HotDocs Models

Copyright

Table of Contents

Introduction: Create HotDocs Models

As you draft documents for your clients, you frequently modify existing documents by replacing the changeable text in the document (such as names, dates, and figures) with the current client's information. This process can be time-consuming and error-prone, especially because you must go through this process for each new client or matter.

One way you can solve this problem is to create a HotDocs template from the document. This process requires someone with HotDocs knowledge to replace changeable text with variables, conditions, and scripts. Depending on how complex the template is, sometimes the automated template can be difficult for non-HotDocs developers to read and understand.

Another option is to instead create a *HotDocs Model*. HotDocs Models are ideal for subject matter experts who want to review a template without having to learn or install HotDocs on their local machine. The variable and instructions are clearly visible in the document text and can be color-coded to differentiate between types (see Apply Color to Markup). This makes the entire template process easily accessible for the non-HotDocs user and allows them to edit the HotDocs Model before passing it back to their HotDocs developer.

Once you have a HotDocs Model, you can use it with HotDocs to generate documents for your clients. If you are using HotDocs Developer, you can also convert your HotDocs Models to HotDocs template format so they can be added to your collection of templates.

For example, say you have a will that could include different provisions for different inheritance scenarios. Some provisions should be included if you're drafting a will for one type of client, but those same provisions should be left out for other types of clients. With a HotDocs Model, you keep all of the different text variations in a single document, rather than storing the text in separate documents. Then you can mark up the document so that only the correct versions are used, depending on answers the user gives.

As you mark up the HotDocs Model directly in Microsoft Word, your documents must be in DOCX, DOC or RTF format.

Creating a Simple HotDocs Model

Basic Markup

A HotDocs Model is an exemplar document used as the basis for drafting documents of the same type for specific clients. The markup used in HotDocs Models to identify text that varies from one client to another provides an unambiguous description of how a HotDocs Model should be used to draft client-specific documents.

This topic describes only the basics of Markup. For a complete description, see the section of this help file entitled "Creating Full HotDocs Models".

The process of marking up a HotDocs Model can be broken into several steps, as follows:

1. Mark Variable Text
2. Mark Conditional Text
3. Mark Repeated Text
4. Mark Special Characters
5. Hide Markup Clutter
6. Apply Color to Markup

Next Topic

Previous Topic Next Topic

Mark Variable Text

You can replace names, dates, amounts, and other variable text throughout a document with variable fields. A variable field consists of a name and type, separated by a semicolon and enclosed in square brackets. For example, the document text:

```
The client, Jane Doe, hereby rescinds all previous claims.
```

is marked up as:

```
The client, [ClientName;te], hereby rescinds all previous claims.
```

In this example, **ClientName** is the variable name and **te** (which stands for text) is the type.

Variable names:

- Can include letters, numbers, and underscores but they cannot contain spaces or other special characters.
- Are case-sensitive (meaning that **GiftAmount** and **giftamount** are different names).
- Must not be composed entirely of capital letters.
- Must be 50 characters or less in length.

In general, a variable name consists of at least two words that identify the subject and the aspect of the field. Best practice is to capitalize the first letter of each word in a variable name, regardless of its part of speech. Some examples:

ClientCity	**Client** is the subject and **City** is the aspect.
ClientZipCode	**Client** is the subject and **ZipCode** is the aspect.
AgreementEffectiveDate	**Agreement** is the subject and **EffectiveDate** is the aspect.
TermOfAgreement	**Agreement** is the subject and **TermOf** is the aspect.

Variable field types are specified using the first two letters of the word that represents the type:

- **te** for text fields
- **nu** for number fields
- **da** for date fields

When no type is specified in a field (for example, **[ClientName]**), text is assumed as the type.

Variable fields may optionally include a format following the field type. For example, if you want the client's name to be inserted into the document in all uppercase letters, use:

```
The client, [ClientName;te;format=upper], hereby rescinds all previous claims.
```

Such markup would produce the following in a completed document:

```
The client, ANNA JAMES, hereby rescinds all previous claims.
```

In the example field above, notice the semicolon between the field type and the word **format**. Two commonly used formats are:

- **upper** forces alphabetic characters to uppercase
- **alpha** spells out a number instead of using digits

For a complete list of Markup formats, see Format Answers in the Document.

Previous Topic Next Topic

Previous Topic Next Topic

Mark Conditional Text

Text in a document is conditional if it is included only when certain conditions are met. To mark conditional text, enclose the text in square brackets. Immediately following the opening bracket, type the word **if** followed by a colon. After the colon, type a condition name followed by another colon. For example:

```
[if:ClientIsMarried:I give to my spouse all interest in property, both real and
personal, including buildings, fixtures and improvements to such property.]
```

A condition name is an affirmative statement specifying the conditions under which the conditional text is included in the document. All of the rules that apply to variable field names apply to condition names.

