Conditional Cross-Reference Labels for Numbered Headings

or

How to Create Cross-Reference Labels that Adapt to Heading Level Changes

Introduction

Find out how to set your heading cross-references free. This article solves a perennial problem with cross-references that technical writers have encountered in technical documents that use numbered headings. The article:

- Provides methods for creating conditional cross-reference labels (e.g., Chapter, Section, Topic) that automagically adapt to heading level changes.
- Describes in detail how to create the solution in Microsoft Word and explains why it works.
- Will help technical writers who use Microsoft Word, because they will no longer have to take time to correct or maintain cross-reference labels, which would normally require special interventions (such as updating each instance manually or using a repair macro).

If you use the conditional cross-reference labels as described here, they will be unaffected by any future changes you may make to the headings in the heading level hierarchy. That is, you may move, promote, or demote them in the heading level hierarchy without needing to change the labels.

This article also contains some helpful information on creating structured cross-references.

How To Use

Although, with the help of this article, you will be able to create the conditional cross-references manually, it is not necessary if you download the Sample Document (LabeledXrefsSample.doc). The Sample Document contains working cross-references with conditional cross-reference labels.

The following guidelines describe the best way to approach this and how to make best use of the information in this article.

Familiarize

In case you need a summary of basic information on the Microsoft Word features used in this solution, see ***Background***. The features described include:

- ***Outline Numbering (Multilevel Heading Numbers)***
- ***Cross-Reference Fields (REF)***
- ***Conditional Text Fields (IF)***
- ***Formula Fields (Numeric Calculations)***

Quick Start

Although you can create the conditional cross-references manually, the easiest and fastest way to implement the solution is to download the Sample Document and use the field codes in it as a basis. It's also a good way to see first-hand how the solution works. Here are some suggested steps to do this.

- Read ***Introduction*** and ***The Problem*** if you haven't already. (Skip them if you already know about conditional cross-reference labels.)
- Download the Sample Document and the Sample Autotext Template. Put the Autotext Template in your templates folder (or make it an add-in).
- Follow the instructions in the Sample Document to see how the solution works.
- To use conditional cross-references in your documents, take the **3-label solution with autotext** from the Sample Document. (Although the Sample Document contains 3-, 4-, and 5-label solutions, most people will only need the 3-label solution. Also, the Sample Document shows two text methods (autotext and literal text); the autotext version is recommended because it allows you change the text globally.)
- To take full advantage of the solution, you will need to create a macro for automatically inserting the cross-references into documents. For a simple way to insert them into documents, see ***Automation***.

Digging Deeper

If you want to delve into the workings:

- To explore some useful ways to adapt the solution to your needs, see ***Customizations***.
- To understand how the basic solution works, read ***The 3-Label Solution***.
- To understand the how the 3-label solution was extended, read ***Extensions***, ***The 4-Label Extension***, and ***The 5-Label Extension***..
- To go crazy with it, read ***More Is Possible***.

Downloads

The following sample files are provided to help make things easier:

- A PDF of this article (LabeledXrefsArticle.pdf).
- A Sample Document (LabeledXrefsSample.doc & pdf): Contains conditional cross-reference fields in full working order (illustrating the solutions described in this article), which you can use to create your own. See also ***Quick Start*** in this article, which contains instructions on adapting the content of the Sample Document to your needs.
- A Sample Autotext Template (LabeledXrefsAutotext.dot & pdf): Contains autotext entries (and boilerplate text explanations) for use with the conditional cross-reference fields in the Sample Document that rely on autotext.

The Problem

A widespread practice in technical documentation is to use numbered headings, and when referring to these headings, to identify them with an electronic cross-reference that includes the heading number and heading text. Electronic cross-references are not "hard-coded" and so do not require individual maintenance to update them. If you change the heading text, the heading cross-reference will automatically be maintained when you update the fields in the document.

A common convention in these documents is to use different labels to refer to the heading based on its heading level. Heading 1s are typically identified as "Chapters" while subheadings below that (Headings 2-x) are identified as "Sections." If the heading is non-numbered, the label may be "Topic," or something else, or even no label. Examples of cross-references to 4 different heading levels:

> See "Chapter 1: Introduction."
>
> See "Section 1.1: Features."
>
> See "Section 1.1.1: Standard Features."
>
> See "Topic: Hardware."

The label (e.g., Chapter, Section, Topic) is part of the supporting text for the cross-reference, which also includes the introductory clause ("See" or "For details, see" and so on), delimiters between the elements, and punctuation. These are the elements of a **structured cross-reference**.

Although it is sometimes done, the label is not usually a part of the heading numbering because it would result in very busy and cumbersome looking headings. In this type of documentation, most people prefer a simple heading number rather than a label and number inherent in each heading.

The problem is that if you change the heading level or promote or demote it in the heading level hierarchy, the cross-reference does not get updated with the appropriate label. Using the example above, if you promote heading "1.1 Features" to the chapter level by tagging it with Heading 1 style, the updated cross-reference will read: See "Section 1: Features" rather than See "Chapter 1: Features." These incorrect labels would then require some type of special intervention to update them, such as manual updating or a repair macro.

Many writers try to get around this by adopting a generic label for all heading levels. One generic label that is frequently adopted by writers is "Section." This avoids the need to update the labels, but there are still those purists among us who find it an awkward though expedient solution. A chapter is still a chapter after all, no matter how many people call it a section, and there's no use trying to convince these traditionalists otherwise.

The methods described here provide a solution to this problem.

Requirements

For the purposes of demonstration, this example adopts the following requirements:

The Headings: Headings 1-3 are outline numbered, and Headings 4-9 are non-numbered, as in the following (simulated) samples of Headings 1-4:

1. **Introduction**

1.1 **Features**

1.1.1 ***Standard Features***

Hardware

The Cross-References: The desired results for the cross-references to these headings are as follows:

See Chapter: 1. Introduction on page X.

See Section: 1.1. Features on page X.

See Section: 1.1.1. Standard Features on page X.

