions you supply for any required content constraints.

Prototypical formatted results are essential in establishing your own expectations for formatting as well as giving us an effective picture of our objectives in meeting your needs. These mockups need not have bona-fide data, though it would help tremendously if the mockups reflected the test data of the scenarios. If you are working from a legacy application, then samples of legacy outputs would be helpful in addition to the mockups you prepare for the new stylesheet environment. Please make a conscious assessment of the differences between your legacy and stylesheet requirements.

While PDF layouts can be useful, word processor-based layouts can capture many more nuances of layouts in the properties of the revisable format. We can accept mockups in Open Office format and in Microsoft Word. Even though we can look into the revisable form and establish the various settings for the items listed below, it is easy for customers to overlook nuances of detail in their mockups. If all we have to work with are the mockups, time may be unnecessarily occupied trying to accommodate differences that were never meant to be in the mockup.

Note that it is acceptable to document which colors or decorations (patterned underlines, background colors, text colors, etc.) in mockups might be reserved for documentary purposes and to then take advantage of utilizing different appearances throughout the mockup to reflect the documented differences or other details. Absent any such description, we

can only assume that colors and decorations used in the scenario result are significant and efforts will be made to mimic them all accordingly.

An example of an effective use of color decoration is to use pastel shades as backgrounds to black text, much as in the appearance created when using highlighter markers. Especially important is distinguishing which content in the result is generated content (static or derived from and by the stylesheet) and which content in the result is copied from the source information. When aggregating information from multiple sources, different shades or decorations could be used to distinguish the different sources. Predominantly, most reports are black letters on white background and this needn't change for the bulk of your mockup, but where there is a significant difference in the source of information (e.g. from external files) using the light blue, pink or yellow backgrounds as a highlighter will help bring out any distinction.

It is acceptable to adopt an annotation convention such as brief sequences of red-letter text in square brackets as "instructions to the reader", though if this is very long and might disturb the spacing in the document, please see below regarding annotation citations.

3. Key references and annotations

An effective set of tools in documenting formatting specifications is the use of key references and annotations. References document the source of information from the source node tree, while annotations document supplemental prose you write elsewhere. Annotations are only really necessary when you cannot fit your comments into the white-space or margin areas on your mockups.

A key reference reads as in these examples: "!23!", "!127!", "!127.3!" such that the numbers point to XPath expressions documented in a key report. Whole numbers point to elements. Fractional numbers point to attributes documented for the element numbered by the whole part. Tools documented below can be used to create key reports from sample instances.

An annotation reads as in these examples: "!A!", "!B1!", "!B2!" such that the letter/number combinations make reference to another document you maintain with the prose associated with an annotation. This is a low-tech approach to saving real estate on your mockup pages, and has an appearance not unlike "callouts". No tools are provided for such annotations, this is merely a convention that is consistent with the key references for use on pages. The only caveat is to uniquely identify the collection of annotations and then ensure the document with the annotations correctly refers to the unique identification. Date and time stamps should be sufficient for uniqueness.

3.1. Creating key files

Three XSLT stylesheets work together to create key reference files and associated reports. A key reference file is a transliterated sample data file, with the data values replaced with key references.

The extent of coverage of a key reference file is limited to the extent to which a sample file covers the possible input combinations. For this reason, it is recommended that a contrived data example be created in which all data elements are found. This contrived example would be included along with a number of bona-fide examples representative of real-world use of your information. Note that it is not necessary for the contrived example to be validated against any kind of document model, as often it is not likely that such a contrivance would meet all of the expressed constraints. Stylesheet processors accept well-formed XML without need for validation.

Utilizing this contrived data sample in particular, and other bona-fide samples if also available, key reference files are created using the `keyxml.xsl` XSLT stylesheet. Reports of the key reference information are created using the `key2html.xsl` and `key2text.xsl` XSLT stylesheets.

3.1.1. `keyxml.xsl`

This XSLT stylesheet reads a data file and outputs all non-white-space PCDATA and all attribute values as key references. Elements are counted from 1 in document order throughout the document. Attributes are counted in the order utilized by the XSLT processor and such counting may differ between different implementations of XSLT processors.

XPath 1.0 addresses are recorded in each element's start or empty tag. By default sibling elements are distinguished using numeric predicates in their respective XPath addresses.

A single recommended parameter should be specified during invocation, though it is not mandatory:

- `label=some-text-to-identify-the-file-and-date`
    - this will record inside the file some information about the file for the purposes of reporting
    - to avoid ambiguous reporting of differing file contents and file revisions, the text used here should include the file name and either the date and time stamp of when the report file was created or (preferred) the date and time stamp of the file being analyzed

A single optional parameter may be specified during invocation:

- `repeat=yes` (or any non-empty-string value)
    - this will preserve the distinctions in XPath addresses between repeated sibling elements of the same name
    - use this parameter when wanting to expose all XPath sibling relationships and not just unique hierarchy

3.1.2. `key2text.xsl` and `key2html.xsl`

This XSLT stylesheet reads a key reference file and outputs the XPath information, and optionally any sample values, found embedded in the document through the use of `key2xml.xsl`.

Two optional parameters may be specified during invocation:

- `repeat=yes` (or any non-empty-string value)
    - this will expose the distinctions in XPath addresses between sibling elements of the same name
- `values=yes` (or any non-empty-string value)
    - this will expose any sample values recorded during the creation of the key reference file

The `key2text.xsl` output is a simple file of text lines

The `key2html.xsl` output is an HTML file with embedded clipboard logic behind the XPath address values. The intervention of the operator clicking on an XPath address will copy the XPath address to the system clipboard in a format suitable for pasting into another document. This alleviates much of the repetitive typing otherwise necessary when recording lengthy XPath addresses in other documents.

