Single Sourcing – An Introduction

By Justin Darley

12 January 2003

Summary

Many documentation departments produce detailed and well-designed **paper** documentation. Increasingly, however, one paper manual is not enough. Pressure is growing to deliver your information online, perhaps on several different user platforms. You may need to reuse the information in your manuals for quick reference guides, training courses and marketing publications. This can mean that you find yourself re-creating virtually identical content over and over again. Duplicating your material for multiple media, multiple uses and multiple audiences is time-consuming and costly.

Single sourcing offers a solution to this problem by allowing you to output each document in many different formats, but only store and work with a single version.

This article:

- Provides an introduction to the benefits of single sourcing
- Summarises the obstacles to successfully completing a single source project
- Outlines some single sourcing strategies and writing techniques
- Introduces some single sourcing tools and technologies.

For more information

Cherryleaf Ltd

Cherryleaf House

31 Arlington Road

Ashford

Middlesex

TW15 2LS

www.cherryleaf.com

Tel: 01784 258672
justin@cherryleaf.com

Section 1 - What is single sourcing?

Single sourcing is:

The process of producing multiple outputs from one "single source" of information.

Successful single sourcing allows you to create and maintain one single set of information and yet produce a range of different outputs. For example, with a successful single sourcing system you can produce online Help, a website, marketing literature and paper installation guides.

True single sourcing takes into account the differing needs of all of your audiences and the strengths and weaknesses of each of your output media. For example, you may, at the simplest level, want to produce both paper and online Help from a single source. However these two media are very different. Paper is easy on the eye, reflective, high resolution. A computer screen on the other hand is emissive, low resolution, and, as a direct result, tiring and stressful to read from for more than a few minutes at a time. In addition, paper documents are structured linearly and the pages are numbered. Online information is much more anarchic in its structure and allows access at multiple points and topics to be read in various sequences. And we haven't even got to the differences between audiences!

As you can see, single sourcing is complex and challenging, but it also yields benefits that we will investigate in the next section.

Benefits of single sourcing

Single sourcing has been regarded, for some time, as the "holy grail" of publishing. Why is it such a sought after goal? The simple answer comes down to hard cash. Single sourcing can, particularly where large information sets are involved, deliver large savings in time and effort. When a piece of information is created, not only must it be written, but it must also be formatted, reviewed and published. When this information is duplicated this production effort is also, to a greater or lesser extent, multiplied. The following table shows **some** of the ways in which single sourcing addresses this problem:

Problem	**Why single sourcing can save time**
Information written with a paper medium in mind may well be too dense and wordy for an online user.	Rules applied to content modify the level of detail according to output medium or purpose.
Cross-references in hard copy manuals need to be replaced with hyperlinks online.	Cross-references are automatically replaced with hyperlinks where appropriate.
Formatting designed for high-resolution printing may make your information difficult to read from a screen.	Design issues are automatically dealt with by rules applied to content depending on its output medium or purpose.
When information is duplicated inconsistency in content is likely to creep in.	Information is written once.
As all of your publications become different (or perhaps were never the same) the requirements for review grow.	Accuracy of content is reviewed once in the single source. Review of outputs is for look-and-feel, fitness for purpose etc.

Thanks to these increases in efficiency, information can be produced both more quickly and to a higher standard.

All this talk of automation may lead to some uneasiness in the minds of those currently responsible for producing content. While single sourcing may reduce the overall effort required in the production of information, it could also lead to an increase both in specialisation and skills. Indeed, communicators of all sorts may find themselves writing, designing, managing or manipulating information across areas of their organisation that have hitherto been outside of their remit.

Obstacles to single sourcing

So far so good, but clearly the issues that created problems when we were simply duplicating information have not disappeared. If single sourcing were easy we would have been doing it for years. So what are the obstacles that have kept us cutting, pasting, editing and reformatting for so long?

The problems associated with single sourcing are intimately linked with the benefits mentioned above. Successful single sourcing will, in the long run, free us from manually editing content to suit a range of different outputs. Unfortunately, to achieve the potential gains, and avoid the duplication of effort we may have been used to, we need to create mechanisms that will account for the different media, purposes and audiences that we aim to cater for.