See Topic: Hardware on page X.

The supporting text, including the label names, the punctuation (colon-space after the label name) and the delimiter after the number (period-space) can be chosen as desired.

The goal is to create complete cross-references to these headings that are independent of the heading level hierarchy, and even more, that adapt to the hierarchy, so that if the heading levels are changed or moved, the cross-references, including the heading labels and numbers, still give the desired results once the fields are updated.

Assumptions

The multilevel heading numbers use simple outline numbering with period separators, such as 1.1.1.

The heading numbers do not contain labels inherent in the outline numbering definition; that is, the heading numbers are plain multilevel numbers and do not contain descriptive labels, such as “Chapter,” “Section,” “Topic,” “Appendix,” etc.

Background

The following background information describes several capabilities of Microsoft Word and how they apply to the subject of this article. They may be of use for following the instructions in this article.

Working with Field Codes

The conditional cross-reference labels are created through field codes. To manually enter the field codes, use Ctrl-F9 to insert field code delimiters (which look like curly braces), Shift-F9 to toggle the selected field between the field code and the result, Alt-F9 to toggle all field codes, and F9 to update the selected fields. **Tips:**

- It is sometimes easier to type the field coding, select it, and press F9 to insert the field code delimiters, which will enclose the selected text.
- When working with **revealed field codes**, if you select a field code delimiter, which looks like either an open-brace character (" { ") or close-brace character (" } "), it will select the entire field. (The field code delimiters are special characters that are different from the curly braces displayed in normal text.) This is one way of revealing the interior parts of nested fields that belong together to visualize the groupings or understand the logic better. **One neat trick** to make this method **even better** is to search for the end of field delimiter (field code close-brace). To do this, enter ^21 in the Find dialog to specify its character code. (After you find the first, you can then use F4 to repeat the find.) When working in nested fields, each time you find the end of field delimiter (close-brace), it will select the next level of nested fields. If instead you were to search for the start of field delimiter (field code open-brace) (^19), it would select the entire group of nested fields.

Outline Numbering (Multilevel Heading Numbers)

Numbered headings are often used to help readers and customer support personnel locate and refer to content, and they are especially helpful in long documents, which is one reason they are so widespread.

Each heading is numbered sequentially as expected (so, for example, chapters are numbered 1, 2, 3, etc.). Subheadings are also numbered sequentially, but in addition they are prefixed with the numbers from the prior levels above it, which therefore shows the heading's position in the heading hierarchy and gives it a unique number sequence within the document. This combination of numbers is a multilevel heading number, with each number in the sequence separated by a delimiter, which is usually a period.

Example: Heading number 1.1.1 represents chapter 1, section 1, subsection 1, and the heading paragraph is tagged with a Heading 3 style.

Microsoft Word provides many ways to implement numbering, including heading numbers. A good resource for information on numbering methods is http://word.mvps.org/FAQs/Numbering.htm.

Outline numbering is used by the solution in this article. Excellent instructions for this touchy numbering area are provided by Word MVP Shauna Kelly at www.shaunakelly.com/word/numbering/OutlineNumbering.html, as well as others, including Microsystems at www.microsystems.com/pdfs/sevenlaws.pdf.

Cross-Reference Fields (REF)

When you insert an electronic cross-reference to a heading or heading number, Microsoft Word creates an internal or hidden bookmark at the heading. The bookmark name takes the form _Ref123456789. The _Ref prefix identifies it as a hidden, internal reference-field bookmark. (A checkbox to see Hidden Bookmarks on the Insert Bookmarks dialog was introduced in Word 97.) Microsoft Word also inserts a REF field using that bookmark name. For example: { REF _Ref123456789 \h }. The optional \h switch in the field creates a hyperlink to the bookmark. Among other field switch options, you can insert a cross-reference to the heading number, the heading text, or the entire heading including the number and text. You can also insert a cross-reference to the page on which the heading appears. The switches used by the examples in this article are as follows:

\h (hyperlink)
\n (heading number)
\p (page reference above/below/on page <page-number>)

The formatting switches * CHARFORMAT and * MERGEFORMAT are also used when adding character styles, which are useful for identifying the cross-reference elements and distinguishing them from the surrounding text.

Conditional Text Fields (IF)

The solution described in this article pivots on the use of the conditional text field (IF field). It is often used with the mail merge feature in Microsoft Word. The conditional text field evaluates a logical condition (e.g., compares 2 values or expressions) and depending upon the result, outputs one of 2 text values.

The basic syntax for the IF field is { IF Condition “TrueText” “FalseText” }. This is an IF/THEN/ELSE construction, with implied THEN and ELSE keywords. Alternatively, the output of the IF condition ("TrueText" or "FalseText") can be numeric (TrueValue or FalseValue).

In Microsoft Word, IF fields can be nested 20 levels deep, but the 3-label solution needs only 2 levels, each with a single instance. The 4-label solution uses 3 levels, each with a single instance. The 5-label solution uses the same 3 levels as the 4-label solution, but contains a second instance of the third level. It also uses a fourth level with 8 instances to produce a numeric result within a formula expression.

This article also describes extensions to the basic solution, and these employ wildcard characters for the logical comparison of strings.

Formula Fields (Numeric Calculations)

The = (Formula) field, which performs numeric calculations, is used in the conditional text field comparison. The basic syntax for the formula field is { = Formula [Bookmark] }. Although an optional numeric picture switch is also allowed, it is not needed for this solution. A formula field contains an expression that consists of any combination of numbers, bookmarked numbers, and fields that output numbers. Numeric operators and functions can also be used, including arithmetic and comparison operators.

The solution uses the = (Formula) field with a bookmark, which in this case is a cross-reference to the heading number (e.g., 1.1), which is a REF field with a numeric switch. The formula also uses the INT function, which operates on a decimal number in the form x.y and returns the integer portion, i.e., the numbers to the left of the decimal place (x).

The 3-Label Solution

The solution generates the appropriate labels in the cross-reference (e.g., Chapter 1, Section 1.1, Section 1.1.1, or Topic), and the labels automatically adapt to heading level changes.

