Typefi® Publish 6

Additional Modules Guide

August 2013

Typéfi®

typefi.com

This document was created with Typefi Publish 6.

6.0.4

Adobe | SOLUTION PARTNER Silver

Contents

Overset Text Detection with Warnings

Conditional Keeps

InDesign paragraph styles offer a number of keep options (such as 'Keep With Next'), but many publications require more stringent keep options. The Typefi Publish workflow, with its automated calling of scripts during document composition, allows for extra properties to be added to paragraph styles to support greater functionality.

Because these extras are implemented by scripts using data created, and managed, by the scripts, these extra features are not extensions to the paragraph styles themselves. Rather, the features are implemented by using local formatting to override the paragraph style settings where dictated by the specifications of the Conditional Keeps.

Conditional Keeps Options

Six conditional keeps are provided:

- Keep With Next Paragraph If …
- Keep With Next Paragraph If Not …
- Keep With Previous Paragraph If …
- Keep With Previous Paragraph If Not …
- Keep With Next If Previous Not …
- Keep With Previous If Next Not …

They all work along the same lines. You identify a paragraph style for which you want to apply a conditional keep and you provide a list of paragraph styles for which the conditional keep applies.

Naming Styles

Because, since InDesign CS3, paragraph styles can be grouped in the Paragraph Styles panel, a naming technique has been created to name styles that are contained in a group (or even deeper in the hierarchy—groups can be nested to any depth). The vertical bar character '|' is used to separate group names from each other when identifying a style in a group. ***This means that the vertical bar character cannot be used as part of a paragraph style name if that style is to be referenced by this feature***.

Examples of correct naming of paragraph styles:

BodyFirst identifies the style names 'BodyFirst' at the top level of the panel.

IndexStyles|firstLevel identifies the style 'firstLevel' in the top-level group 'IndexStyles'**.**

Defining Keeps in a Document

All Keep options are defined within the first table in a text frame labelled 'Conditional Keeps'. (This table should be placed on a master page created specifically for this purpose.)

All the keeps must be defined in the **first** table of the parent story of the text frame—this allows for the table to be overset if it becomes too long for the text frame.

To label the text frame:

- Open the Layers panel and click the triangle next to the layer name
- Click once on the name '<text frame>' (see *Figure 1*), pause, then click again
- Give the object the necessary name.

The table can have up to seven columns and any number of rows. The first column defines the styles to which conditional keeps are added. The remaining columns define the actual keep relationships. The first row holds the names of the relationships. The remaining rows list the styles to be considered when evaluating the conditional keep.

Here are two sample table which show all six different kinds of keep—two tables are presented because of the practical issue of column width in this document; in a real deployment of this feature, all the keeps should be in the same table. It doesn't matter if the table extends off the page:

Style	KeepWith Next_If	KeepWith Next_IfNot	KeepWith Previous_If	KeepWith Previous_IfNot
FigInline	FigCaption			FigInline

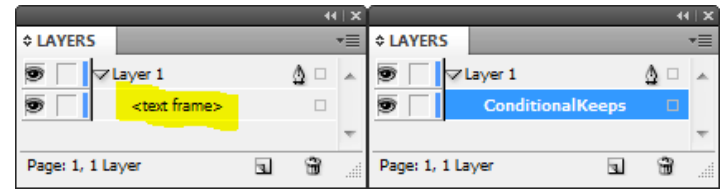

Figure 1: The text frame before and after naming it 'ConditionalKeeps'

Style	KeepWith Next_If	KeepWith Next_IfNot	KeepWith Previous_If	KeepWith Previous_IfNot
NL		NL NLsub		NL NLsub
BL			BL	

This table establishes five conditional keep relationships:

- Keep any paragraph in the style ***FigInline*** with the next paragraph If it is in the style ***FigCaption***.
- Keep any paragraph in the style ***FigInline*** with the previous paragraph if is not in the style ***FigInline***.
- Keep any paragraph in the style ***NL*** with the next paragraph if it is not in any of the styles ***NL*** and ***NLsub***.
- Keep any paragraph in the style ***NL*** with the previous paragraph if it is not in the style ***NL***.
- Keep any paragraph in the style ***BL*** with the previous paragraph if its style is ***BL***.