This can be a time-consuming and potentially costly process. It is a process in fact that impinges on each of the areas of information development. The following table shows just some of the areas for concern:

Writers	• Need to take account of phrases that may exclude users accessing their information via particular media or platforms. How will users click on a paper manual? How will Mac users push the Windows key? • Need to be aware of each of their audiences and of which pieces of information are relevant to each.
Information designers (responsible for the visual design of information)	• Need to research extensively the differences in the proposed media. Will that fetching shade of cerise be rendered appropriately on a 256-colour display? Will it be legible if printed in greyscale? Are the fonts you have chosen installed on a Macintosh?
Information technicians (responsible for developing the technical process)	• Need either to create a technological framework, or to learn and make use of one of the off-the-shelf solutions.
Knowledge managers	• Need to be familiar with the revised roles and considerations of both writers and information designers. It may well fall to knowledge managers to hold the whole process together. • May find themselves liaising with content developers who have previously moved in very different circles and written to very different rules.

A need for a user focus

All of these obstacles and changes of roles serve one major purpose: ensuring that single sourcing addresses the needs of users. The aim is to ensure that all outputs are usable and the key is user focus. All parties involved need a clear perception of the needs and limitations of the targeted users.

Each output must be customised to cater as closely as possible for its potential users. If your single sourcing effort is 100% successful then users should receive information as perfect for their use as if it had been written solely for them. Perhaps this goal is unrealistic, but I believe we can get very close. With thought and care it is possible to deal with these obstacles. By adopting the following strategies you can create single sourced information that is customisable to meet the needs of a range of users, diverse media and multiple goals.

Strategies for single sourcing

The required strategies for a successful single sourcing project fall into two categories: analysis and granularity.

Analysis

Those of you familiar with Cherryleaf's user-focused philosophy may well know what's coming next! The key to any successful piece of documentation is to analyse the intended audience. This focus on the user's needs is, if anything, strengthened in a single sourcing environment. As mentioned above, we need to know all about our users. In a single sourcing environment, however, the analysis doesn't stop here. Not only do we not have a unified (or at least broadly similar) audience for our information, but we also have users in a range of environments using a range of equipment.

To cater for this change in circumstances we need to be better informed on a range of technical issues. At the simplest level we need to re-acquaint ourselves with the fundamental differences between paper and online formats, but it doesn't stop there. If we are truly single sourcing then our users will be accessing the information on a range of platforms. You will need to know what the Macintosh equivalent of the Windows key is (or the PC equivalent of the Apple key for Mac users). The online formats that you produce may also differ: you will very likely need to move beyond Microsoft HTML Help. Content itself may differ: installing software, for example, on a Linux machine may be very different to installing on a PC.

Aside from the technology, you may well need to appreciate the requirements of a broader range of audiences. Training courses can be very similar to user guides but not identical: they are likely to require a greater depth of explanation and the provision of practical exercises. A sales brochure on the other hand may require much higher-level explanations of functionality.

In brief, you need to analyse the following for each output:

- Who are the users of this information?
- What do they need to know?
- On what medium are they viewing the information?

For online outputs, or for software documentation:

- What platform are your users using?
- What are the limitations and possibilities of this platform?