When conditional text consists of a complete paragraph, it is usually desirable to include the paragraph mark as part of the conditional text. To do this, type the closing bracket after the paragraph mark, such that it appears at the beginning of the following paragraph. For example:

```
My name is [ClientName;te]. ¶

[if:ClientIsMarried:I am married to [SpouseName;te]. ¶

][if:ClientHasChildren:My children are ¶ [ChildrenNamesAndBirthdates;te]. ¶

]Except as specifically noted in this document … ¶
```

Conditional text fields may be "nested" within other conditional text fields. For example:

```
[if:ClientIsMarried:I give to my spouse all interest in property, both real and
personal, including buildings

[if:GiftsIncludeFixtures:, fixtures,] and improvements to the property.]
```

Conditional text fields may include one or more alternatives, only one of which will be included in the final document. To specify alternatives, introduce each alternative with a forward slash (/) character followed by **elseif** or **else**. Use **elseif** if the alternative is conditional. Use **else** if the alternative is conditionless, meaning that it is included in the final document when no other alternative is included.

Here is an example showing several conditional alternatives followed by a conditionless alternative:

```
I, [ClientName;te;format=upper], direct my personal representative to see that
my body is [if:ClientToBeBuried:buried in the
[ClientBurialCemetery;te]/elseif:ClientToBeCremated: cremated and my ashes
scattered whithersoever my personal representative shall
direct/elseif:ClientBodyToMedicine:donated to
[ClientBodyInstitution;te]/else:disposed of as my personal representative shall
direct].
```

A conditionless alternative (**else**) is not required, but if present, it must follow all conditional alternatives.

When a HotDocs Model includes large sections of conditional text or nested conditional text, it can be difficult to find the beginning and ending of conditional text fields. If you have installed the HotDocs Markup Tools, use the **Select Field** button to easily see where the conditional text field begins and ends. (Click once to select or highlight the immediate condition, and click twice to view the entire condition.)

Previous Topic Next Topic

Previous Topic Next Topic

Mark Repeated Text

Repeated text is text that can appear in a document two or more times in succession. Repeated text can consist of anything from a few words to an entire paragraph. To mark repeated text, enclose a single instance of the text in square brackets. Immediately following the opening bracket, type the word **repeat** followed by a colon (:). After the colon, type a repeat dialog name followed by a semi-colon and the list of variables or text you want included in the repeat instruction, separated by forward slashes.

The following is an example of repeated in-line text:

```
My children are: [repeat:ChildList;format="a; b; and
c";contents=ChildName/ChildBirthDate:[ChildName], born [ChildBirthDate;da]].
```

In this example, the entire paragraph is repeated:

```
Specific Gifts

[repeat:SpecificGiftList;contents=SpecificGiftRecipientRelationship/SpecificGif
tRecipientName/SpecificGiftItem: To my [SpecificGiftRecipientRelationship],
[SpecificGiftRecipientName], I give [SpecificGiftItem].

]
```

In general, a repeat dialog name consists of at least two words, the first of which identifies the subject of the list. Best practice is to include the word List as part of a repeat dialog name, although this is not mandatory. All of the rules that apply to variable names apply to repeat dialog names.

Previous Topic Next Topic

Previous Topic Next Topic

Mark Special Characters

As you are marking up your document, you may come across characters in your document text that also have a purpose in Markup. Specifically, Markup attaches special meaning to square bracket, forward slash, and backslash characters. If any of these characters are to appear as literal text in the final document, they must be marked in the HotDocs Model to show that they are just text, rather than part of the Markup.

To mark a special character, type a backslash before it in the document. So:

- [is marked as \[
-] is marked as \]
- / is marked as \/
- \is marked as \\

For example, this text in a HotDocs Model:

```
For a complete description, see H:\\Publications\\Complete Description.doc
\[revised 2006\]
```

appears like this in the final document:

```
For a complete description, see H:\Publications\Complete Description.doc
[revised 2006]
```

Here are two reasons for marking special characters in HotDocs Models: First, macros used to manipulate Markup in HotDocs Models rely on the proper use of special characters. Unmarked special characters appear to be part of the Markup, causing incorrect macro behavior. Second, document drafting applications (such as HotDocs) that accept HotDocs Models as input cannot operate properly if special characters are not marked.

Previous Topic Next Topic

Previous Topic Next Topic

Hide Markup Clutter

To make a HotDocs Model easier to read, you can hide the type and format information for variable fields and the condition information for conditional text fields. With this “clutter” hidden, this conditional example:

```
I, [ClientName;te;format=upper], direct my personal representative to see that
my body is [if:ClientToBeBuried:buried in the
[ClientBurialCemetery;te]/elseif:ClientToBeCremated: cremated and my ashes
scattered whithersoever my personal representative shall
direct/elseif:ClientBodyToMedicine:donated to
[ClientBodyInstitution;te]/else:disposed of as my personal representative shall
direct].
```

becomes:

```
I, [ClientName], direct my personal representative to see that my body is
[buried in the [ClientBurialCemetery]/cremated and my ashes scattered
whithersoever my personal representative shall direct/donated to
[ClientBodyInstitution]/disposed of as my personal representative shall
direct].
```

Notice that in variable fields, markup from the colon up to the closing bracket is hidden. In conditional text fields, markup after the opening bracket or forward slash up to the first character of the document text is hidden.