The first declares that a figure is kept with its caption, if there is one. The second allows a sequence of inline figures to break between figures, but declares that the first figure is kept with the preceding paragraph. The third declares that the first paragraph after a numbered list is kept with the last paragraph of the list (this is an example and not a recommendation). The fourth declares that the first paragraph before a numbered list is kept with the list (be aware this could have undesired effects if you have paragraphs interrupting a numbered list). The fifth keeps all the members of a bulleted list together.

Style	KeepWithNext_IfPreviousNot	KeepWithPrevious_IfNextNot
FigInline		
NL	NL	
BL		BL

This table shows two simple uses of these two somewhat complex conditions. In these examples, it is assumed that the lists start and end with variants of the list styles, such as ***NL_first***, ***NL_last***, ***BL_first*** and ***BL_last***.

- This rule, keep NL with the next paragraph if the preceding paragraph style is any other style, would assure at least the first two members of a list stayed together (although this could be more simply achieved by using ***KeepWithNext*** applied to the ***NL_first*** style).
- This rule, keep ***BL*** with the previous paragraph if the next paragraph is any other style, would assure that the last two members of a list were kept together (in CS5 and later paragraph style options, 'Keep with Previous' applied to the ***BL_last*** style would achieve the same effect).

Implementation of these rules is done by applying local overrides to the ***Keep With Next Lines*** property of the actual paragraphs. ***If the result creates the need for InDesign to override the usual keep options in order to compose the pages, it will do so***.

Note also that using 'Keep with Previous' with large figures can result in undesirably large blank spaces if InDesign is unable to keep the previous paragraph with the figure because there is not enough space on the page.

The Script

Conditional Keeps Script (ConditionalKeeps.jsxbin)

This is a ***spill.end*** script. It applies the conditional keeps described in the table found on the master page. It operates by first creating a cache of the keep rules to speed up further runs of the script (a ***spill.end*** script might be run hundreds of times during a Publish job).

Because of the possibility that a Typefi Publish-composed document may be reused as a template, it is important when including this script in a job to include the *DocumentStart.jsxbin* script, even if you have no 'DocumentStart' folder and no other scripts to run at this event.

To run this script interactively,

- you'll need to have a story selected in the document—the one to be processed, and you are advised to run this one-line script from the JavaScript Console of ESTK first:

```
app.scriptArgs.clear()
```

- you must run a ***SpillEnd.jsxbin*** script that will call this script by finding it in the 'SpillEnd' folder. You will not be able to run the script directly from ESTK.

Note: when running interactively, the script always updates the cache of stored conditions by interrogating the table of conditions on the master page.

Conditional Spacing

Controlling the space above (or 'before') and below (or 'after') is usually fine-tuned as needed before a job is printed. In the case of a Typefi project, this can be controlled automatically by the ***Conditional Spacing*** module. As the name suggests, the normal spacing may be over-ridden in certain situations, depending on the preceding or following paragraph.

Two conditional spacing options are provided:

- Suppress Space Above if preceding paragraph uses …
- Suppress Space Below if following paragraph uses …

Naming Styles

Because InDesign paragraph styles can be grouped in the Paragraph Styles panel, a naming technique is used to name styles that are contained in a group (or even deeper in the hierarchy—groups can be nested to any depth). There are normally no restrictions limiting the characters that can be used in style names, so Typefi have created a special limitation character if you wish to use Conditional Spacing with grouped styles. The vertical bar character '|' is used to separate group names from each other when identifying a style in a group.

Here are some examples:

- **BodyFirst** identifies the style named 'BodyFirst' at the top level of the panel.
- **IndexStyles|firstLevel** identifies the style 'firstLevel' in the group 'IndexStyles'

Defining Conditional Spacing

All spacing options are defined within the first table in a text frame that has been given the label 'Conditional Spacing'. (This table will be placed on a master page set up for the purpose.)

To label the text frame, open the Layers panel and click the triangle next to the layer name. Click once on the name '<text frame>' (see *Figure 1*), pause, then click again. You will then be able to give the object the necessary name.