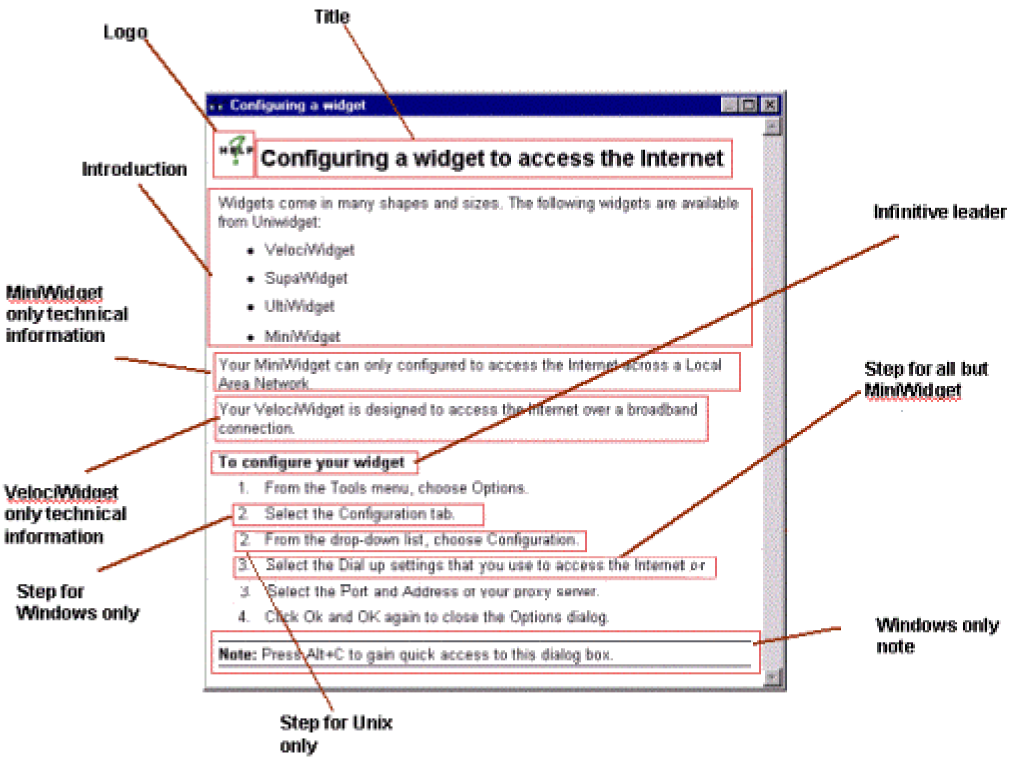
Once these areas have been examined we can move on to the technique that will provide us with the degree of flexibility needed to make one source of information cater for a wide range of uses and audiences: granularity.

Granularity

To achieve the flexibility required to produce multiple outputs from your single source you need to be able to do three things:

- Represent a unit of information in more than one way
- Include or exclude units of information based on output
- Use a unit of information to carry out more than one function

Doing any of these requires that these units of information be identified. As a Help Author by trade I was used to working with units of information generally no smaller than a topic. Those of you who primarily write for paper may work in larger units still. For single sourcing we need to increase the level of granularity of our information, in other words we need to think about our information in smaller units. An example:



Configuring a widget to access the Internet

Widgets come in many shapes and sizes. The following widgets are available from Uniwidget:

- VelociWidget
- SupaWidget
- UltiWidget
- MiniWidget

Your MiniWidget can only configured to access the Internet across a Local Area Network.

Your VelociWidget is designed to access the Internet over a broadband connection.

To configure your widget

1. From the Tools menu, choose Options.
2. Select the Configuration tab.
2. From the drop-down list, choose Configuration.
3. Select the Dial up settings that you use to access the Internet or
3. Select the Port and Address or your proxy server.
4. Click Ok and OK again to close the Options dialog.

Note: Press Alt+C to gain quick access to this dialog box.

As you can see the size of a unit (or grain) of information may well be much smaller than a page or topic. We might need to exclude content as small as a single sentence or even a word or letter. We may also need to exclude tables or images.

Granularity is not just about including or excluding content. We will also need to identify grains of information to alter their appearance or to change how they are used. The title of the topic above would probably be a simple heading in a paper output. In HTML however we may well want to out put this information twice: as a heading and as the title element for the file. We may also want to perform string splicing and convert it to serve as the name of the file.

These points:

- Excluding content
- Altering appearance
- Altering function

serve to identify the grains that we need to control. Anything in your information that you may need to exclude, reformat or alter the function of will need to be identifiable as a grain.

Your chosen single sourcing technology will then collect these grains and assemble and format them as appropriate for each of your outputs. In other words, if the process that generates the outputs is to include or exclude something, or to change how it looks, it needs to be able to find it. This is usually done by one of two methods: by holding the grains as fields in a database, or using a mark-up language.

Section 3 The technological framework

Database driven single sourcing

Database driven single sourcing identifies grains of information as fields within a database. These fields can then be identified and assembled using a selection of queries. It is not always immediately obvious that a particular solution is making use of database technology; often the workings of the database are hidden behind a graphical user interface (GUI). You may configure your single source project simply by checking boxes and making selections from lists.