To hide markup clutter using the built-in features of Word, select (or highlight) the text to be hidden, choose **Format** on the Word menu bar, choose **Font** in the drop-down menu, and select **Hidden**. To show hidden markup clutter for review or editing, click the **Show/Hide ¶** button on the Word toolbar.

If you have installed the HotDocs Markup Tools, you can first click the  **Apply Clutter** button on the toolbar to apply the hidden attribute to markup clutter across the entire HotDocs Model. Then, to easily show or re-hide the clutter you have hidden, click the  **Hide/Show Clutter** button.

Using the  **Hide/Show Clutter** button instead of **Show/Hide ¶** has the advantage that it shows hidden markup clutter without showing Word formatting marks which can make the HotDocs Model difficult to read.

Previous Topic Next Topic

Previous Topic

Apply Color to Markup

HotDocs Models often use blue to identify variable fields, green to identify conditional text fields, and maroon to identify repeat fields. While color is not mandatory, it is much easier to see markup in a HotDocs Model when color has been applied.

To apply color using the built-in features of Word, select (or highlight) the text to be colored, choose **Format** on the Word menu bar, choose **Font** in the drop-down menu, and select the font color. Or, after selecting text, click the **Font Color** button on the Word toolbar.

If you have installed the HotDocs Markup Tools, you can click the  **Apply Color** button on the HotDocs Markup Toolbar to apply the standard markup colors used in this document to all variable fields and conditional text fields in a HotDocs Model.

Previous Topic

Creating a Full HotDocs Model

1. Preparing Your Documents

Identify a HotDocs Model

A HotDocs Model is one in which the text of the document changes, depending on the user. Frequently, a HotDocs Model also contains variations of text, which you can change, add, or remove, depending on the user. Examples of HotDocs Models include wills, legal filings, real estate contracts, and so forth.

For example, say you have a will that could include different provisions for different inheritance scenarios. Some provisions should be included if you're drafting a will for one type of client, but those same provisions should be left out for other types of clients. With a HotDocs Model, you keep all of the different text variations in a single document, rather than storing the text in separate documents. Then you can markup the document so that only the correct versions are used, depending on answers the user gives.

You mark up HotDocs Models directly in Microsoft Word so your documents must be in DOCX, DOC, or RTF format.

The Markup described in this help file requires the use of square bracket, forward slash, and backslash characters. If a HotDocs Model includes any of these special characters, you must mark them to show they are intended to appear as literal text in the document rather than as markup characters. See Mark Special Characters for instructions on doing this.

Next Topic

Previous Topic Next Topic

Mark Special Characters

As you are marking up your document, you may come across characters in your document text that also have a purpose in Markup. Specifically, Markup attaches special meaning to square bracket, forward slash, and backslash characters. If any of these characters are to appear as literal text in the final document, they must be marked in the HotDocs Model to show that they are just text, rather than part of the Markup.

To mark a special character, type a backslash before it in the document. So:

[is marked as \[

] is marked as \]

/ is marked as \/

\ is marked as \\

For example, this text in a HotDocs Model:

```
For a complete description, see H:\\Publications\\Complete Description.doc
\[revised 2006\]
```

appears like this in the final document:

```
For a complete description, see H:\Publications\Complete Description.doc
[revised 2006]
```

Here are two reasons for marking special characters in HotDocs Models: First, macros used to manipulate Markup in HotDocs Models rely on the proper use of special characters. Unmarked special characters appear to be part of the Markup, causing incorrect macro behavior. Second, document drafting applications (such as HotDocs) that accept HotDocs Models as input cannot operate properly if special characters are not marked.

Previous Topic Next Topic

Previous Topic Next Topic

Rules for Naming Variable Fields

As you name your variables, you must follow these rules:

- Names can include letters, numbers, and underscores but they cannot contain spaces or other special characters.
- Names must begin with a letter.
- Names are case-sensitive (meaning that **GiftAmount** and **giftamount** are two different names).
- Names must not be composed entirely of capital letters.
- Names must be 50 characters or less in length.