3.1.3. Sample invocation of key reference stylesheets

Consider the following `key.bat` invocation shell file for a Windows command-line processor, utilizing the Saxon brand of XSLT processor:

```
saxon -o %1-key.xml %1.xml ..\key\keyxml.xsl "label=%1.xml %date% %time%"
saxon -o %1-key.htm %1-key.xml ..\key\key2html.xsl
saxon -o %1-key.txt %1-key.xml ..\key\key2text.xsl
```

When in the data subdirectory, one would invoke the above using the following for a test file named `test1.xml`:

```
..\key\key test1
```

The three files produced would be the key reference file `test1-key.xml` and the two report files `test1-key.text` and `test1-key.html`.

4. Generated and sourced content

It is essential to distinguish an sample outputs (be they output structures for simple XSLT transformation or output layouts when using XSL-FO) which content is generated or supplied by the stylesheets and which content is sourced from the input XML documents being processed by the stylesheets.

One technique for distinguishing generated and source content in mockup examples is to use color or some other decoration (e.g. italic, underscore, etc) provided you establish your conventions so that the differences in appearance are not interpreted as instructions for formatting.

4.1. Generated content

Generated content is calculated by and supplied by the stylesheet either as fixed boilerplate or conditional generated content based on the presence or absence of data.

Fixed boilerplate might be the document heading or fixed strings used to indicate headings triggered by the presence of a particular section element in the source. At times boilerplate is quite structured itself, with different components of the boilerplate emphasized or displayed quite differently (e.g. headings with two components, the main component bold and underline and a contiguous but less emphasized component trailing behind the heading).

Generated content might be a lengthy checklist of items, some of which might be checked by the presence of source content, but all needing to be listed for completeness for the reader. This kind of generated content is often found in supplemental source files where enumerations of available values are stored. It is important to have complete examples of the enumerations and to detail where the content is to be located and how it is to be processed.

4.2. Sourced content

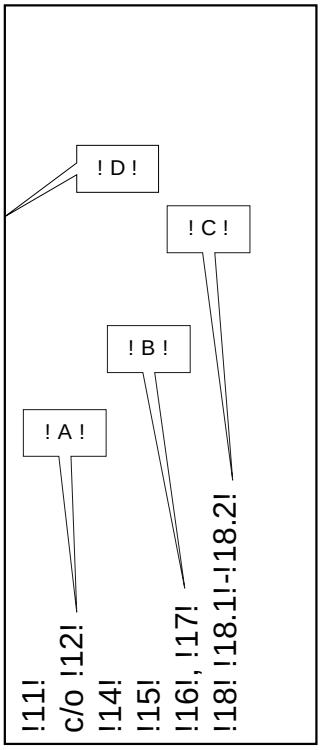
Sourced content is information copied directly from elements or attributes of the source input files and added to the result either a verbatim or manipulated fashion.

The locations of sourced content should be recorded in two places in two different ways to help ensure longevity and accuracy in the formatting specifications. Both involve using the key reference file and reports for a comprehensive test input file, documenting the XPath addresses of the information found in the source file.

In the actual mockups and example outputs, use the key reference number as a succinct indication of the XPath address. Optionally, in a supplemental document, record in some kind of prose a reference to where you are in the output, and the full XPath address of the information being retrieved. This supplemental record will be useful should the sample data change, thus skewing the position-sensitive key reference numbers. Full XPath addresses are more resilient to change than key reference numbers, but are often far too long to include in mockups.

4.3. Example documentation

Consider the following example mailing label that is part of a formatted result. It combines both generated and sourced information, with necessary instructions regarding the generated content.



The associated instructions in prose might read as follows:

- the mailing label is comprised of the bill-to information as follows:
  - `/invoice/bill-to/name`
  - `/invoice/bill-to/contact`
  - `/invoice/bill-to/addressline1`
  - `/invoice/bill-to/addressline2`
  - `/invoice/bill-to/city`
  - `/invoice/bill-to/stateprov`
  - `/invoice/bill-to/country`
  - `/invoice/bill-to/country/@postalcode`
  - `/invoice/bill-to/country/@postalsubcode`
- the generated content instructions are as follows:
  - `A` - when the contact is present, prefix the contact with the sequence "c/o "; if absent present nothing on the line but leave the line blank should the reader wish to annotate the label and return it
  - `B` - always follow the city with the sequence ", " and when the province/state is present, display it, otherwise put all of the country line information on the same line as the city line following the comma
  - `C` - when the postal sub-code is present, separate it from the postal code with a hyphen

- `D` - to meet up with the envelope window, the mailing label is 5cm by 1.2cm in size, positioned 1cm from the top of the page and 1cm from the right of the page, on the first page only; note that margins for the entire first section of content shall be reduced by .5cm; if the first section of content is shorter than the mailing label, the second section of content is started below the mailing label with the full column width as margins