The wizardry of this solution is that the heading number cross-references are independent of the heading level/hierarchy, which allows you to move or to promote or demote the headings/sections to any heading level without requiring any changes in the cross-references other than updating the fields with F9. The field labels (Chapter, Section, Topic) and heading numbers will automatically adapt to the heading level.

The solution uses the = (Formula) field combined with nested IF fields. It provides 3 different labeling options (Chapter, Section, Topic) based on the heading level:

Heading 1 (numbered with positive integers)
Headings 2-3 (decimal or multilevel numbered)
Headings 4-9 (non-numbered)

However, the solution works equally well with outline heading numbering in which Headings 1-6 are numbered and Headings 7-8 are non-numbered, as well as with other variations. It does not matter how many levels are numbered, because all numbered headings below Heading 1 will fall into the Section category and all non-numbered headings will fall into the Topic category. If there were no numbered headings at all, they would all would fall into the Topic category.

Field Coding for 3 Conditional Labels

If you insert a cross-reference to a heading and build the following fields from the cross-reference bookmark, it will create conditional labels that adapt to the heading level. Of course, you will need to substitute a valid bookmark name for the "_Ref123456789" placeholder shown in this example. Only a single cross-reference bookmark is needed to build these fields. That is, you can build the cross-reference fields from a single _Ref bookmark name, because field switches control whether to use the heading number or heading text, and the page reference also uses the same bookmark name. (The following is a text representation of the field coding. For keyboard entry, use Ctrl-F9 to insert field code delimiters, Shift-F9 to toggle selected fields between the field code and the result, Alt-F9 to toggle all field codes, and F9 to update the selected fields.)

```
See { IF { REF _Ref123456789 \n } = 0 "Topic: " "{ IF { = INT ( { REF _Ref123456789
\n } ) } < { REF _Ref123456789 \n } "Section: " "Chapter: " }{ REF _Ref123456789 \n
}.  " }{ REF _Ref123456789 \h } { PAGEREF _Ref123456789 \p \h }.
```

Note: Pre-built field codes are available in the Sample Document and the Sample Autotext Template.

The Cross-Reference Fields

As mentioned, the 5 REF fields use the same heading bookmark, as does the PAGEREF field. Of these, the 3 conditional REF fields are identical, and are simply a REF to the heading bookmark using a heading number switch (\n) and no other switches.

The other 2 REF fields are used for output: One is a heading number field and the other a heading text field. The heading text field contains a hyperlink switch (\h), whereas the heading number field that is used for output does not use a hyperlink switch. This is because the heading number field is output from within the conditional IF field, and even if the switch were present, it would not result in an active hyperlink that is clickable in Microsoft Word, so it is unnecessary.

Finally, the PAGEREF field uses the same heading bookmark and also relies upon the above/below field switch (\p) to accomplish the "on page X/above/below."

The Conditional Logic for 3 Labels

The pseudocode for the 3-label conditional cross-reference follows. One fact on which this solution relies: If you create a heading number cross-reference field that points to a non-numbered heading, the field result is a zero. So for non-numbered headings I use a well-known test for heading number 0, while for numbered headings, I am able to use the new technique described here to distinguish between chapter numbers and section numbers and to output the labels accordingly.

In the following pseudocode, a "section number" means a heading number with 2 levels of numbering or more (i.e., in the format of x.x or x.x.x or longer).

```
IF no heading number (HeadingNumber = 0) THEN
    Use Topic Label
ELSE
    IF a section number x.x or x.x.x or longer (INT < HeadingNumber) THEN
        Use Section Label
    ELSE
        Use Chapter Label (HeadingNumber > 0)
    [ENDIF]
    Output the HeadingNumber + Delimiter (for numbered headings)
[ENDIF]
Output the (unconditional) HeadingText + PageReference
```

Explanation of the Logic for 3 Labels

If the heading has no number (Headings 4-9), the heading number cross-reference is zero (0). The real trick is to distinguish between Heading 1 chapter numbers (e.g., 1) and Heading 2 and 3 section numbers (e.g., 1.1 and 1.1.1). This is done through the nested IF condition. The following is a more detailed explanation of that portion of the pseudocode.

It is reasonably easy to distinguish between Heading 1 numbers (e.g., 1) and Heading 2 numbers (e.g., 1.1) by comparing the integer portion of the number with the full number, because for Heading 1 numbers they will be the same, whereas for Heading 2 numbers the integer portion will be less than the full number. Therefore, the following test can be used to compare Heading 1 with Heading 2 numbers:

```
IF the INTeger portion of the (HeadingNumber) < HeadingNumber THEN
    It is a Level 2 Section Number (x.x), so use the Section Label.
ELSE
    It is a Level 1 Chapter Number (x), so use the Chapter Label.
```

Test Cases for Levels 1 and 2:

- If the HeadingNumber is 1.1 (a Level 2 section number), then the test for the INT (1.1) < 1.1 is **true** (because 1<1.1), so the **Section** label is used.
- If the HeadingNumber is 1 (a chapter number), then the test for the INT (1) < 1 is **false**, so the **Chapter** label is used.

However, the real hurdle occurs with Heading 3 which has 3 levels of numbers (e.g., 1.1.1). These do not represent a proper mathematical number that can be compared the same way. What happens is that Heading 3 numbers cause an error in the formula (which can’t understand the integer of a number like 1.1.1), and luckily, the error registers as a negative number (though not -1). When the INT of a multilevel number (like 1.1.1) causes an error in the formula, the error is swallowed by the conditional IF field in which it is contained. That is, it causes an internal syntax error that evaluates logically as a negative number and also does not create an error in the overall field results.

So the formula works the same way for Heading 3 numbers as for Heading 2 numbers; that is, the integer portion of a Heading 3 is a negative number, which is less than the full multilevel number (1.1.1).