Mark-up based single sourcing

Where grains of information are not identified as fields in database they may well be identified by use of a mark-up language. HTML, perhaps the most widespread mark-up language, is not of itself flexible enough to allow for single sourcing as described above. In HTML it is easy to identify that a piece of text is a paragraph, or a list, but how do you identify that it is a paragraph describing conceptual information? Or that one of the sentences in the paragraph is relevant only to Unix versions of the software?

What is needed is a mechanism to add more information to the mark-up. This can be done, for example, by supplementing the HTML mark-up with CSS classes. These classes can be intuitively named to identify the elements required:

```
<p class=”ConceptualUnix”>Information</p>
```

Alternatively a different mark-up scheme can be used. XML, for example, allows the creation of elements. This increases the flexibility and does away with the need for the kind of ‘secondary’ mark-up shown above.

```
<ConceptualUnix>Information</ConceptualUnix>
```

Section 4 - The software to use

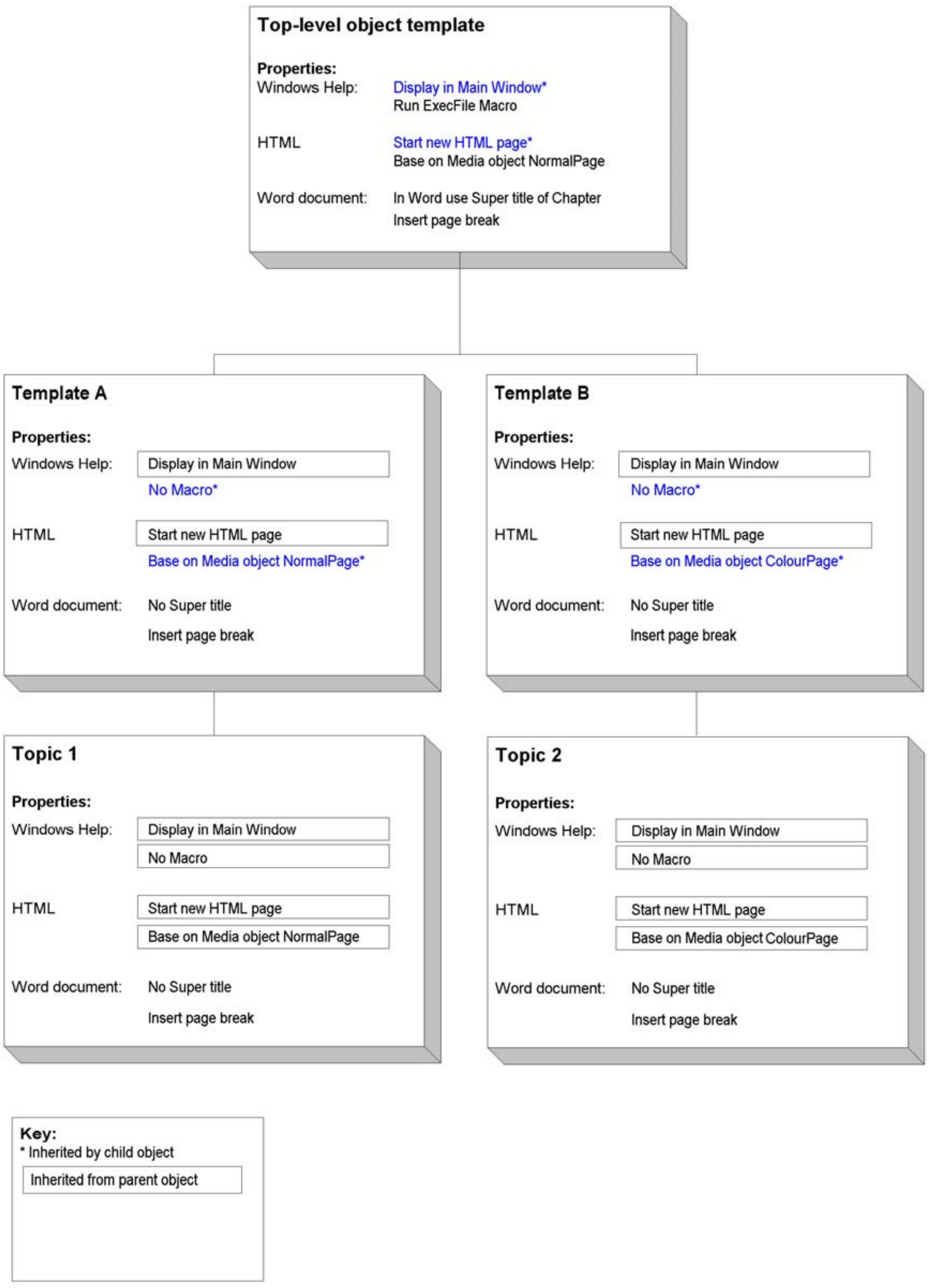
So we have looked at the benefits and obstacles to single sourcing, at the design implications and briefly at the technology that underpins it. Do we now need to learn database skills or the programming languages associated with XML and set about developing our own solution? The answer, as so often, is that it depends. Good off-the-shelf solutions are available and we will look at these in the next section. However, if you want or need to maximize the flexibility of your single sourcing system, the technologies that enable you to do this may not be beyond your grasp.