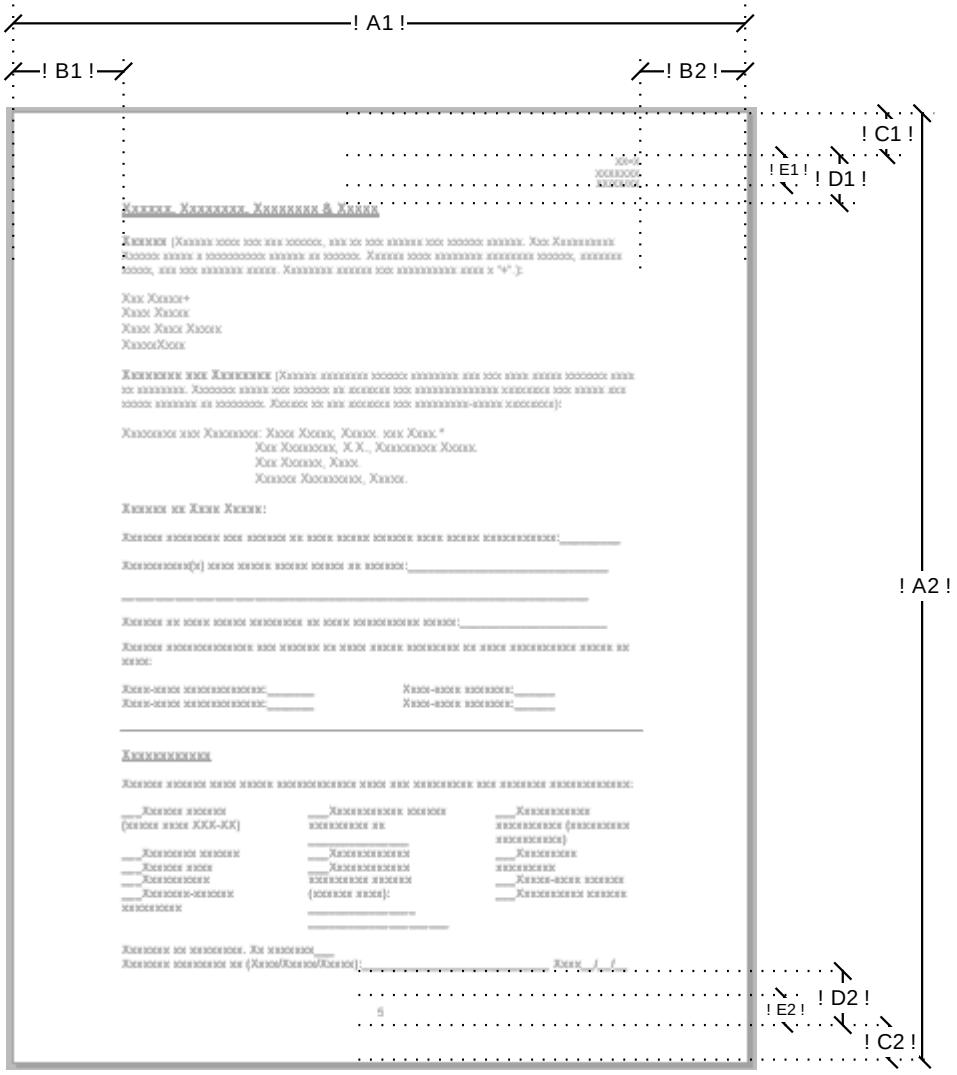
5. Page composition requirements

The following list of properties is not comprehensive or exclusive. Any differences between pages or information on pages needs to be described, and these are only guidelines of things that may differ, and they attempt to bring to light properties of the layout that the reader may be unaware of.

Please do not hesitate to indicate any properties not listed in this section, or ask for clarification on the use of any of the values.

5.1. Pages and geometry

All possible layout geometries for the rendered pages need to be enumerated and given semantic labels. Determine the number of unique combinations of margins and running headers.



5.1.1. Page dimensions

- `A1` - page width
- `A2` - page height
- are there multiple page sizes used in the document?
- any double-sized "right hand folds"?

5.1.2. Horizontal page margins

- `B1` - left margin
- `B2` - right margin
- are there differences between odd and even pages, perhaps due to the binding?
- do you need a mirror layout specifying margins on the inside edges and the outside edges of a folded open document?

5.1.3. Vertical page margins

- `C1` - top page margin
- `C2` - bottom page margin
- are there differences between the first page and subsequent pages of the document?
- are there differences between the first page and subsequent pages of a section of the document?
- are there differences on the last page of the document or a section?

5.1.4. Vertical body margins

- `D1` - top body margin - the distance between the top of the header and the top of the body
- `D2` - bottom body margin - the distance between the bottom of the body and the bottom of the footer

5.1.5. Header and footer extents

- `E1` - header extent - the fixed height of the header (must be less than or equal to `D1`)
- `E2` - footer extent - the fixed height of the footer (must be less than or equal to `D2`)