By default, when you mark up a HotDocs Model, you are required to follow the naming rules described above—particularly by using only letters, numbers, and underscores in field names. If you haven't followed these rules, when you attempt to draft or assemble a HotDocs Model, HotDocs will generate an error message and stop the assembly process. When automating templates in HotDocs, however, you can include spaces and non-alphanumeric characters in variable names. This discrepancy between naming standards becomes problematic when you convert a HotDocs Template to a HotDocs Model and then try to assemble the HotDocs Model. In order to remain compatible, you can define a setting either in HotDocs Options or in the HotDocs Model that allows you to use non-standard names (or names that have spaces and non-alphabetic characters) for fields. See Define Settings for a HotDocs Model for details.

A variable field name typically consists of at least two words that identify the subject and the aspect of the field. Some examples are:

ClientCity	**Client** is the subject and **City** is the aspect.
ClientZipCode	**Client** is the subject and **ZipCode** is the aspect.
AgreementEffectiveDate	**Agreement** is the subject and **EffectiveDate** is the aspect.
TermOfAgreement	**Agreement** is the subject and **TermOf** is the aspect.

While you can use any scheme you choose for naming variables, it's recommended that you capitalize the first letter of each word in a variable field name, regardless of its part of speech, as is shown in the examples above. One reason for doing this is because during a HotDocs interview, HotDocs displays the questions from the document so the user can answer them. If no prompt is specified for a variable in the markup, HotDocs can generate default prompts based on this scheme, adding spaces where each word is capitalized. So, for example, the variable field **ClientName** will automatically be assigned the prompt **Client name**.

Previous Topic Next Topic

Previous Topic Go to Section 2. Marking Up the Document

Steps for Creating a HotDocs Model

The following instructions define, in simple terms, the steps you go through to create a HotDocs Model. The instructions include links to topics that describe the process in further detail.

The basic steps for creating a HotDocs Model include:

- Marking variable fields.
- Marking conditional text.
- Marking repeated text.
- Grouping variables in dialogs and creating an interview.

To create a HotDocs Model with variable fields

1. Open the document in Microsoft Word.
2. Identify the text that will change, depending on the user. For example, find all names, dates, descriptions, and so forth.
3. Replace the text with variable fields. For example, replace all references to the client's name with a markup field, like **[ClientName;te]**. (See Mark Variable Text.)

   You can use Word's **Find and Replace** feature to find multiple instances of the same word in the document.

4. Specify any additional information about the variables in tables at the end of the document. (See Define Field Properties.)

To mark conditional text in the document

1. Identify the sections of the document that are conditional.
2. Mark these sections using any combination of **IF**, **Else IF,** or **Else** fields. (See Mark Conditional Text.)
3. Optionally, hide the conditions in the fields to make the markup easier to read. (See Manage Markup Fields in the Document.)

To mark repeated text in the document

1. Identify the sections of the document that must be repeated.
2. Mark these sections using **REPEAT** fields. (See Mark Repeated Text.)
3. Define the dialogs for the variable fields you need repeated. (Make sure you also define repeat styles for the dialogs.) (See Group Variables in Dialogs.)
4. Optionally, hide the repeat field properties to make the markup easier to read. (See Manage Markup Fields in the Document.)

To group variables into dialogs

1. In the tables section of the document, create a **Dialogs** table.
2. List the variables you want included in each dialog, in the order you want the variables asked, in the **Contents** column of the table.
3. Optionally, create an Interview table and define the order you want the dialogs asked in the interview. (See Define the Order of Dialogs in the Interview.)

Once your documents are marked up, you can either assemble them using HotDocs, or you can convert them to HotDocs template format. For instructions on doing this, please see Create HotDocs Template from Model Or Assemble a HotDocs Model.

You can also define settings that control how the HotDocs Model is assembled. For details, see Define Settings for a HotDocs Model.

For examples of how to mark up a HotDocs Model, see Simple Markup Example (Contract), Simple Markup Example (Agreement), and Complex Markup Example with Tables (Last Will and Testament).

Previous Topic Go to Section 2. Marking Up the Document

2. Markup the Document

Previous Topic Next Topic

Mark Variable Text

When marking up a HotDocs Model, you can identify text that changes in the document, depending on the user. If you are careful to use the Markup described in this help file, you can convert a HotDocs Model to HotDocs template format, or you can assemble the document directly in HotDocs, while still maintaining a reviewable copy of the document.

As you mark up the HotDocs Model, you replace names, dates, amounts, and other client-specific or matter-specific text with variable fields.

There are six different field types:

Field Property	Name Description
text (or te)	Used for text fields
number (or nu)	Used for number fields (monetary amounts, figures, etc.)
date (or da)	Used for date fields
true/false (or tf)	Used for true/false fields (text that is represented by yes/no or true/false answers, or text that is merged whether a condition is true or false)
multiple choice (or mc)	Used for predefined options, such as gender.
computation (or co)	Used for computation fields (values that are calculated, etc.)

Variable field types can be specified using either the full type name (like **text**, **number**, **date**, **true/false**, **multiple choice**, and **computation**) or the first two letters of the word that represents the type (like **te**, **nu**, **da**, **tf**, **mc**, and **co**, respectively). Field types are case-insensitive, but it is recommended you specify them using lowercase letters.