The table can have up to three (3) columns and any number of rows. The first column lists the styles to which conditional spacing is to be applied. The remaining columns define the relationships:

- The first row holds the names of the relationships.
- The remaining rows list the styles to be considered when evaluating conditional spacing.

Implementation of these rules is achieved by the Conditional Spacing module applying local overrides to the Space Before or Space Above property of the affected paragraphs.

Enhanced Table Styling

Adobe® InDesign® allows table formatting to be stored as table and cell styles which can be applied to any table within a document. Table and cell styles allow you to quickly format a table with a predefined collection of formatting. When you make changes to a style, all the tables to which the style is applied are instantly updated. In a Typefi Publish workflow, table and cell styles make it possible to completely automate the formatting of tables. Although InDesign's table and cell styles can meet the design requirements of many publications, they do have limitations. Consequently, publications with more complex design and workflow requirements often require a more robust tool. Typefi Systems provides ***Enhanced Table Styling*** to satisfy more demanding requirements and avoid the need for manual modification of the generated InDesign document.

Enhanced Table Styling is designed to run at the Typefi Publish's spill.end event (which is a specific stage in the automated pagination process). It processes any new tables introduced by the spill.end event into the document being composed, and provides a variety of services that enhance the formatting of tables and their cells.

This module provides two mechanisms for enhancing the formatting of tables:

- **Region-based formatting:** you can set up a collection of particular cell styles that are automatically applied to specific regions of a table when the InDesign document is paginated.
- **Cell-specific formatting:** you can apply specific paragraph styles to the text within a table to trigger the application of corresponding cell styles, which gives you precise control over the formatting of individual cells. The paragraph styles are applied to the tables in Microsoft Word. The corresponding cell styles are then applied to the table cells when the InDesign document is paginated.

Region-based Formatting

Enhanced Table Styling utilises two sets of predefined cell styles: one for row-oriented regions and another for column-oriented regions. One or both sets are linked to a table style by placing them into a **cell style group** that has the same name as the table style with which it is associated. You can include all of the row- and column-oriented cell styles within a style group, or just the styles needed for a particular table design.

By setting up a template this way, your source documents only need to specify which table style should be applied to each table. The region-specific cell styles are then automatically applied when the InDesign document is paginated. Tables that lack a table style or have the [Basic Table] style applied are ignored.

Row-oriented Regions

By default, InDesign divides a table into three row-oriented regions: the header row, the body row, and the footer row. ***Enhanced Table Styling*** allows you to define far more specific regions, as indicated by the following list of cell style/region names:

When creating these names, don't include the colon in the name of the cell style!

- **row-head-first** : Applied to the first row of any multi-row header.
- **row-head-last** : Applied to the last row of any multi-row header.
- **row-head-single** : Applied to the header row when there is just one.
- **row-body-first** : Applied to the first row of the body of a table when the table has a header row.
- **row-table-first** : Applied to the first row of a table that lacks a header row.
- **row-table-single** : Applied to a table that consists of a single row and lacks a header row and footer row.
- **row-table-last** : Applied to the last row of a table that lacks a footer row.
- **row-body-last** : Applied to the last row of the body of a table when the table has a footer row.
- **row-foot-first** : Applied to the first row of a multi-row footer.
- **row-foot-last** : Applied to the last row of a multi-row footer.
- **row-foot-single** : Applied to the footer row if there is just one.

The row-oriented styles are applied in the order listed above, although certain combinations are not possible within an individual table. For example, the three 'single' row variants in a region are never applied unless a table region (header, body, or footer) contains just one row.. There is never a conflict among these regions, although there can be conflicts along the border between two regions.

There can be overlapping among some of the row-oriented regions, particularly when a table includes vertically merged cells. For example, in Figure 1, the left-most cells in the header row are merged, resulting in a cell that is a combination of both the first and last header row (as well as the first column of the header). Consequently, this cell could have as many as three styles applied to it. There can be overlapping of some row-oriented regions, particularly when a table includes vertically merged cells. For example, in Figure 1 the left-most cells in the two header rows are merged together, resulting in a single cell that is in both the first and last header rows (as well as the first column of the header). Consequently, this cell qualifies for as many as three styles being applied to it: row-head-first, row-head-last, and column-head-first.