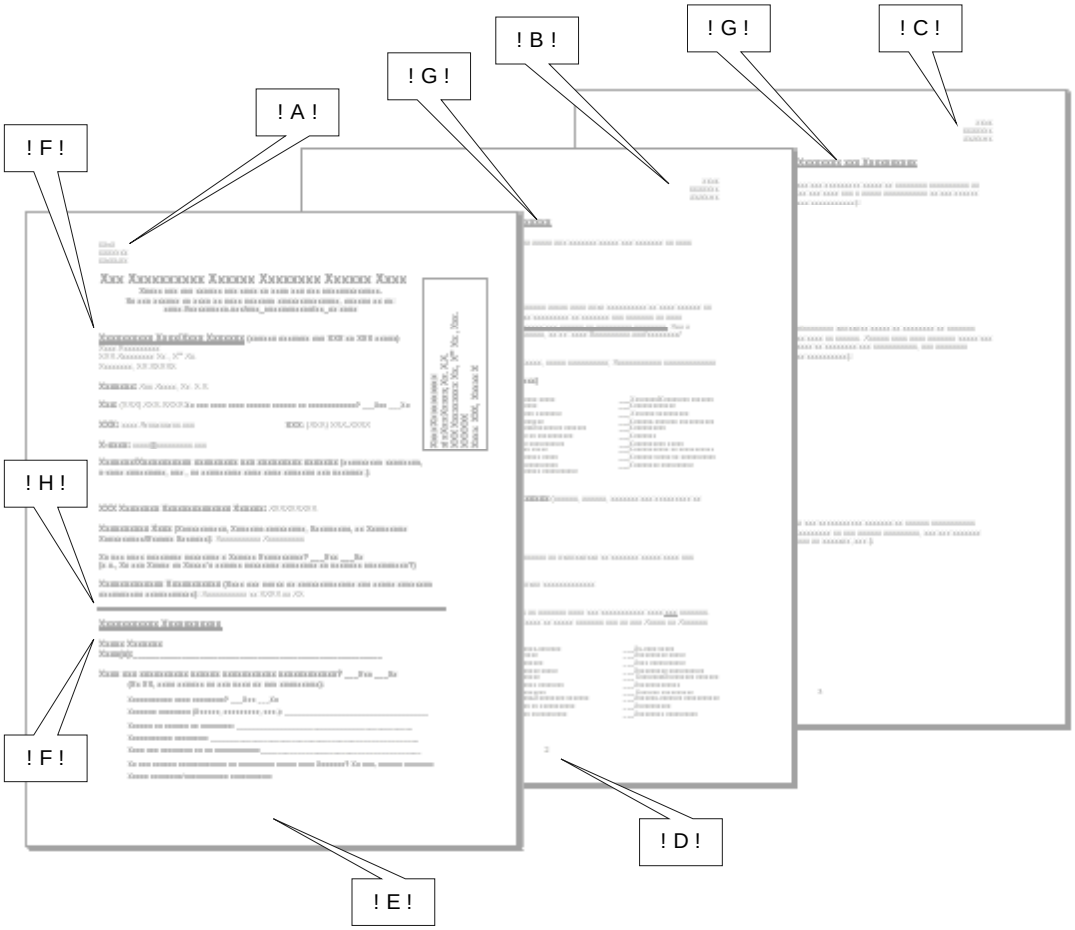
5.1.6. Column geometry and layout grid

Does your documentation predominantly use flowed columns (where an indeterminate amount of information flows from one column to the next)? At what points are all of the columns spanned with title, graphic or any other information? Are there any requirements to balance the lengths of multiple columns across the page when the last content does not fill the page?

Where information is determinate, is there any preset layout grid within which the information is placed? Can the preset layout be expressed as individual and spanned cells of an invisible border table? Need any edges of the table be visible, and if so with any kind of repetitive pattern (e.g. dots, dashes, etc.)?

5.2. Headers, footers and headings

Navigational aids to the reader of the document help them easily find a given document or find their place within the document.



5.2.1. Page header and footer content

- A - what information is found in the running headers and footers? Is any of it static (e.g. document type)? Is any obtained from the source files (e.g. document name)? Is any obtained during invocation (e.g. execution date)?
- B - are the running headers and footers always in the same location for every page? Is there an odd/even page number requirement for the inside or outside edge of a double page presentation? Is the positioning only different for the first page and the same for all subsequent pages? Is it different on the last page? Are there any requirements for a "`(continued...)`" signal to show up in a header or footer when in the middle of the content of a section?
- C - how is the header and footer content aligned vertically within the extents and horizontally within the margins? Is the header content flush against the top of the header and the footer content flush against the bottom of the footer?
- D - is there a page number and, if so, where is it located? How are pages numbered (e.g. decimal, alphabetic, roman numerals, etc.)?
- E - is the page number displayed on the first page of the document?

5.2.2. Section headings

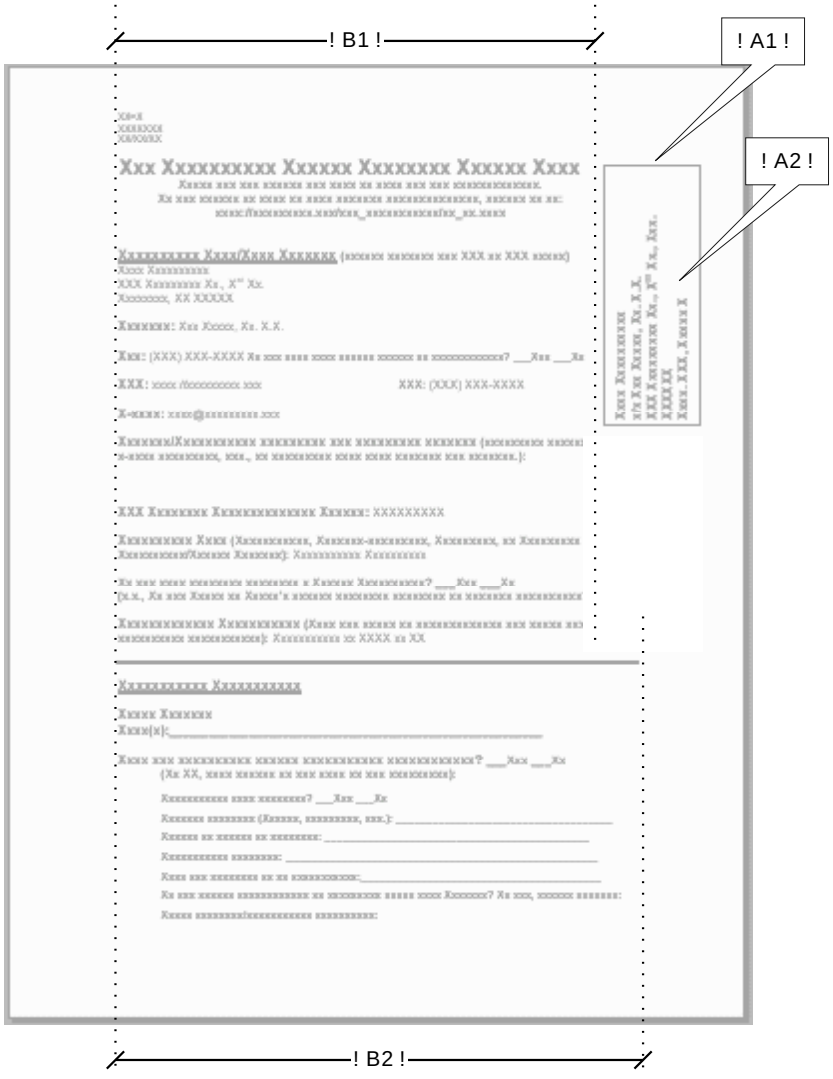
- F - how are section headings displayed? How much section content must be associated with (i.e. kept together with) the heading before being orphaned from the rest of the section? How much room is left after a heading and the start of the section content? How is the heading emphasized and distinguished from section content? Is any non-emphasized supplemental information displayed on the same line and along side the heading?