AuthorIT – a database driven solution

AuthorIT (the developers don't mind how you pronounce it as long as you buy it!) hails from New Zealand and is, to the best of my knowledge, the only Help Authoring Tool to implement an object-oriented philosophy. This philosophy results in a very new way of working. Each grain of information is stored as an object within the project. Rather than working in units of pages, or even documents, each topic, or part of a topic, each style, each window and each image (to name some of the objects available) is stored as a separate object with its own properties.

These properties include settings which dictate whether or not the object should appear in a particular output format and if so how it should appear. The objects are re-usable, which allows flexibility in producing multiple outputs. Rather than duplicating content the objects are simply referenced into a number of "Book" objects. Each Book object represents a subset of content to be output from the project.

Finally, AuthorIT implements the object-oriented concept of inheritance as seen in the following diagram:

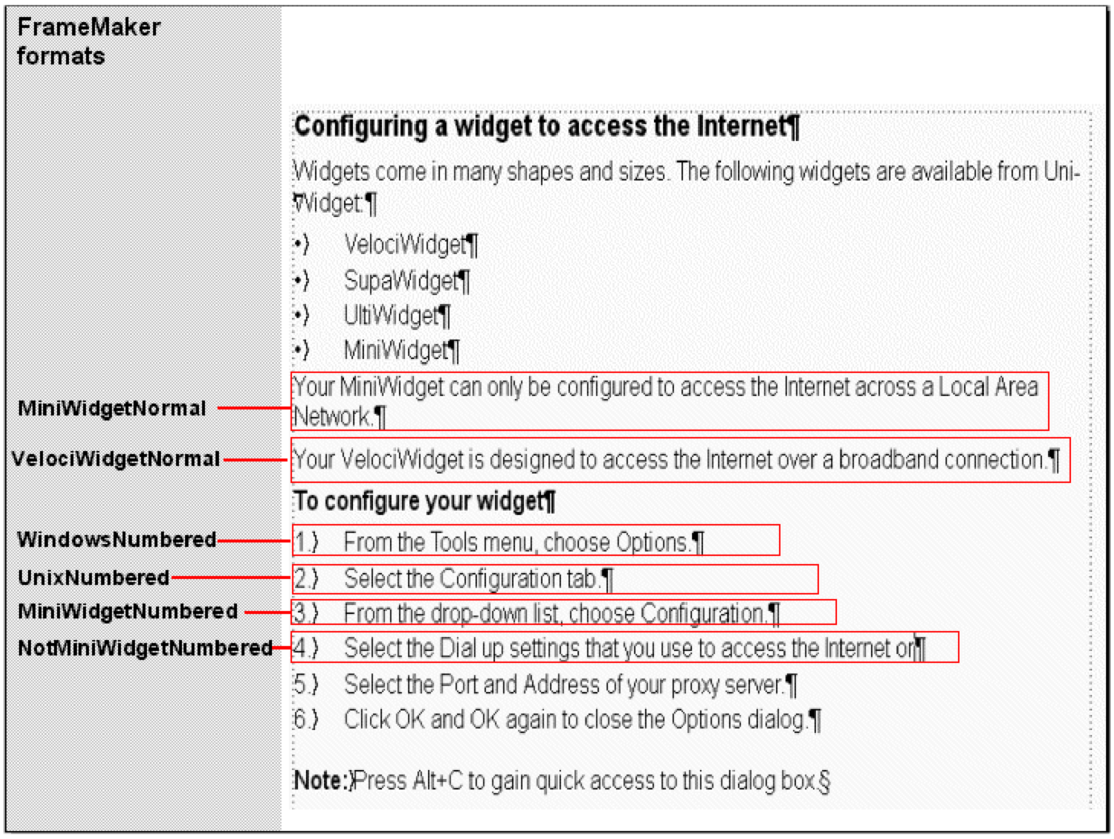


As you can see, properties can be inherited from their parent objects. Any changes then made to these properties in the parent will ripple through to all child objects. In addition inherited properties in AuthorIT can only be altered in the parent object. Inheritance provides global control over the objects in a project.

WebWorks – a mark-up based solution

WebWorks Professional from Quadralay works in conjunction with FrameMaker documents. Elements are identified by means of FrameMaker paragraph and character formats. Within the WebWorks interface these FrameMaker formats are mapped to WebWorks styles. Using these styles you can exclude content where appropriate. In addition, WebWorks styles, as well as the obvious formatting function implied by the name, provide access to a macro language allowing sophisticated changes to be made to the output produced.