The reason you can use a multilevel heading number (a non-mathematical number like 1.1.1) in the comparison is that a multilevel number is treated as an expression when it appears in an IF field condition (unless a math function is used on it). What happens is that the multilevel number is combined mathematically into a decimal number. Each level below the whole number is treated as a decimal portion and added to the total (there’s an assumed + operator before each decimal point). For example: 1.9.14 = 1 + .9 + .14 = 2.04. So the result of the following IF field is true: { IF 2.04 = 1.9.14 "True" "False" }. The multilevel number (1.9.14) can be used on either side of the expression.

As we’ve seen, when the multilevel number is an argument in the INT function, it is not combined into a decimal number and causes an error. So the left side of the comparison evaluates to a negative number (due to the INT function error) and the right side of the comparison evaluates to a decimal (due to the multilevel number being mathematically combined into a decimal result).

Test Case for Level 3:

> If the HeadingNumber is 1.1.1 (a Level 3 section number), then the test for the INT (1.1.1) < 1.1.1, is **true** (just like Level 2), because the test evaluates to IF (negative value) < 1.2, which is true, so the **Section** label is used.

In this test case, the INT of a multilevel number (1.1.1) causes a “!Missing Operator” error, which registers as a negative number on the left side of the comparison. On the right side of the comparison, the multilevel number evaluates to a decimal number, which is 1.2. The negative number is always less than the decimal number (1.2); therefore the statement is always true for multilevel numbers.

The level 3 test case is crucial because of its non-standard numeric format. The logic works because the multilevel heading number (e.g., 1.1.1) is treated as an implied expression on the right side of the comparison and an invalid numeric expression on the left side (due to the INT function).

This new discovery in multilevel number functioning is what allows this solution to work, a solution which is non-intuitive to say the least, which is probably why it hasn't been uncovered before.

Notes and Limitations

This solution enables 3 different labels, which is what most people would need. For more than this, see the 4- and 5-label solutions described under ***Extensions***.

The cross-reference fields can be manually built, but any sane person would want to use a macro to generate them. See ***Automation***.

The **heading number** cross-reference will not be hyperlinked, because it is output from within the IF field which makes the hyperlink inactive. If you were to encapsulate the set of cross-reference fields in a QUOTE field, it would also kill the hyperlinking for the **heading text** cross-reference.

While not a limitation of Microsoft Word, there is a bug in Adobe Acrobat 8 PDFMaker such that it does not produce the conditional output properly or may not produce output at all. Printing to the Adobe printer produces the proper results but, unlike PDFMaker, does not generate hyperlinks.

Although it does not cause a problem in functioning, on some infrequent occasions, Microsoft Word will surreptitiously add a MERGEFORMAT switch into the heading number REF fields that are used in the IF field condition. This does not have any effect on the fields, other than cluttering the field codes unnecessarily. For example, it will change { REF _Ref123456789 \n } to { REF _Ref123456789 \n * MERGEFORMAT }. This might occur after making cross-reference style and autotext field changes. You can remove the unwanted MERGEFORMAT switches from these fields if you want, but be careful not to delete all MERGEFORMAT switches indiscriminately.

Applicability

This solution has been tested on Microsoft Word 97, 2000, XP (2002), 2003 and Microsoft Windows XP. (Microsoft Word 95 does not support outline numbering or the PAGEREF \p switch, but the fields do not cause an error.)

Customizations

The solution can be customized to meet your cross-reference structure and formatting requirements, while automating the creation and insertion of the cross-reference allows you to take full advantage of the solution.

Structure and Formatting

The following information describes the structure and formatting that can be used for conditional cross-references. The structure you choose is up to you, so you can add or re-arrange the punctuation or change the styles and label text.

Use Styles and Autotext

Applying **character styles** helps to distinguish the components, and the character formatting can replace the use of quotation marks surrounding the reference.

Using character styles also enables conditionalizing of the components so that any unwanted components (e.g., heading label, heading number, and page reference) can be excluded (e.g., for repurposing the document to create on-line Help). You can conditionalize the components by setting the character styles to hidden text or visible text (to turn them on/off) or through single sourcing software that enables conditional styles.

Using **autotext fields** for the supporting text, including the labels, punctuation and delimiters, allows you to maintain the text in one place so that you can change the text globally if necessary. It also helps to establish and "**enforce**" a cross-reference structure so that writers do not introduce unnecessary variations among the labels and text. Character styles can be applied to the text before it is saved in autotext, which makes it unnecessary to add styles to the autotext fields in the cross-reference. Autotext can also be used to turn particular labels on or off (see below).

One consideration for using autotext in the cross-reference is that a document template is then required because autotext is stored in a template or add-in.

Controlling the Presence of Labels

To turn particular **labels on or off**, do the following:

- If you use an **autotext** solution, apply a hidden style or direct formatting to the label text and save it over the old autotext entry. For example, to turn off the Topic label, hide the text (Topic:) by applying a style or direct formatting to it and then saving it with the same autotext name (xrLabel_0Topic), so it replaces the old autotext entry.
- If you use don't use autotext (i.e., if you use a **literal text** solution), you must create separate styles for each label. The solutions use a single character style (cXrefLabel) for all the labels (e.g., Chapter, Section, Topic). If you want more control over the individual labels, you can create and use individual styles for them. For example, you could create a cXrefLabelTopic style to enable independent control over the Topic label (in case you want to turn it off and leave the other labels on).

Formatted-Style Cross-Reference

A **formatted-style cross-reference** uses character formatting to set the cross-reference apart from the surrounding context (e.g. body text). For example, the following cross-reference uses a sans serif font (e.g., Arial), which works well when the body text is a serif font (e.g., Times). If you select this type, make sure the cross-reference and body text typefaces mesh well together so that the cross-reference doesn't looked outsized. It shows one possibility for the structured cross-reference format. (This cross-reference format is used in the Sample Document.)

See *Chapter: 1.* ***Introduction*** on page 1.

The components of this structured cross-reference are as follows:

See *Label HdgNumber HnumDelimiter* ***HeadingText*** PageReference.