- G - do any sections force the start of a new page, perhaps so that the page can be manually extracted from a document easily? In a double sided presentation does the new page have to always be an odd or even page number? If a section does not force a new page, how much room is left before the section heading?

- H - are there any decorations (e.g. horizontal rules, graphic images, etc.) that belong before section headings?

5.3. Body

Does the body contain any out-of-line information and, if so, might its presence adjust margins temporarily?



5.3.1. Absolutely-positioned and out-of-line constructs

- A1 - are there any out-of-line constructs? What are the dimensions of each? What is the placement of each on the page? Is the placement relative to the flow or relative to the page?

- `A2` - what are the formatting rules for the construct and its content? Is the construct bordered? What are the margins of the information kept inside of the construct?

5.3.2. Impacts on flow margins

- `B1` and `B2` - does the presence of an object impact on the margins of information in the body? Where do any changes get restored to the full width of the column?

5.4. Blocks

Blocks of lines are the most frequently occurring construct in most printed outputs.

- which blocks have hanging indents where wrapped lines are indented further than the first line of a block, and how deep is that indent?
- what is the justification for a given block?
  - e.g. FL/RR - "flush left with ragged right"
  - e.g. Flandr - "flush left and right" (justified)
- what is the minimum number of lines left behind in a block that wraps the bottom of a column (orphan count)?
- what is the minimum number of lines at the end of a block that has wrapped to the top of a column (widow count)?
- is the line height determined by the tallest item on the line (a graphic, a character with a font larger than the block's font size, a superscript or subscript, etc.)? Should superscripts and subscripts be discounted from adjusting the line height? Should the line height be fixed for all lines in a block regardless of the content on the line (even distance between all text baselines)?

Block decorations can serve many purposes.

- does the block or line contain a horizontal rule for the purposes of inviting the reader to annotate the document?
- is the decoration exclusive of any document data that appears on the line, or in addition to any document data that appears on the line?

5.5. Fonts, styles and leading

Consistency in the usage of fonts serves the document reader well. Inadvertent unnecessary changes in font use can mislead the reader to imply significance where none is intended. To help ensure consistency, it would help if every use of font, in every situation of the document, be identified from a named list of style combinations. This process also helps to minimize the number of style combinations that might, when unchecked, result in a distracting display of character presentations.

It is best to create a table of style combinations with a name and the collection of properties for each entry in the table. For the first occurrences of each use of a style combination at documented points in the information, indicate enough of the names of the style combinations to establish a pattern with which all of the remaining blocks can be inferred. Examples of style names might be "`heading1`", "`heading1-supplemental`", "`heading2`", "`footer-info`", "`header-info`", "`mailing-label`", "`check-table`", "`boilerplate`", "`customer-data`", etc.

When setting the properties, consider the following facets:

- size of the text (e.g. `12pt`)

- size of the text in conjunction with the size of the line (e.g. `12pt/14pt` which implies a `2pt` leading)
- font family or a prioritized list of fallback font families should a particular implementation not support the desired font family
- any variant to the font (e.g. small caps)
- weight of the font (e.g. bold)
- style of the font (e.g. italic)
- decoration of the font (e.g. underline)
- superscript and subscript font sizes and amount of shift desired
- color foreground and color background
- any bordered edges?

5.6. Graphics

Many documents incorporate graphic images for either decoration (e.g. logos) or for conveying information (e.g. photographs, charts, etc.).

- in what data formats are the graphics stored?
- do the graphics have captions, and if so, how are the captions to be formatted in relation to the graphic?
- are there any distinctions between inline, block-level and out-of-line graphic images used in the document? How are each formatted and are out-of-line graphics displayed at the top or bottom of the formatted page?
- if the graphic is to be synthesized dynamically (such as with Scalable Vector Graphics (SVG) or MathML), what is the algorithm for deriving the image from the information?