Similarly, the single body row is both the first and last body row in the table, so it qualifies for both styles being applied to them. Furthermore, they are applied in the order listed above, so if two styles have the same attribute, the value in the second (or subsequent) style prevails.

When ***Enhanced Table Styling*** processed the sample table above, the 'row-head-first' style was initially applied to the first row of the header. Momentarily, the bottom stroke of the merged cell became a 2-point red stroke just like the bottom stroke of the other cells in the first row. Then the ***'row-headlast'*** style was applied to the last header row, which changed the bottom stroke of the merged cell to blue, to keep the formatting consistent across the row.

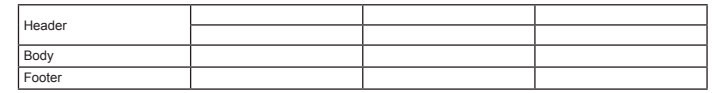

Figure 1: A small sample table

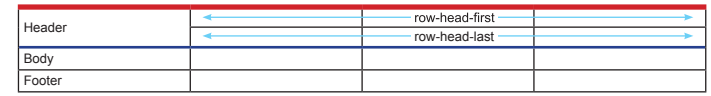

Figure 1a: The sample table after applying enhanced styles

Column-oriented Regions

InDesign doesn't provide a feature for column-oriented regions, but ***Enhanced Table Styling*** supports the following regions available for use in your templates:

If you use the first three cell styles in the list, the following nine styles are ignored; styles for the fragmented regions are applied only if the corresponding whole-column style is not defined. The styles will only be available if they have been created in the template.

- **column-single** : Applied to the whole table if it has a single column
- **column-first** : Applied to the whole first column if the table has more than one column
- **column-last** : Applied to the whole last column if the table has more than one column
- **column-head-single** : Applied to the header cells of a single-column table
- **column-head-first** : Applied to the header cells of the first column of a multi-column table
- **column-head-last** : Applied to the header cells of the last column of a multi-column table
- **column-body-single** : Applied to the body cells of a single-column table
- **column-body-first** : Applied to the body cells of the first column of a multi-column table

- **column-body-last** : Applied to the body cells of the last column of a multi-column table
- **column-foot-single** : Applied to the footer cells of a single-column table
- **column-foot-first** : Applied to the footer cells of the first column of a multi-column table
- **column-foot-last** : Applied to the footer cells of the last column of a multi-column table

Figure 2 shows a very simple table with just two columns and three rows; no header or footer. It has table style 'TestStyle2" applied to it.

'TestStyle2' requires a 20% Cyan fill to be applied to every other row. In the Cell Styles panel, there are two column-oriented cell styles defined in the TestStyle2 group folder:

- column-first_left stroke and fill
- column-body-last_right stroke

Annotating Style Names

The cell style names used for defining both row-oriented and column-oriented table regions must be named exactly as they are listed above; however the style names can be extended by inserting an underscore after the style name followed by the text of your choice. This allows you to add a more descriptive name to the styles in order to describe more accurately what formatting the style applies.

Here are two examples of cell styles that have additional text appended to the end of their names. The text describes what formatting the style applies:

- row-head-first_4pt red top stroke
- row-head-last_2pt blue bottom stroke

Cell-Specific Formatting

Some tables require special formatting for cells that contain certain kinds of paragraphs. For example, if a table has sub-heads in it that divide it into regions, it is a common design strategy to give the cells that contain such sub-heads a distinctive look to set off the regions within the table.

Cell Styles within 'ParagraphStyles' cell style group

The enhanced table styling script provides this by applying cell styles named for paragraph styles. It finds these styles by checking for a cell-style group named "Para-graphStyles" within the group of cell styles associated with the table style.

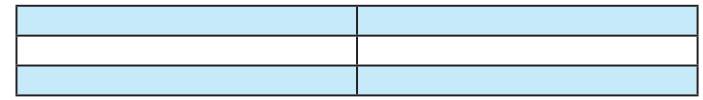

Figure 2: Small table with TestStyle2 applied

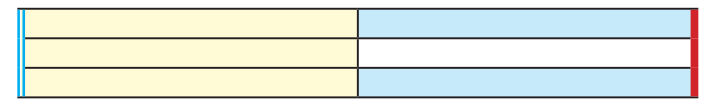

Figure 3: Same table after running script.