The following character styles are applied to the components: *cXrefLabel*, *cXrefHnum*, *cXrefHnumDelimit*, ***cXrefHdgTxtLink***, and cXrefPgRef.

The field coding (with autotext fields) for this structured cross-reference is as follows:

```
See { IF { REF _Ref123456789 \n } = 0 "{ AUTOTEXT xrLabel_0Topic }" "{ IF { = INT
( { REF _Ref123456789 \n } ) } < { REF _Ref123456789 \n } "{ AUTOTEXT
xrLabel_2Section }" "{ AUTOTEXT xrLabel_1Chapter }" }{ REF _Ref123456789 \n \*
MERGEFORMAT \* CHARFORMAT }{ AUTOTEXT xrHnum_Delimiter }" }{ REF
_Ref123456789 \h \* MERGEFORMAT \* CHARFORMAT } { PAGEREF
_Ref123456789 \p \h \* MERGEFORMAT \* CHARFORMAT }.
```

The *italic sans serif* typeface sets the cross-reference off from the surrounding text. The ***blue bold sans serif*** typeface shows the cross-reference hyperlink while allowing users to print the document with good results without setting "print all colors as black."

The MERGEFORMAT and CHARFORMAT field switches keep the character style applied to the fields. The CHARFORMAT is especially necessary for the hyperlinked heading text. It ensures the character style is applied across the entire heading text. It is necessary because if you change the text of a heading and update the cross-reference, the character style sometimes gets lost from part of the heading text.

I leave the page reference as an active hyperlink so it can be used, but I do not mark it with special character formatting because I don't feel it is necessary to highlight as a hyperlink. It jumps to the same location as the heading text hyperlink next to it, and I prefer to leave its appearance undisturbed, which results in a visually cleaner cross-reference (it looks less busy). However, I apply a "transparent" (default font) character style to it anyway, in case a formatting requirement arises in the future or for conditionalizing the styled content with hidden text or through single sourcing software.

It is better to leave the introductory phrase (e.g., "See" or "For details, see…") unenforced by the structured cross-reference to allow the cross-reference to be used in non-sentence constructions.

Quotation-Style Cross-Reference

The structured cross-reference must be designed to mesh with all cases where it is used in a document. This means you have to carefully consider all cases and how the structured cross-reference would fit them.

One typical construction of a cross-reference is to use quotation marks to set it off from the surrounding text (a **quotation-style cross-reference**). The quotation marks could be applied to the entire cross-reference or to just parts of it and may be used in conjunction with other punctuation. Examples:

> See "Chapter 1: Introduction" on page X.
>
> See Chapter 1, "Introduction" on page X.

However, the quotation-style cross-reference can be complicated, and there is a problem with creating this type as a structured cross-reference in which all the elements are standardized and enforced for consistency. One reason is that the structured cross-reference cannot have a terminating period located within it. When the cross-reference is used in a sentence (as contrasted with a simple list), some text must follow the cross-reference and appear before the concluding period of the sentence. The reason is that if the period were to appear inside the closing quotation marks, which is standard editorial style, the cross-reference would always have a period. This would be undesirable in some cases, such as a topic list, which is usually introduced by a phrase (e.g., See the following information) and is followed by a simple list (usually without page references), like this:

> "Chapter 1: Introduction"
>
> "Section 1.1: Features"

In short, you could:

- Accept trailing punctuation outside the closing quotation mark, which is non-standard editorial style.
- Always have intervening text between the cross-reference and the end of the sentence. (The page reference qualifies for this, if it is **always** present.)

It can be used, but I think the quotation-style structured cross-reference is awkward. That is why I prefer the cross-reference style that uses formatting to distinguish it from surrounding text.

Short Form

If you want to offer the option to insert only the label and heading number (e.g., Chapter 1) as a **short form** of the cross-reference, you must design your cross-reference structure and your macro to do this. Two options are as follows:

- Continue to insert the label and heading number as a "linked" pair (within the conditional statement the way they are done now), in which case you must delink (or remove) the heading number delimiter (period-space-space) from it. You can do this by turning it off (setting the cXrefLabel style to hidden text), or simply deleting it and using an unconditional heading number delimiter that is added only when the cross-reference is followed by heading text. The drawback with this option is that it does not offer built-in hyperlinking.
- Delink the label from the heading number, so that both the heading number (and optional heading number delimiter) can be output outside the condition. This offers the benefit of being able to add a hyperlink switch (and hyperlink formatting) to the heading number when it is used in this short form. But the drawback, which outweighs the benefit, is that they could then get separated from each other during editorial work by a writer, such that the labels accidentally end up with the wrong heading numbers. (*Horrors!*)

In my opinion, it is always better to keep the label and heading number linked, so the first option would be more desirable.

Autotext Formatting Precedence

In Microsoft Word, the formatting of the **template autotext entry** is carried into the document, but if the document overrides it with formatting applied to the **document autotext field**, the formatting may or may not come through. The order of precedence depends on how the formatting is applied (either by style or direct formatting) and whether it is local or remote. (**Remote** refers to formatting saved in the template autotext entry. **Local** refers to formatting applied onto the document autotext field.)

- **Remote direct formatting** takes precedence over everything (local style or local direct formatting).
- **Local direct formatting** (override) takes precedence over remote style.
- **Local style** (override) takes precedence over remote (template) style (with same or different name).

In the 3-, 4-, and 5-label solutions, the conditional cross-references that use autotext have remote styles for autotext entries and no local autotext field formatting. (Styles are applied to the autotext entries in the Sample Autotext Template, and no styles or direct formatting are applied to the cross-reference fields that get inserted into the document.) This means you can change the formatting of the autotext entry in the template, either by applying styles or direct formatting, and it will be carried into the documents attached to that template.

Automation

After you have successfully built and tested your own prototype conditional cross-reference (or copied one from the Sample Document provided under ***Downloads***), you will need a way to insert it into your documents so that it refers to a specific heading in the document. The best way is to use a macro. If you need an interim macro solution because you don't have the time or means to write full macro code, you can insert them by following a few simple steps, described here. Basically, you will be replacing the bookmark name in the prototype cross-reference with a live cross-reference.