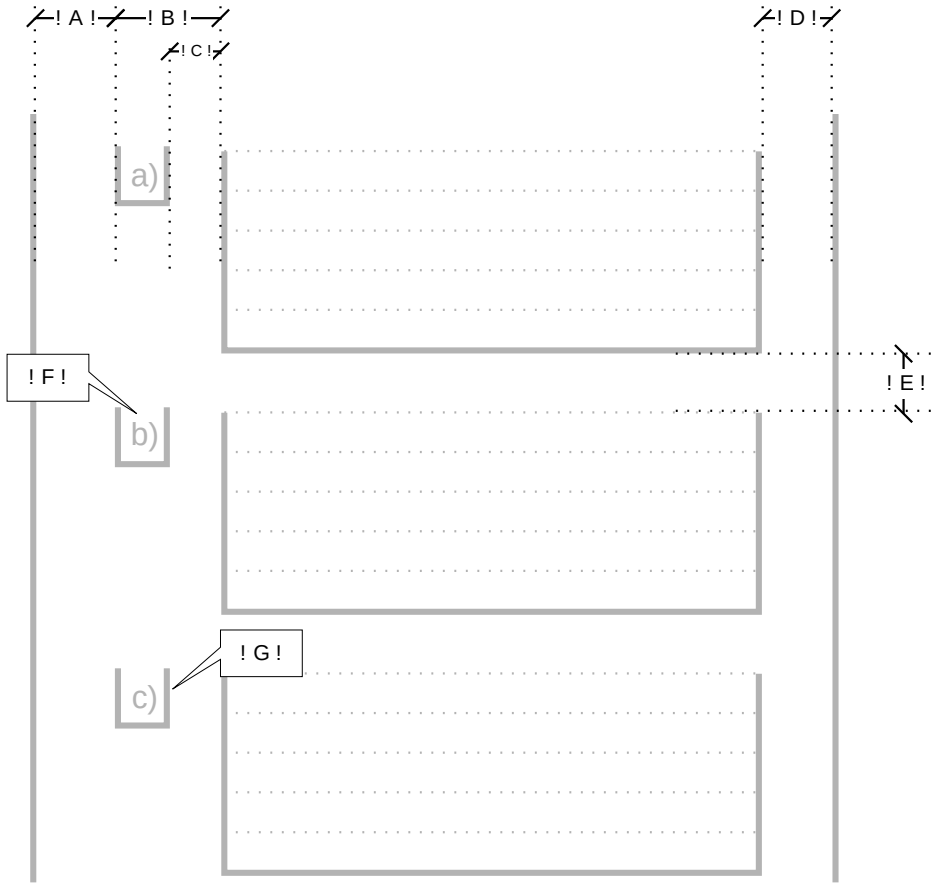
5.7. Lists

Many choices are available in traditionally formatted lists. A list is a collection of items of block level bodies, each distinguished by a label of some kind. Lists may be bulleted (unordered) or enumerated (ordered).

The edges of the labels and body are typically fixed for all items of the list, but may for special requirements be specified on a per item basis. For the typical case, the values requested below ensure a consistent display of list members across differing column widths.

The rules for each of the item's label and the item's body are the block rules described above. Many separate blocks may be used in either the label or the body block area of a list item.

See the discussion of tables for lengthy serpentine lists that are arranged in temporary columns (table cells) in the middle of the flow to preserve real estate on the page.



5.7.1. List item dimensions

- `A` - what is the distance between the left margin of the column and the start of the label area?
- `B` - what is the distance between the start of the label area and the start of the body area (this will set the start of the body area to be the sum of `A` plus `B`)?
- `C` - what is the distance between the end of the label area and the start of the body area?
- `D` - what is the distance between the end of the body area and the right margin of the column?

5.7.1.1. List item identification

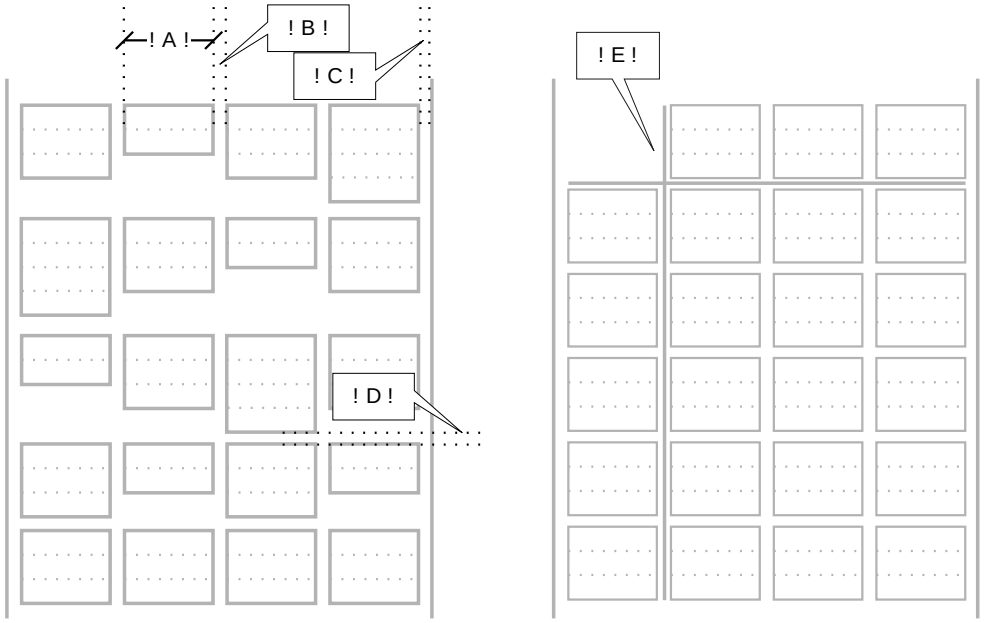
- `F` - how is each list item identified? What is the basis for any sequencing of enumerated (ordered) lists? Is there any punctuation used with the sequencing? Is any particular glyph or graphic used for itemized (unordered) lists?
- `G` - how is each list item label aligned within the label area? Is the alignment to the start of the label to ensure a common distance in all items between the margin and the label? Is the alignment to the end of the label to ensure a common distance in all items between the label and the body? Is the alignment to the center of the label to ensure a common center point between all labels of all items?

5.8. Tables

There are typically three uses for tables: tuples of block-aligned areas, a Cartesian coordinate layout of cells corresponding to row and column indexes, and serpentine lists of items populated in columns to save space on a page.

In these requirements, it is assumed that the column widths used in the table are fixed for the entire table. If you need varying widths of columns for the rows of the table, this will need to be described in detail in order to establish a strategy for layout that is atypical of standardized tables.