The names of the cell styles in the ParagraphStyles group must exactly match the paragraph styles used in the table. For paragraph styles that are grouped in the Paragraph Styles panel, only the name of the paragraph style is considered, not its path.

An Example of Paragraph Styles Driving Cell Styling

The simple table in figure 4 is set using table style TestStyle3. The region heads use a paragraph style named "Subhead".

Although it is not visible in the figure, the Subhead style is also applied to the cells in the same row as those containing the Region Head text.

You'll see that TestStyle3 bears a close relationship to TestStyle2 in that it has column-based region styles. But it also has a "ParagraphStyles" group that includes a cell style named "SubHead" which is the name of the paragraph style applied to the Region Head cells (see *Figure 5*).

In addition to demonstrating that the script did indeed apply the paragraph-style based cell style 'SubHead' to the region head cells, this also shows that the paragraph style-based cell styles override the region-based styles.

> Using ***Enhanced Table Styling***, when more than one cell style is applied to a cell as a result of overlapping regions, the characteristics of the cell style are merged to produce a composite style (unlike normal InDesign styling of cells). Only if there is a conflict in the style definitions does the order of application matter. When there is a conflict, the style applied later will prevail.

Tables within Tables

The script operates on all tables, even those that are contained within other tables.

Installing the Script

The script is designed to run at ***spill.end*** in a Typefi Publish job. It can also be run interactively from the InDesign Scripts panel when working with a desktop installation of Adobe InDesign, however, in that case, it will not run if directly activated from the panel. Instead, you must run the Typefi event-handling script 'SpillEnd.jsxbin' which will call the script – provided it is properly located in an adjacent folder named 'SpillEnd'.

Region Head 1	
Region Head 2	

Figure 4: Sample table with regions

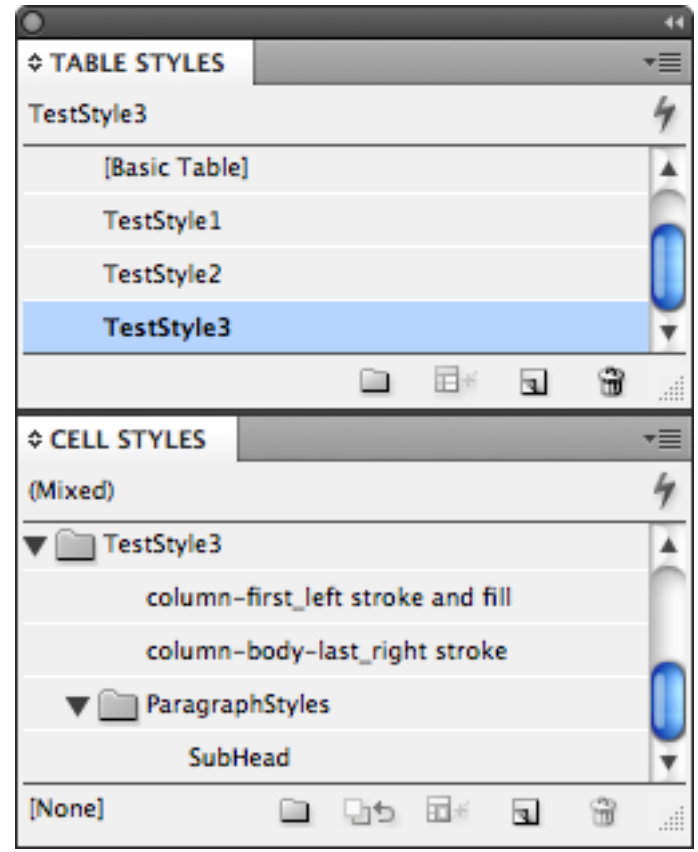

Figure 5: TestStyle3 and its associated Cell Styles Group

Setting up a template for Enhanced Table Styling

Step 1

Create the following default set of cell styles to correspond with each table style you create. The entire set must be placed within a ***style set*** (folder in the Paragraph Styles panel) that has **exactly** the same name as the table style they correspond to.