- You may want to rename the heading bookmark name in your prototype conditional cross-reference fields to give it a unique pattern, such as "_RefXYZ", which will be used as a search target. (This is not required but it may make things a little easier.)
- Save the prototype conditional cross-reference fields as an autotext entry in your document template. This enables you to insert these fields in documents quickly and easily. (See LabeledXrefsAutotext.dot for an example.)

Then, anytime you want to build a "live" conditional cross-reference from the prototype, do the following:

1. Insert a live cross-reference to a document heading in any form (heading text, heading number, page number, etc.). This creates a heading bookmark, and inserts a REF field, which refers to the bookmark.
2. Insert the cross-reference prototype from the autotext.
3. Select the prototype and the live cross-reference fields, and reveal their field codes by pressing Shift-F9.
4. Select and copy the bookmark name from the live cross-reference field to the clipboard.
5. Select the prototype cross-reference fields, which are exposed, and enter their bookmark name as the search target (e.g., "_RefXYZ"). Enter ^c as the replacement

value, which specifies the clipboard contents. Replace all instances of the bookmark name in the field.

6. Press F9 to update the field, and . . . *Voila!* You have a live conditional cross-reference.
7. Delete the cross-reference you inserted in Step 1.

No one wants to repeat these steps on a regular basis. You could record a simple macro to automate steps 2-7, but the best answer is to automate them fully by writing macro code, which is beyond the scope of this article.

Extensions

The solution can be extended to accommodate 4-label and 5-label conditional cross-references.

The 3-label solution uses the "Section" label for all numbered subsections (levels 2 and 3 in the example). This is a very common convention. If, for some reason, you needed to identify levels 2 and 3 (or higher) with different labels (e.g., Section, SubSection), you could do so using the following extensions.

The 4-label extension adds one additional labeling option (e.g., SubSection) and the 5-label extension adds 2 additional labeling options (e.g., SubSection, Sub-SubSection).

The 4-label extension is based on the same outline numbering plan as the 3-label solution (Headings 1-3 are numbered), whereas the 5-label extension requires one additional numbering level (Headings 1-4 are numbered).

In general, it is unnecessarily complicated to use more than 3 levels of numbered headings in technical documents, and if possible, it should be avoided for improved readability and usability of the document. However, if it is necessary, the solution can be extended to accommodate these cases. The following table shows the label types that can be used in the 3-, 4-, and 5-label solutions.

Table 1—Conditional Labeling Capabilities

Level	Label Type	Heading Number	Used in 3-Labels	Used in 4-Labels	Used in 5-Labels
(x-9)**	Topic	0 (none)	Y	Y	Y
1	Chapter	1	Y	Y	Y
2	Section	1.1	Y*	Y	Y
3	SubSection	1.1.1		Y*	Y
4	Sub-SubSection	1.1.1.1			Y*

*The lowest-level label in each label category is also applied in any cross-references to numbered headings below that level. For example, if Headings 5-9 were numbered, they would receive the lowest-level label for that category (in the 3-label category this would be "Section").

**(x-9): The Topic label applies to levels x-9 (e.g., Headings 4-9, 5-9, or any other variation), because it applies to any non-numbered heading.

The extra labeling options provided by the extensions can offer the flexibility for adoption in the future while using a 3-label result in the present. That is, the 4- or 5-label solution can easily be used to accomplish a 3-label result simply by setting the lower level labels to match the level 2 label.

This allows you to select a solution for future expansion. For example, if you think you might want 4 different labels in the future although you only use 3 now, you can use the 4-label solution and set level 3 (SubSection) to match level 2 (Section). You can do this in the autotext by saving the Section label as the autotext entry for the SubSection label (xrLabel_3SubSection), replacing the existing one. That way, a Section label will be applied to both Headings 2 and 3 in the 4-label solution, and can be changed back later if you need it. A similar approach can be used with the 5-label solution, but then most people will never need this many labels.

IF Field Extensions (Wildcards)

The extensions build upon the logic of the basic solution.

The extensions employ **wildcard** characters for the logical comparison of strings in the IF field. The only wildcards supported by the IF field are the question mark (?) and asterisk (*) characters. The question mark matches any single character, and the asterisk matches any number of characters. Any wildcards must be specified in a string on the right side of the comparison, and the only logical operators that can be used with wildcard strings are the equal to (=) and not equal to (<>) operators. Also, when using the asterisk wildcard, the length of the matching string plus the length of the wildcard string must not exceed 128.

The 4-label extension uses the asterisk wildcard in a comparison. The 5-label solution adds the question mark wildcard.

The 4-Label Extension

Level 3 numbers (e.g., 1.1.1) cause a “!Missing Operator” error in the INT function. The 4-label extension takes advantage of this by adding an error trap to distinguish between level 2 numbers that don’t cause an error (e.g., 1.1) and level 3 numbers that do. This error test adds an additional level of nesting to the conditional logic.

By this means, another label can be introduced, resulting in 4 possible labels (e.g., Chapter, Section, SubSection, Topic) instead of the usual 3 (Chapter, Section, Topic). When the new label is used in cross-references, it appears similar to the following example, which shows a Heading 3 cross-reference:

> See SubSection: 1.1.1. Standard Features on page X.

The error string can be identified by the following test:

```
IF the INTeger portion of the HeadingNumber = "!Missing Operator" THEN
    It is a multilevel number x.x.x or longer
```

However, it is shorter and simpler to use the asterisk wildcard in the test, as follows:

```
IF the INTeger portion of the HeadingNumber = "!*" THEN
    It is a multilevel number x.x.x or longer
```

Field Coding for 4 Conditional Labels

The cross-reference field coding for the 4-label solution is as follows (without any character formatting or formatting switches):

```
See { IF { REF _Ref123456789 \n } = 0 "Topic: " "{ IF { = INT ( { REF _Ref123456789
\n } ) } = "!*" "SubSection: " "{ IF { = INT ( { REF _Ref123456789 \n } ) } < { REF
_Ref123456789 \n } "Section: " "Chapter: " }" }{ REF _Ref123456789 \n }.  " }{ REF
_Ref123456789 \h } { PAGEREF _Ref123456789 \p \h }.
```

Note: Pre-built field codes are available in the Sample Document and the Sample Autotext Template.