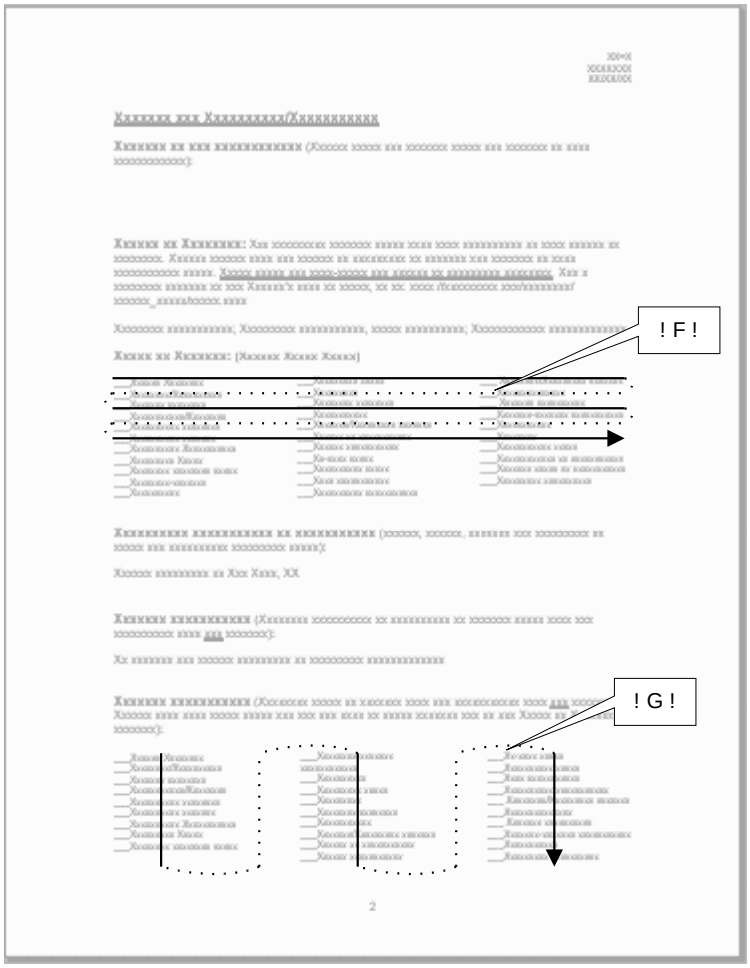
5.8.1. Traditional table layouts



- `A` - for each of the columns in the table, what is the width of that column (for the entire table)?
- `B` - between each of the columns of the table, are the edges coincident (overlapping one on top of the other) or is there a fixed separation between all of the columns of the table?
- `C` - what is the distance between the table and both of the margins?
- `D` - between each of the rows of the table, are the edges coincident (overlapping on top of the other) or is there a fixed separation between all rows of the table?
- `E` - are there any decorations around the table or inside of the table? Are any or all of the cells bordered with any particular visible rule pattern?

5.8.2. Serpentine lists in tables

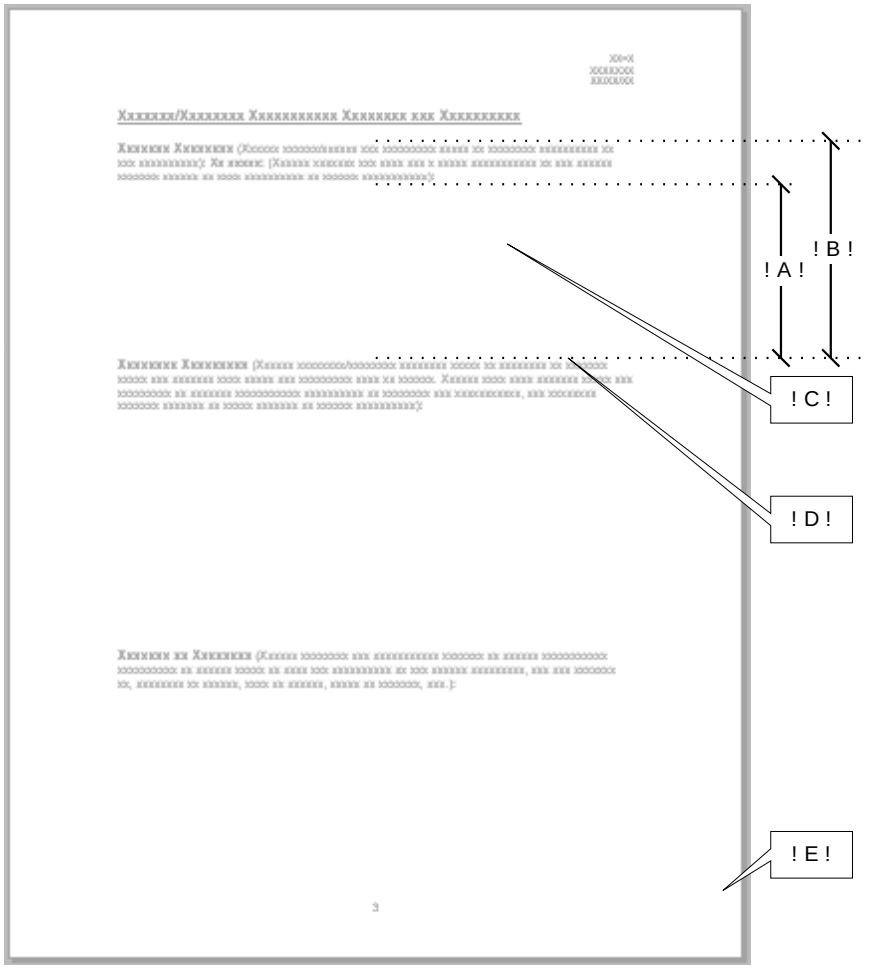
In the following page layout, there are two instances of what appears to be a three-column table on the page. The content of each table is a lengthy list of some kind, containing many members that would when otherwise formatted occupy too much length and leave too much white-space on the page. By organizing these lengthy lists into columns, real-estate is saved and the reader of the document can navigate the many members much more quickly.



- `F` - is a serpentine list populating the table columns in a row-dominant fashion?
- `G` - is a serpentine list populating the table columns in a column-dominant fashion?