- row-head-first
- **row-head**
- row-head-last
- row-head-single
- row-body-first
- **row-body**
- row-body-last
- row-foot-first
- **row-foot**
- row-foot-last
- row-foot-single
- row-table-first
- row-table-last
- row-table-single

Step 2

After creating the group of cell styles, apply the following cell styles to a sample tables on a master page. The sample tables should have 3 header rows, 3 body rows, and 3 footer rows.

- row-head (apply to all three header rows)
- row-body (apply to all three body rows)
- row-foot (apply to all three footer rows)

Step 3

After applying the cell styles, edit the table style, click on the Layout Settings category, click ***Use Selected Table as Example***, and then click OK. **You must do this step; otherwise the row-body, row-foot, and row-head styles won't be applied. Instead [None] will be applied. This is due to the fact that Enhanced Table Styling doesn't apply these styles.**

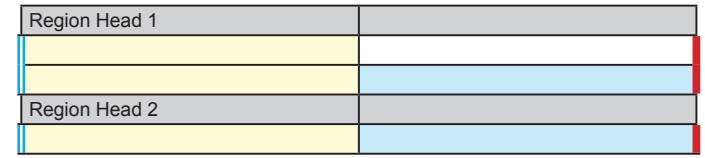

Figure 6: Sample table after running script

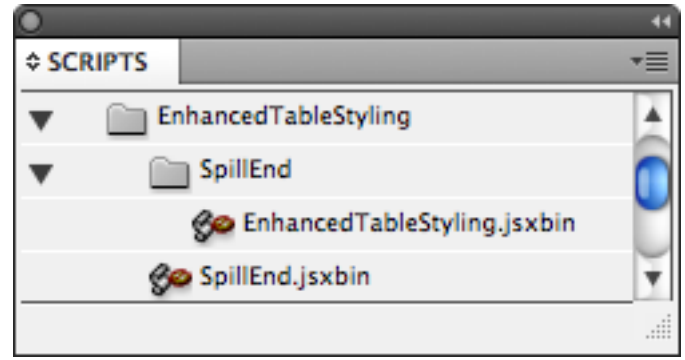

Figure 7: Enhanced Table Styling Script deployed in Scripts panel

Additional Notes

If you want to create custom cell formatting, set up a ***'ParagraphStyles'*** cell style set and add the custom cell styles to it. Ensure that a paragraph style with the same name also exists. Don't create paragraph styles for the standard group of cell styles listed in Step 1. You only need to create paragraph styles for each cell style that you put within the 'ParagraphStyles' style set.

The paragraph styles that link to cell styles located within the 'ParagraphStyles' style set must be available in the Word document so an editor can apply the styles. So ensure that they export to the TXML file.

If a table contains empty cells, you must insert a space character in the cell to ensure the correct cell style is applied to it in InDesign.

It's very important NOT to specify cell styles within a table style. Specify [None] for each cell style, which allows Enhanced Table Styling to do its job.

You must install/configure the ***DocumentStart.jsxbin*** script to clear the cache of the template.

When setting up cell styles, we HIGHLY recommend that you DO NOT base them on other cell styles. Specify just the cell properties you care about in each style, leaving all other properties set at Ignore. This is due to the fact that the script grabs the properties of the cell style in order to do the merging and that grabbing does not pick up every cell property in a reliable way. (This is due to an InDesign bug. ***https://typefi.basecamphq.com/projects/574013-indesign-event-scripts/posts/39620631/comments#comment_126914014***)

Make sure that the sample tables on the master pages don't contain local formatting that overrides the cell styles that are applied; otherwise that local formatting will continue to override those cell styles in the paginated InDesign document. This is happening because the table has been specified as the example table to use in the table style settings.

When setting up a sample table on a master page, pay close attention to the row heights. When you select the ***Use Selected Table as Example*** setting in the table style, it copies the row heights and continues to use them in the paginated InDesign document.

Overset Text Detection with Warnings

One of the major problems with publishing using InDesign is 'overset' text – content that does not fit within the assigned text frames and does not appear or print. This can happen as the result of text being added to a story, or typographical changes such as font, tracking or leading changes. Or perhaps an image might be replaced with a larger one, which will result in text being overset at the end of the story. To help overcome this problem, Typefi have created the 'Overset Text Detection' module.