From a design point of view, this mark-up based approach will change the way you use FrameMaker formats. Where previously you will probably have used FrameMaker formats primarily or entirely to control the visual appearance of your information, they now have a more far-reaching role. In a mark-up solution the mark-up is the only way to identify your grains of information. Since the mark-up in this solution is the FrameMaker formats this is likely to result in what may seem to be duplication of formats. In other words you may well need to create several styles that, in FrameMaker, all look the same. This is demonstrated in the diagram below:



Configuring a widget to access the Internet¶

Widgets come in many shapes and sizes. The following widgets are available from Uni-Widget:¶

•) VelociWidget¶
•) SupaWidget¶
•) UltiWidget¶
•) MiniWidget¶

Your MiniWidget can only be configured to access the Internet across a Local Area Network.¶

Your VelociWidget is designed to access the Internet over a broadband connection.¶

To configure your widget¶

1.) From the Tools menu, choose Options.¶
2.) Select the Configuration tab.¶
3.) From the drop-down list, choose Configuration.¶
4.) Select the Dial up settings that you use to access the Internet on¶
5.) Select the Port and Address of your proxy server.¶
6.) Click OK and OK again to close the Options dialog.¶

Note:)Press Alt+C to gain quick access to this dialog box.§

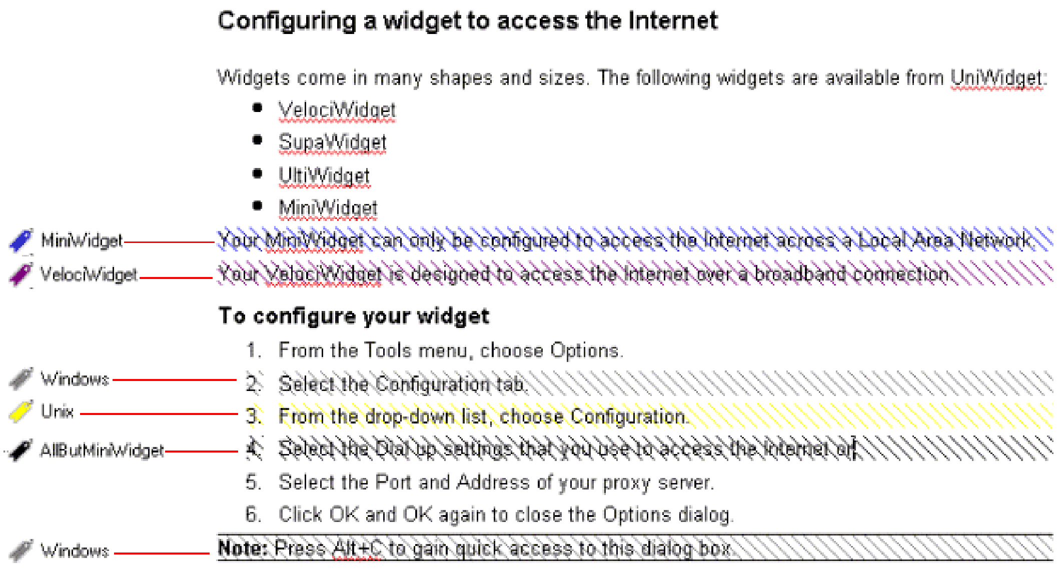
In the above diagram, MiniWidgetNormal and VelociWidgetNormal are identical in appearance as are all of the numbered steps. The additional styles exist to identify these grains of information.