See Rules for Naming Variable Fields for an explanation of how to name your variables

At its most basic, a variable field should include a name and a field type; however, it can also include other properties, such as a format that defines how the answer should appear in the final document.

A field is enclosed in square brackets. The field name appears first, followed immediately by a semicolon and then the field type. Any other properties you need to assign to the field may appear after the type and must be separated by semi-colons. (You can also define these properties in a variable table. See Define Field Properties for details.)

For example, the following text:

```
The client, JANE DOE, hereby rescinds all previous claims.
```

would be marked like this:

```
The client, [ClientName;te;format=upper], hereby rescinds all previous claims.
```

In this example, **ClientName** is the variable field name and **te** (which stands for **text**) is the type. The **format** should be **upper** (which stands for **uppercase**).

You can assign additional properties to a field. For a list of these properties, see Define Field Properties.

See Simple Markup Example (Contract), Simple Markup Example (Agreement), and Complex Markup Example with Tables (Last Will and Testament) for examples of how to mark up a HotDocs Model.

Previous Topic Next Topic

Previous Topic Next Topic

Format Answers in the Document

As you mark up the HotDocs Model, you may want an answer to appear in a special format. For example, you may want to spell out a number (twenty-five vs. 25) or you may want to use the long form of a date (such as 14th day of March, 2007). To do this, you can assign a format property to a variable. When the document is assembled, the answer will be formatted according to the markup.

To format an answer, you first define the format property, followed by an equal sign and the specific format you want to use.

For example, the following marked up date:

```
[AgreementDate;da;format="dth day of Mn, YYYY"]
```

would merge the following answer in the document:

```
6th day of November, 2008
```

The type of format you assign depends on the type of markup field you're inserting. See the tables below to view the available formats:

Text Formats

Property Name	Description	Defined As	How It Formats the Answer
upper	All letters upper case	format=upper	THIS IS THE ANSWER.
lower	All letters lower case	format=lower	this is the answer.
title	First letter of each word uppercase	format=title	This Is The Answer.
sentence	First letter of first word upper case	format=sentence	This is the answer.

The sentence format capitalizes the first letter of the first word in the answer, regardless of the answer's placement in the paragraph or document.

Number Formats

Property Name	Description	Defined As	How It Formats the Answer
alpha	Case-sensitive, spelled out number	format=alpha	nine
		format=Alpha	Nine
		format=Alpha	NINE
ordinal	Case-sensitive, spelled out ordinal number	format=ordinal	ninth
		format=Ordinal	Ninth
		format=ORDINAL	NINTH
09	Number including a leading 0 if the user enters one in the answer	format=09	04
			78
9	Simple numeral	format=9	98
			7,952
9 1/8	Fraction	format="9 1/8"	2 1/4
9,999.00	Monetary amount with cents (even if no cents are entered)	format=9,999.00	9,216.00
			9,216.92
9.9	Whole numeral with optional decimal places	format=9.9	87.1254
9999	Numeral with no thousands separator	format=9999	12875
9th	Ordinal number	format=9th	23rd
Nine Dollars and Twelve cents	Spelled out monetary amount	format=Nine Dollars and Twelve cents	Five Dollars and Thirty-Six Cents
ninth	Spelled out ordinal number	format=ninth	thirteenth

	format=Ninth	Thirteenth
	format=NINTH	THIRTEENTH

Date Formats

HotDocs 2008 and Earlier Users

Property Name	Description	Defined as	How It Formats the Answer
d	Numeric day	format=d	1
m	Numeric month	format=m	1
y	Numeric year (4 digits)	format=y	2008
dd	Two-digit numeric day	format=dd	01
mm	Two-digit numeric month	format=mm	01
yy	Two-digit numeric year	format=yy	08
yyyy	Four-digit numeric year	format=yyyy	2008
dth	Case-sensitive, numeric ordinal day	format=dth	1st
		format=dTH	1ST
dy	Case-sensitive, spelled out day	format=dy	first
		format=Dy	First
		format=DY	FIRST
mn	Case-sensitive, spelled out month	format=mn	january
		format=Mn	January
		format=MN	JANUARY
yr	Case-sensitive, spelled out year	format=yr	two thousand eight
		format=Yr	Two Thousand Eight
		format=YR	TWO THOUSAND EIGHT

wd	Case-sensitive, spelled out weekday	format=wd	monday
		format=Wd	Monday
		format=WD	MONDAY
mnt	Case-sensitive month abbreviation	format=mnt	jan
		format=Mnt	Jan
		format=MNT	JAN
wdy	Case-sensitive weekday abbreviation	format=wdy	mon
		format=Wdy	Mon
		format=WDY	MON