The Conditional Logic for 4 Labels

The pseudocode for the 4-label solution is as follows (the 4-label extension is separated from the 3-label logic by blank lines and is marked on the left with vertical bars):

```
IF no heading number (HeadingNumber = 0) THEN
    Use Topic Label
ELSE

|   IF a subsection number x.x.x or longer (INT function error = "!*") THEN
|       Use SubSection Label
|   ELSE

        IF a section number x.x (INT < HeadingNumber) THEN
            Use Section Label
        ELSE
            Use Chapter Label (HeadingNumber > 0)
        [ENDIF]
|   [ENDIF]
    Output the HeadingNumber + Delimiter (for numbered headings)
[ENDIF]
Output the (unconditional) HeadingText + PageReference
```

Field Coding for 4 Labels Showing Logical Structure

The following field coding for the 4-label solution shows the logical structure of the conditions in parallel with the pseudocode.

Line breaks were added within the field codes to show the logical structure. They are used for purposes of illustration only. The following shows how the implied elements can be formatted to reveal the standard programming structure.

IF is explicit.
" THEN is implied by leading quotes for TrueText.
" ELSE is implied by leading quotes for FalseText.
} ENDIF is implied by the IF EndOfFieldDelimiter (ClosingFieldBrace).

For example, the following format reveals the logical structure of a simple IF statement:

```
{ IF Condition " [THEN]
    Output A"
" [ELSE]
    Output B"
} [ENDIF]
```

The field format with line breaks is as follows (with vertical bars added on the left to mark the differences from the 3-label solution):

```
{ IF { REF _Ref123456789 \n } = 0 "
    Topic: "
"
|   { IF { = INT ( { REF _Ref123456789 \n } ) } = "!*" "
|       SubSection: "
|   "
        { IF { = INT ( { REF _Ref123456789 \n } ) } < { REF _Ref123456789 \n } "
            Section: "
        "
            Chapter: "
        }"
|   }
    { REF _Ref123456789 \n }.  "
}
{ REF _Ref123456789 \h } { PAGEREF _Ref123456789 \p \h }.
```

The 5-Label Extension

By the following means, another label can be introduced, resulting in 5 possible labels (e.g., Chapter, Section, SubSection, Sub-SubSection, Topic). When the new label is used in cross-references, it appears similar to the following example, which shows a Heading 4 cross-reference:

See Sub-SubSection: 1.1.1.1. Hardware on page X.

First of all, to use a fifth level of conditional labeling, you need 4 levels of numbered headings (e.g., 1.1.1.1 for Heading 4), which together with the non-numbered headings, makes 5 levels.

However, extending the solution beyond 4 labels gets tricky. By combining the 4-label method with an additional technique, you can achieve 5 labels, but it is more complicated and has limitations.

To achieve 5 labels, we need to distinguish between level 3 numbers (e.g., 1.1.1) and level 4 numbers (e.g., 1.1.1.1). But we don’t need to test for both Heading 3 and Heading 4 numbers, because if we test for just one level, we can exclude the other level, and simply handle them through the FalseText (or ELSE) part of the condition. So by identifying level 3 numbers, the level 4 numbers will automatically fall under the spell of the FalseText logic.

One way to identify the level 3 numbers is to use the “?” wildcard, which matches any single character. A comparison string can be created using periods and question marks to match level 3 numbers. For example, the wildcard string “?.?.?” will match heading number 1.1.1 as well as 1.1.2, all the way up to 9.9.9. The pseudocode for this is:

```
IF HeadingNumber = "?.?.?" THEN
```

When using a wildcard string comparison, the wildcard pattern must be placed on the right side of the comparison. The multilevel heading number on the left side of the comparison is not combined mathematically into a decimal number, so decimal place addition does not occur in this case. This means that multilevel numbers can be evaluated by wildcard patterns.

The only trouble is that the wildcard pattern must match all combinations of level 3 numbers. When the number of headings at a given level reaches 10, the “?.?.?” pattern will no longer match. Two-digit heading numbers increase the complexity, and 3-digit heading numbers even more so.

Using the ? wildcard means testing for each possible combination of digit-places and heading level (decimal) places. So a greater number of tests is required as the number of headings increases to the next power of 10 as well as for each additional numbered heading level that you want to be uniquely-labeled.

It would be inefficient and impractical to use wildcard patterns that would handle all heading number combinations. But, if you knew that there would never be more than 99 headings per level, it would limit the complexity greatly.

We can rely on the 4-label test to handle wildcard patterns for levels 1 and 2. And if we limit the number of headings to 99, we only need to test combinations up to 2 digits for each level of the Heading 3 number. This would require only (2^3) = 8 combinations. The Heading 3 combinations are (using 1 as the placeholder for easier reading):

1.1.1
1.1.11
1.11.1
11.1.1
11.11.1
11.1.11
1.11.11
11.11.11
Total: 8 combinations.

So we will test for Heading 3 using the “?” wildcard in the following 8 patterns:

?.?.?
?.?.??
?.??.?
??.?.?
??.??.?
??.?.??
?.??.??
??.??.??

The wildcard testing adds a fourth level of nesting to the conditional logic and has 8 instances (one for each pattern). However, rather than repeating the output labels at each instance, we can greatly simplify it by simulating a logical OR to determine if any of the patterns match before the labels are output.

This is done by testing each wildcard pattern and setting the TrueValue to 1 and the FalseValue to 0. Each test result is then added together and if the total result equals 1 (or the total result is greater than 0), then the heading is a level 3 number and the label is output (e.g. SubSection). Otherwise, the level 4 label is output (e.g., Sub-SubSection).