RoboHelp X3

Finally, let’s look at a new kid on the block. Many of you may know RoboHelp well and have been using it for years. It is, by quite some distance, the most widely used Help Authoring Tool but, until recently, allowed only fairly limited single sourcing. The launch of X3, however, has moved the tool squarely into the single sourcing arena.

RoboHelp X3 is a mark-up based solution but, rather than rely on standard styles as seen in WebWorks, eHelp have implemented secondary mark-up in the form of “build tags”. Those of you familiar with previous versions of RoboHelp may well have used these build tags in the past to include and exclude topics. The difference now is that the tags can be applied not only to whole topics but also to paragraphs, characters, tables or images. In other words the level of granularity has been increased.

Identifying your grains of information is achieved as shown in the following diagram:



Configuring a widget to access the Internet

Widgets come in many shapes and sizes. The following widgets are available from UniWidget:

- VelociWidget
- SupaWidget
- UltiWidget
- MiniWidget

Your MiniWidget can only be configured to access the Internet across a Local Area Network.

Your VelociWidget is designed to access the Internet over a broadband connection.

To configure your widget

1. From the Tools menu, choose Options.
2. Select the Configuration tab.
3. From the drop-down list, choose Configuration.
4. Select the Dial up settings that you use to access the Internet or
5. Select the Port and Address of your proxy server.
6. Click OK and OK again to close the Options dialog.

Note: Press Alt+C to gain quick access to this dialog box.

The build tags are assigned to the relevant content and identified by colour coding. These build tags can then be excluded using “conditional build expressions”. Conditional build expressions make use of Boolean logic and can range from simply excluding certain build tags from an output through to complicated expressions requiring combinations of tags.

XML and XSLT – a non-proprietary system

If the above proprietary solutions are not flexible enough for your requirements, and you aren’t afraid of a bit of coding, there is a non-proprietary system that may well provide you with the flexibility you need. This solution involves two technologies; one may be familiar one probably less so.

What is XML?

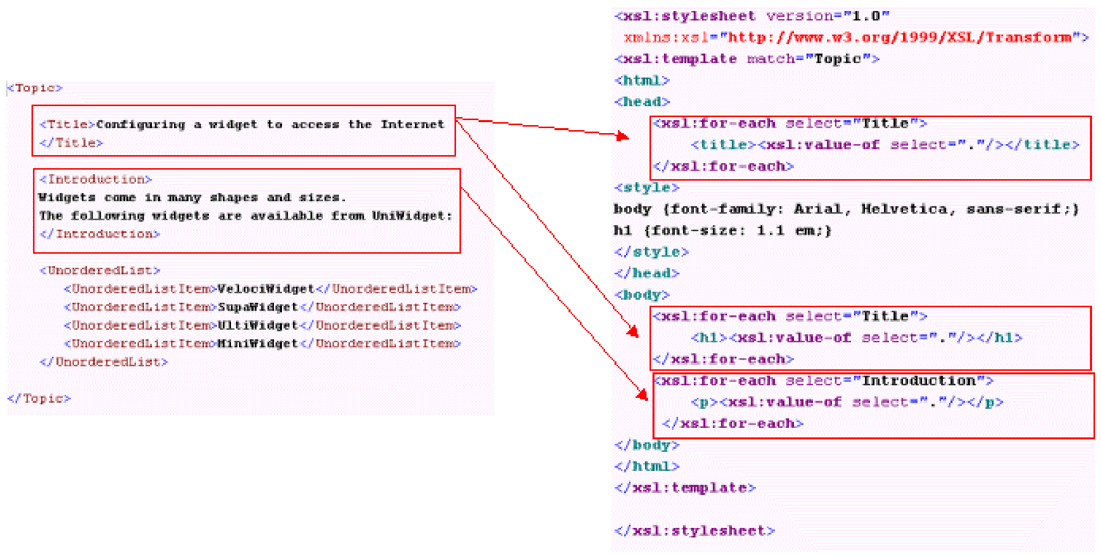
XML (eXtensible Mark-up Language), as mentioned above, is a mark-up language that allows you to define your own tags. This flexibility allows you to define a tag for each grain of information that you need to identify. So far so good, but having identified these grains how are we going to exclude them or alter their functions? One answer is to use XSLT.