HotDocs 2009 and Later Users

Property Name	Description	Defined as	How It Formats the Answer
d	Numeric day	format=d	1
dd	Two-digit numeric day	format=dd	01
ddd	Case-sensitive weekday abbreviation	format=ddd	mon
		format=Ddd	Mon
		format=DDD	MON
dddd	Case-sensitive spelled-out weekday	format=dddd	monday
		format=Dddd	Monday
		format=DDDD	MONDAY
dth	Case-sensitive numeric ordinal day	format=dth	1st
		format=dTH	1ST

dy	Case-sensitive spelled-out day	format=dy	first
		format=Dy	First
		format=DY	FIRST
m	Numeric month	format=m	1
mm	Two-digit numeric month	format=mm	01
mmm	Case-sensitive month abbreviation	format=mmm	jan
		format=Mmm	Jan
		format=MMM	JAN
mmmm	Case-sensitive spelled-out month	format=mmmm	january
		format=Mmmm	January
		format=MMMM	JANUARY
y	Numeric year	format=y	2009
yy	Two-digit numeric year	format=yy	09
yyyy	Four-digit numeric year	format=yyyy	2009
yr	Case-sensitive spelled-out year	format=yr	two thousand nine
		format=Yr	Two Thousand Nine
		format=YR	TWO THOUSAND NINE

True/False Formats

Property Name	Description	Defined as	How It Formats the Answer
true/false	If the answer is true, merges text to the left of the forward slash; if false, merges the text to the right of the slash	format="true/false"	True text False text
yes/no	If the answer is true, merges text to the left of the forward slash; if false, merges the text to the right	format="yes/no"	True text False text

	of the slash		
truetext/falsetext	If the answer is true, merges text to the left of the forward slash; if false, merges the text to the right of the slash	format="truetext/falsetext"	True text False text

List Formats

Property Name	Description	Defined as	How It Formats the Answer
a, and b	Comma after the first in the series, even if there are only two items in the series	format=a, and b	apples, and oranges
a, b	Leaves out the conjunction	format=a, b	apples, oranges, cherries
a, b and c	Leaves out the comma before the final item in the series; items in list are not capitalized	format=a, b and c	apples, oranges and cherries
A, b and c	Leaves out the comma before the final item in the series; first item in list is capitalized	format=A, b and c	Apples, oranges and cherries
A, B and C	Leaves out the comma before the final item in the series; items are capitalized	format=A, B and C	Apples, Oranges and Cherries
A, B AND C	Leaves out the comma before the final item in the series; items are capitalized; conjunction is upper case	format=A, B AND C	Apples, Oranges AND Cherries
a, b or c	Uses the conjunction or to separate items in the list	format=a, b or c	apples, oranges or cherries
a, b, and c	Includes the last comma in the series	format=a, b, and c	apples, oranges, and cherries
a; b; and c	Uses semi-colons to separate items in the list	format=a; b; and c	apples; oranges; and cherries

Previous Topic Next Topic

Previous Topic Next Topic

Define Field Properties

When marking up a document, there are two aspects of the markup you must be aware of—how the answer will appear in the final document and how the answer will appear in the HotDocs interview used to create the final document. You can define these properties in two different places—directly in the field or in properties tables at the end of the document.

This topic includes the following sections:

- Creating Properties Tables for Storing Properties
- Specifying How Answers Are Merged in the Final Document
- Specifying How Questions Appear During the Interview

Creating Properties Tables for Storing Properties

When assigning multiple properties to a variable, rather than include the properties directly in the field, you can include them in a properties table. Using a properties table lets you specify multiple properties for each of your variables without cluttering the actual text of the document.

Properties tables are Word tables that are located at the end of the document, after a special **[EndDocument]** marker. While you can format the look of the table any way you choose (for example, by making text in the table bold or by shading cells), the actual content of the table must be organized a specific way:

- The first row of the table must identify the variable type. (In the example below, this is Text Variables.) Nothing else should be specified in this row.
- The second row of the table lists the most common properties you assign to all your variables (for example, prompts). This row must include the **Name** column, but all other columns are optional and can represent the properties you assign most frequently to your variables. If there are more properties for some variables than can be displayed reasonably in the table, you can include an **Additional** column, where you list the additional properties. (Separate the properties in this column using a semi-colon.)
- The third and all following rows list the individual variables with their associated property values.

Properties are separated by semi-colons. Properties are identified in name/value pair combinations. For example, if you want to create a number variable, you would define the following properties:

```
format=9,999.00;currency=$;decimal=2
```

When you use variable tables, you must specify the place in the document where the document text ends and the tables begin. You do this using the **[EndDocument]** field marker. It appears after the last section of document text but before the first variable table.