If the heading number exceeds the provisional limits of more than 99 headings per level or more than 3 levels of numbering, the cross-reference will use the lowest label from the catch-all condition (e.g., Sub-SubSection).

Thanks to Word MVP Graham Mayor for information on his website that helped me create the 5-label solution. Also thanks to Paul Edstein (pseudonym Macropod) for his amazing Word Field Maths document, available on Graham Mayor’s website at http://gmayor.com/downloads.htm#Third_party.

Field Coding for 5 Conditional Labels

The cross-reference field coding for the 5-label solution is as follows (without any character formatting or formatting switches):

```
See { IF { REF _Ref123456789 \n } = 0 "Topic: " "{ IF { = INT ( { REF _Ref123456789
\n } ) } = "!*" "{ IF { = ( { IF { REF _Ref123456789 \n } = "?.?.?" 1 0 } + { IF { REF
_Ref123456789 \n } = "?.?.??" 1 0 } + { IF { REF _Ref123456789 \n } = "?.??.?" 1 0 }
+ { IF { REF _Ref123456789 \n } = "??.?.?" 1 0 } + { IF { REF _Ref123456789 \n } =
"??.??.?" 1 0 } + { IF { REF _Ref123456789 \n } = "??.?.??" 1 0 } + { IF { REF
_Ref123456789 \n } = "?.??.??" 1 0 } + { IF { REF _Ref123456789 \n } = "??.??.??" 1
0 } ) } = 1 "SubSection: " "Sub-SubSection: " }" "{ IF { = INT ( { REF _Ref123456789
\n } ) } < { REF _Ref123456789 \n } "Section: " "Chapter: " }" }{ REF _Ref123456789
\n }.  " }{ REF _Ref123456789 \h } { PAGEREF _Ref123456789 \p \h }.
```

Note: Pre-built field codes are available in the Sample Document and the Sample Autotext Template.

The Conditional Logic for 5 Labels

The ? wildcard tests are added under the subsection number test (the INT function error = “!*”).

The pseudocode for the 5-label solution is as follows (the 5-label extension is separated from the 4-label logic by blank lines and is marked on the left with vertical bars):

```
IF no heading number (HeadingNumber = 0) THEN
    Use Topic Label
ELSE
    IF a subsection number x.x.x or x.x.x.x or longer (INT function error = "!*") THEN

|       IF a subsection number (= x.x.x wildcard patterns) THEN
            Use SubSection Label (x.x.x)
|       ELSE
|           Use Sub-SubSection Label (x.x.x.x or longer)
|       [ENDIF]

    ELSE
        IF a section number x.x (INT < HeadingNumber) THEN
            Use Section Label
        ELSE
            Use Chapter Label (HeadingNumber > 0)
        [ENDIF]
    [ENDIF]
    Output the HeadingNumber + Delimiter (for numbered headings)
[ENDIF]
Output the (unconditional) HeadingText + PageReference
```

Field Coding for 5 Labels Showing Logical Structure

The following field coding for the 5-label solution shows the logical structure of the conditions in parallel with the pseudocode.

Line breaks were added within the field codes to show the logical structure. They are used for purposes of illustration only. The following shows how the implied elements can be formatted to reveal the standard programming structure.

```
IF is explicit.
    " THEN is implied by leading quotes for TrueText.
    " ELSE is implied by leading quotes for FalseText.
} ENDIF is implied by the IF EndOfFieldDelimiter (ClosingFieldBrace).
```

For example, the following format reveals the logical structure of a simple IF statement:

```
{ IF Condition " [THEN]
    Output A"
" [ELSE]
    Output B"
} [ENDIF]
```

The field format with line breaks is as follows (with vertical bars added on the left to mark the differences from the 4-label solution):

```
{ IF { REF _Ref123456789 \n } = 0 "
    Topic: "
"
    { IF { = INT ( { REF _Ref123456789 \n } ) } = "!*" "
|       { IF { = (
|           { IF { REF _Ref123456789 \n } = "?.?.?" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "?.?.??" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "?.??.?" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "??.?.?" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "??.??.?" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "??.?.??" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "?.??.??" 1 0 } +
|           { IF { REF _Ref123456789 \n } = "??.??.??" 1 0 }
|       ) } = 1 "
            SubSection: "
|       "
|           Sub-SubSection: "
|       }"
    "
        { IF { = INT ( { REF _Ref123456789 \n } ) } < { REF _Ref123456789 \n } "
            Section: "
        "
            Chapter: "
        }"
    }
    { REF _Ref123456789 \n }.  "
}
{ REF _Ref123456789 \h } { PAGEREF _Ref123456789 \p \h }.
```

More Is Possible

Expanding the solution further is possible but the tests will grow significantly larger. I will touch briefly on 2 examples that demonstrate how any further expansions will increase the size of the solution to the nth degree.

As a reminder, the 5-label solution relies upon the requirement that the number of headings for each level of Headings 1-3 will never exceed 99. That is, the highest heading number it can handle is 99.99.99. If you want up to expand this up to 999 headings per level, it would take (3^3) or 27 different wildcard combinations rather than the 8 used in the 5-label solution. Let’s just say: *Fuggedaboudit*.

If you were truly heading-number crazy and wanted to expand this solution to include labels for yet another numbered heading level; e.g., 1.1.1.1.1 (Heading 5), while keeping the heading limit of 99, you could accomplish it with “only” (*haha*) 24 different wildcard combinations rather than the 8 used in the 5-label example. This is because you wouldn’t have to test for the (2^5) or 32 combinations for Heading 5, but could test for the 8 combinations for Heading 3 (as in the example shown) plus the 16 combinations for Heading 4, relying on the ELSE part of the condition to handle the 32 combinations for Heading 5 or greater. Let's just say: ***I'm** not going there*.

Now that technical writers have this new solution for conditional cross-reference labels in hand, they can turn their sights toward simpler problems, like single sourcing the world-wide web, or even better, solving global warming.