The script '***zOsetText.jsx***' detects overset text in the stories of a document, highlighting each instance with a warning and showing the overset text on a page added to the back of the document. ***It will also seek out overset text in table cells***. The script can be run at the ***document.end*** event or after the fact, interactively.

The 'z' at the start of the script's name is there to force it to be alphabetically last among any other ***document.end*** scripts that might be part of a project. This forces it to run last so that it operates on a completed document.

Introduction to Script Operation

The script is entirely self-sufficient. It will run against virtually any document—the only known exception would be a document that has just one layer with the name ***'Overset Warning'***. The document does not even need to have been saved. For example, Figure 1 is a simple, one-page document with a single text frame that is overset (having been filled with placeholder text that was then enlarged):

If the script is run against this document, the result is a two-page document with a warning on page one and the overflow text on the second page, which will have been named ***page 1-1***, indicating that the content of the page is the first (and in this case, only) overflow text from page 1.

The results are:

The overset frame is labeled with a marker indicating that it is overset. The label indicates how many words are overset and it shows the first few words, indicating there are more with the ellipsis (hard to see in the figure because of the red text area outline).

A new page named ***1-1*** has been added to the document (see ***Figure 2***). The overset text is contained by a frame which in most circumstances is the same width as the as the overset frame, which allows you to see how the text behaves in it (particularly, how many lines it consumes). Only if a permanently overset condition is being caused by a frame being too narrow will the width of the overset frame be different from that of the original frame.

There's also a blue left-pointing triangle immediately above the frame. This is hyperlinked to the corresponding label, which in turn is hyperlinked to the overset text. These two hyperlinks provide quick navigation when examining the document. Use the ***Hyperlinks*** panel drop-down menu and choose ***Go To Destination*** to quickly flip back and forth between an overset label and the corresponding overset text.

All these new items are assigned to a new layer named "Overset Warning" with the UI colour red.

Notice that the overset frame is not threaded to the frame on page 1-1. There are two main reasons for this:

1. The overset text does not belong on that page; it is there merely to provide information about the contents and extent of the overflow.
2. Threading the frames could cause a recompose of the story changing the contents of the overflow.

The concept here is that the user can fix each of the overset instances and then re-run the script to confirm that there are indeed no more overset stories.

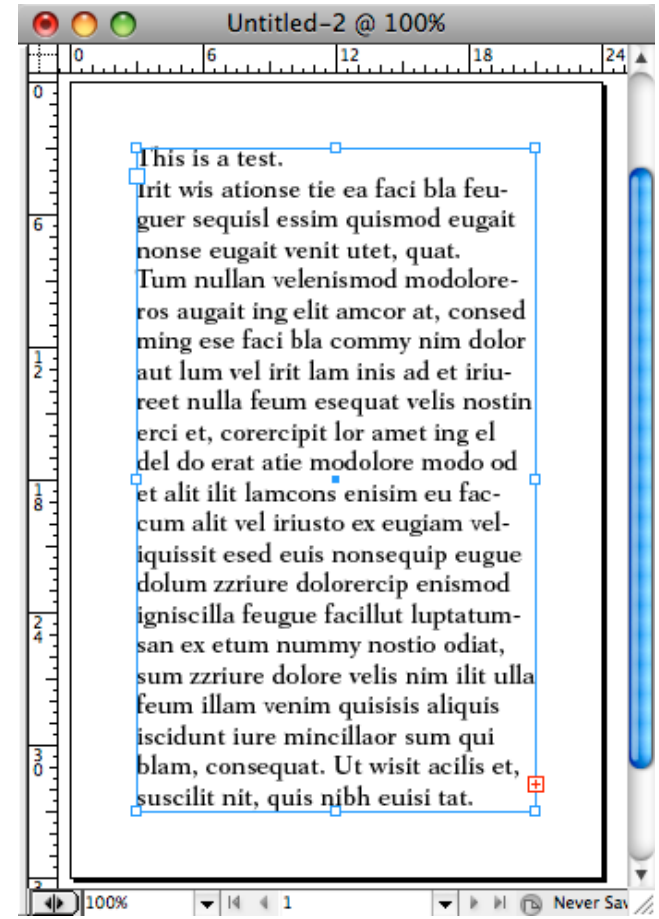

Figure 1: A text frame with overset text (indicated by a red cross in the lower right corner)

Operating within Typefi Publish

It is the nature of things that the main story in a Typefi published document will never be overset (see ***note***), but an element can be, and elements can be either free-standing or anchored. The overset text script distinguishes between these two possibilities.