Here is an example of an **[EndDocument]** marker, followed by a variable table:

```
[EndDocument]

TEXT VARIABLES
```

Name	Title	Prompt	Additional
EmployeeName	Name of Employee	Enter the employee's name	format=upper; resource="All employees must submit to a background check to verify their eligibility to work for Hobble Creek Publishing."
JobTitle	Job Title	Enter the employee's job title	
JobDescription	Job Description	Complete the following sentence: Job duties shall include:	height=3

When you assemble the HotDocs Model using HotDocs (or when you create a HotDocs template from the HotDocs Model), HotDocs looks for variable tables and uses the information in them to create and set properties for HotDocs variables. If no tables are found, HotDocs uses the properties defined in the markup fields of the document.

See Complex Markup Example with Tables (Last Will and Testament) for an example of a HotDocs Model that uses variable tables.

Specify How Answers Are Merged in the Final Document

You can assign properties to a variable that will affect how a user's answer will appear in the final document. For example, you can assign a format for the answer (such as uppercase or spelled out), whether the answer should break across lines of text or not, and how the field should look if the user chooses not to answer it, just to name a few.

These properties can be defined in one of two places—in the variable field (described just below) or in a properties table (described earlier).

Properties are separated by semi-colons and are identified in name/value pair combinations. For example, the following would represent a text variable and how it should be formatted in the document:

```
[ClientName;te;format=upper;nonbreak=yes]
```

Specifically, this markup field will merge the client's name in uppercase, and it will be non-breaking (meaning the name won't break across lines in the document).

When a property is assigned at the field level, it will apply to that field only. If a property is defined in a table, it will be used for all variable references. Where a property is defined both in the field and in a table, the field-level property will take precedence.

The following table describes all of the available field properties with their associated names and values. Default property values are in parentheses. When a property is not specified in a variable field, the default

value is assumed. You can include as many properties as necessary, as long as you separate each property with a semi-colon.

Property Name	Description	Value
type	This specifies the variable type	(text) or (te) number or nu date or da true/false or tf multiple choice or mc computation or co
format (text fields)	This is the format for text fields, including text variables, multiple choice variables, and computation variables that produce a text answer. It specifies how the answer will be formatted in the final document.	(none) When you don't specify a format, text is formatted as the user enters it. upper lower title sentence
format (number fields)	This is the format for number fields, including number variables and computation variables that produce a number answer. It specifies how the answer will be formatted in the final document.	(none) 9,999.00 Nine 9th etc.
format (date fields)	This is the format for date fields, including date variables and computation variables that produce a date answer. It specifies how the answer will be formatted in the final document.	(none) d/m/yy dd/mm/yyyy Mn, D, Y etc.

format (true/false fields)	This is the format for true/false fields, including true/false variables and computation variables that result in true/false. It specifies how the answer will be formatted in the final document. If the value is true, the text to the left of the "/" will be merged. If the value is false, the text to the right of the "/" will be merged.	(none) true/false yes/no truetext/falsetext
format (multiple choice fields)	This is the list format, which is used with multiple-select multiple choice variables. It specifies how the answer will be formatted in the final document.	For single-select options: upper lower title sentence For multiple-select options: A, B and C a; b; and c etc.
nonbreak	This property can be used to keep users' answers from breaking across lines in the final document.	(no) yes
unanswered	This property specifies the text that will be merged into the final document when the variable is unanswered.	any text
font	This property specifies the font that will be used when the user's answer is merged into the final document.	(default font used in document) font name
comment com	This is a comment for the field. It does not affect assembly and will not be merged into the final document. It is used to make in-line notes in the document.	(none) any text

For the full value lists and explanations see Format Answers in the Document

Specify How Questions Appear During the Interview

If you assemble the document using HotDocs, the fields you create in the document will appear as questions in a HotDocs interview where you will enter your answers. As such, you can define how the questions will be formatted during this interview.

Again, properties are separated by semi-colons. They are identified in name/value pair combinations, like this:

```
format=9,999.00;currency=$;decimal=2
```

The following table describes properties common to all variable fields. Subsequent tables then discusses variable-specific properties:

Common Field Properties

Property Name	Description	Value
title	Designates an alternate name for the variable, which is used in the HotDocs interview outline.	any text
prompt	Specifies the text that guides users in knowing how to answer the question during the HotDocs interview. If you specify a prompt, it will be used as the interview "question." If you do not, HotDocs will attempt to generate a prompt based on the field name. It does this by inserting a space wherever it finds a capital letter. So, for example, the field name **ClientAddress** would generate the prompt **Client address**.	any text
resource	Provides additional help text for answering the question. This text appears in the resource pane of the HotDocs assembly window	any text
irrelevant (or irrel)	Specifies how a variable that isn't used in the document (for example, because it's conditioned and the condition resolves to false) should be treated—whether it's grayed out, hidden, or always shown.	(gray) hide show
ask	Specifies whether a variable or dialog is asked automatically during the interview.	(yes) no
save	Specifies whether the answer the user enters can be saved to an answer file at the end of an interview.	(yes) no

warn	Causes HotDocs to display a warning if the variable is left unanswered during the interview.	(yes) no