5.9. Gaps in the flow

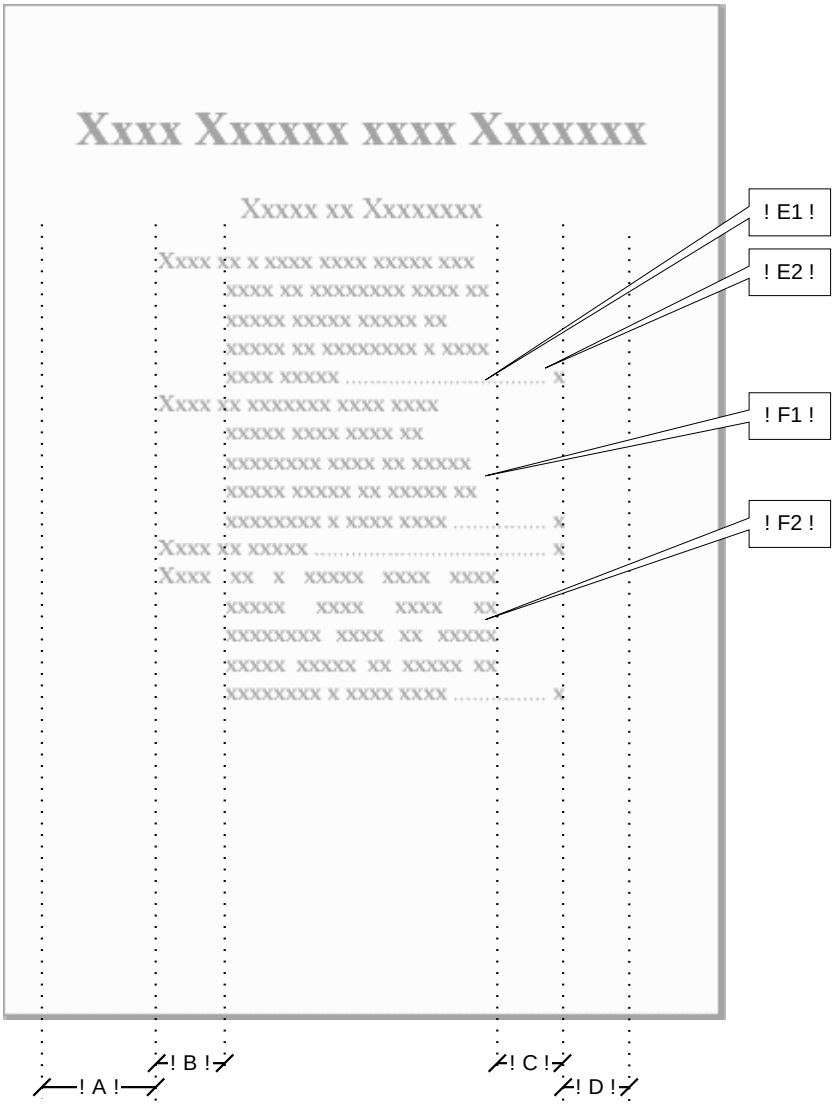
A gap in the flow is often required in order to leave room for the user to add their own information, or to give the impression that a lot of information is anticipated but may not be fulfilled.



- A - is the size of the gap exclusive of the preamble?
- B - is the size of the gap inclusive of the preamble?
- C - is there sourced content that belongs in the gap if present in the source? If absent, are there any generated indications for the reader that the sourced content is absent?
- D - is the size only a minimum, such that it grows when the content is too long to fit? If it does grow, is there a maximum size to which it is allowed to grow? If it does not grow, is the content to be clipped without error, or is an error to be signaled?
- E - if a preamble and the size of the gap do not fit at the bottom of the page, are both moved to the top of the next page?

5.10. Tables of content

There are a number of considerations when displaying tables of content. Each of these considerations is applicable at each level of depth within a table of contents. Even if you believe you will only have short entries that do not wrap, it is best to plan ahead and specify all values requested below in order to accommodate unexpected future needs.



- `A` - the distance between the start margin and the start of the entry
- `B` - the distance between the start of the entry and the point at which long entries wrap on the left
- `C` - the distance between the point at which long entries wrap on the right and the end of an entry
- `D` - the distance between the end of an entry and the end margin
- `E1` - what is the format of the leader (e.g. solid, dot, dash, etc.)?
- `E2` - is there a space separating the leader from the page number?
- `F1` and `F2` - is the end edge of wrapped entries ragged or flush?

5.11. Floats and footnotes

The float and footnote out-of-line constructs are utilized to ensure a smooth rendering of the flow by taking information out of the flow and positioning it at the top or bottom of a page.

5.11.1. Floats

Do any of the information components belong out of the flow and are to be positioned at the top or bottom of each page? Note that floating to the bottom is only supported if there are no footnotes in the information.

When floats are on a page, what style of separator is desired between the set of floats and the body of the content?

5.11.2. Footnotes

Are there any footnotes, acronym expansions or glossary definitions to be taken out of the flow and positioned at the bottom of the page? If so, how are the citations to be identified and sequenced?

When footnotes are on a page, what style of separator is desired between the set of footnotes and the body of the content?

6. Line-level processing

6.1. Keeps and breaks

When section headers are on a page, what are the rules for keeping their contents together in a column or in a page? Perhaps are there rules for keeping the first text of a section together with its respective heading? If so, how much of the content needs to be kept together?

6.2. Widows and orphans

What widows and orphans issues are important to your publication? When the end of a multi-line block flows to the next page, how many lines need to be kept together? If the lines of a section are comprised of many one-line blocks, how many of these need to be kept together when widowed from the rest of the section contents? Of the blocks that are candidates for widow-processing, are any of them conditionally present, or are they always present in the data?

7. Character-level processing

7.1. Hyphenation and breaks

Does your publication have specific hyphenation requirements? What languages are being used and can tools be found to support the hyphenation of these languages? Do different areas of the document have different hyphenation requirements?

Does your publication include very long strings (e.g. Internet URI strings) that may exceed the length of a line? What are the rules for breaking the string when the string length exceeds the line length?

7.2. Fonts and special characters

Does your publication utilize any special fonts? Are these fonts available for the operating system platform on which the tools are being used, and can the tools being used be modified to adapt to specific font requirements?

Can development be done without the fonts and then the delivered stylesheets be tested in the production environment with the desired fonts?