If the element is **free-standing**, the warning takes the elaborate form discussed above. The default settings for the warning label are encapsulated in the object style named ***Overset Text Frame Warning*** with the look of the text being determined by the two paragraph styles ***Overset Warning Head*** and ***Overset Warning Details***.

(Typically, the default is that master pages are set to repeat. This setting may be changed, however, and this could result in the main story being overset. It would mean the main story would only fill one spread or page.)

On the other hand, for an element that is **anchored**, the label consists of just a rectangle drawn exactly over the overset text frame. The default settings for the rectangle are encapsulated in the object style ***Overset Anchored Frame Warning***. The basis for this choice is the assumption that anchored elements are typical relatively small. Thus, there is no room for the more elaborate approach.

User Options

The script makes use of a variety of styles to present the information. If these styles do not already exist in the document, the script creates them and the result has the look of the examples in the Introduction to Script Operation.

To customise the look of the labels, the user should predefine the object styles and paragraph styles by making sure that the project template has them already defined. The user can also affect the UI-colour of the warning layer by creating the layer (named "Overset Warning") in advance and choosing a colour from the options dialog. The script will force that layer to the front of the stack but otherwise work with it the way it is defined.

Object Styles

Overset TextFrame Warning: An object style that is used to label any non-anchored overset text frame. In a Typefi Publish workflow, this would be a text frame in an element. The user is free to create a customised version of this object style. The only restriction is that the text frame must have a text wrap of none. That is, the text wrap option in the object style definition must be enabled and set to None.

Overset Anchored Frame Warning: An object style that is used to highlight any anchored overset text frame. Again, the text wrap option in the object style definition must be enabled and set to None.

Hidden Metadata: If an overset text frame has an object style applied to it with this name, it is ignored by the script.

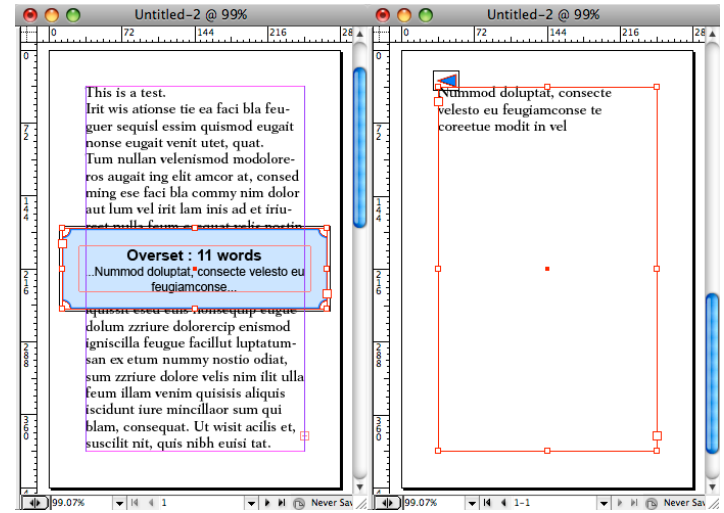

Figure 2: The same page with the Overset Detection warning appearing, and the extra page added to the document, containing a copy of the overset text and a hyperlink to the original page.

Paragraph Styles

Overset Warning Head: This paragraph style is used for the first paragraph of an overset text frame warning.

Overset Warning Details: This paragraph style is used for the second paragraph of an overset text frame warning which displays the first few words of overset text. The paragraph style Keep Options must be set to 'Start Paragraph: Anywhere', otherwise just about any other formatting is acceptable.

Layer

Overset Warning: This layer is used to hold all warnings associated with overset text (in the case of overset non-anchored text frames) as well as a copy of the overset text itself. The user can customise the UI-colour used for this layer by predefining the layer in the template.