What is XSLT?

XSLT (eXtensible Stylesheet Language Transformations) is a language for transforming XML documents. Using XSLT you can manipulate your XML document identifying the required elements and outputting them as any structured language. You are not limited to any particular output: using XSLT you can output your XML documents as HTML, WML, PDF, or if necessary other XML documents. You can, in fact, transform your source information into any structured format.

Example

An extremely simple example of this transformation is shown in the diagram below:



```
<Topic>
    <Title>Configuring a widget to access the Internet
    </Title>
    <Introduction>
    Widgets come in many shapes and sizes.
    The following widgets are available from UniWidget:
    </Introduction>
    <UnorderedList>
        <UnorderedListItem>VelociWidget</UnorderedListItem>
        <UnorderedListItem>SupaWidget</UnorderedListItem>
        <UnorderedListItem>UltiWidget</UnorderedListItem>
        <UnorderedListItem>MiniWidget</UnorderedListItem>
    </UnorderedList>

</Topic>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="Topic">
<html>
<head>
    <xsl:for-each select="Title">
        <title><xsl:value-of select="."/></title>
    </xsl:for-each>
<style>
body {font-family: Arial, Helvetica, sans-serif;}
h1 {font-size: 1.1 em;}
</style>
</head>
<body>
    <xsl:for-each select="Title">
        <h1><xsl:value-of select="."/></h1>
    </xsl:for-each>
    <xsl:for-each select="Introduction">
        <p><xsl:value-of select="."/></p>
     </xsl:for-each>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

The XML file (left) is transformed by the XSLT file (right) into the following HTML:

```
<html>
<head>

<title>Configuring a widget to access the Internet
   </title>
<style>
body {font-family: Arial, Helvetica, sans-serif;}
h1 {font-size: 1.1 em;}
</style>
</head>
<body>
<h1>Configuring a widget to access the Internet
   </h1>
<p>
   Widgets come in many shapes and sizes.
   The following widgets are available from UniWidget:
   </p>
</body>
</html>
```

Notice also that the <Title> element is used twice in the example. The line:

```
<xsl:for-each select="Title">
```

finds all `<Title>` elements in the XML topic, then the line:

```
<title><xsl:value-of select="."/></title>
```

writes this text to the `<title>` element of the HTML output.

This construct is then used again but this time the line:

```
<h1><xsl:value-of select="."/></h1>
```

writes the text to the `<h1>` element of the HTML output.

Conclusion

In conclusion, single sourcing can save you time and effort if you have to produce multiple outputs of similar information. It is not a quick fix. To be successful you need to

- Spend some considerable time in planning and designing.
- Learn about your users and the media and technologies you are using to present your information.
- You may well need to break your information down into smaller units than you are used to. While the technologies depend, behind the scenes, on mark-up or on databases, good proprietary solutions are available and these will shield you from the complexity.
- Finally, if you're feeling brave, you can create your own solution.

Resources

Software vendors

http://www.author-it.com

http://www.webworks.com

http://www.ehelp.com

User lists

http://groups.yahoo.com/group/authorit-users/

http://groups.yahoo.com/group/wwp-users/

Support site

(includes access to user lists and knowledge base)

http://www.helpcommunity.ehelp.com/robohelp/

XSLT

http://www.w3c.org/Style/XSL/

http://www.vbxml.com/xsl/tutorials/intro/