Text Fields

Property Name	Description	Value
height	Determines the height of the answer field that appears in the interview, allowing it to show more than a single line of text	(1) Any number between 1 and 12
maximum (or max)	Determines how many characters can be used in the answer	(as many as the user enters, up to 15,000)
pattern	Determines a specific pattern that will be used for entering the answer (such as a telephone number or time of day)	(none) (999) 999-9999 999-99-9999 etc.
enter	Determines whether a new paragraph is created when the user presses the Enter key in a multi-line field. (By default, pressing Enter simply starts a new line in the same paragraph.)	(break) paragraph

Number Fields

Property Name	Description	Value
minimum (or min)	Specifies the minimum value for the answer	(0) Any number
maximum (or max)	Specifies the maximum value for the answer	(0) Any number

decimal	Specifies the maximum value for the answer	(0) Any number between 0 and 7
currency	Specifies the maximum value for the answer	(none) $ £ € etc.

True/False Fields

Property Name	Description	Value
style	Determines whether the Yes/No prompts for the variable appear on the same line	(row) column

Multiple Choice Fields

Property Name	Description	Value
options	Specifies the options of the multiple choice variable. (You are required to include one or more options with all multiple choice variables.)	Option1/Option2/Option3
optionprompts	Specifies the text that can be used to identify the options (perhaps because the options aren't descriptive enough)	Prompt1/Prompt2/Prompt3
merge	Specifies the text that will be merged in the final document if the user chooses the correlating option during the interview	Merge1/Merge2/Merge3
optionresources	Specifies the resource text for each option you've defined	Resource1/Resource2/Resource3
select	Specifies whether the user can choose one option or multiple options	(single) multiple

other	Determines whether the user can enter an "other" option for single-select options	(no) yes
none	Determines whether the user can select a "none of the above" option for multiple-select options	(no) yes
style	Specifies how the options will be presented to users in the interview	(dropdown) (grid) column list The default option depends on whether the field is set to multiple selection or single selection.

Computation Fields

Property Name	Description	Value
script	This is the formula or calculation (written in HotDocs scripting language) that determines the value for the answer. It is a required property for computation fields.	A valid HotDocs script
merge	Specifies text that can be merged if the computation script generates a true or false value	Text This property can be used when the Computation field generates a true/false value.

Dialogs

Property Name	Description	Value
title	Specifies the text that is used to represent the dialog in the assembly window title bar and in the interview outline If you do not specify a title, HotDocs will attempt to generate a title, based on the dialog name. It does this by inserting a space	Any text

	wherever it finds a capital letter. So, for example, the dialog name **ClientInformation** would generate the title **Client Information**.	
style	Specifies whether the dialog is asked once, as a series of dialogs, as a spreadsheet, or as a spreadsheet on the parent dialog.	(regular) repeated spreadsheet ssonparent
contents (required)	Lists the variables that are to be included in the dialog. (The order in which you list these variables will be the order they are asked in the dialog.)	Variable1/Variable2/ Variable3
group	Groups all True/False variables so they are represented either by check boxes (which allow users to choose multiple options) or option buttons (which allow users to choose only one option).	(none) single multiple
none	Specifies whether a grouped list of True/False variables includes a None of the Above option.	(no) yes
label	Specifies the text that will be used to identify the entire group of answers in a repeated list	Any text This option can only be used when the dialog's style is set to **repeated**
rows	Specifies the number of rows that are visible in a spreadsheet dialog. (This does not affect the number of answers a user can enter—just how many rows of the spreadsheet are visible at a given time.)	Any number This option can only be used when the dialog's style is set to **Spreadsheet**
prompt	Shows the prompt that will be used if the dialog is inserted in a parent dialog Any text	Any text
irrelevant	Causes the dialog to be hidden when all the variables in the dialog are inactive	(hide)

show

Language Fields

Property Name	Description	Value
decimal	Indicates the character that will be used to show the decimal character in a non-English number field	Any character (usually a comma)
grouping	Indicates the character that will be used to show the thousands separator in a non-English number field	Any character (usually a period)

Insert Fields

Property Name	Description	Value
keep	When inserting another document in the parent model, lets you keep the header and/or footer from the inserted document	(none) header footer both
grouping	Indicates the character that will be used to show the thousands separator in a non-English number field	Any character (usually a period)

See Simple Markup Example (Contract), Simple Markup Example (Agreement), and Complex Markup Example with Tables (Last Will and Testament) for examples of how to mark up a HotDocs Model.

You can have HotDocs define default prompts and titles for fields in the HotDocs Model. See Define Settings for a HotDocs Model for details.

Previous Topic Next Topic

Previous Topic Next Topic

Mark Conditional Text

Text in a document is conditional if it is included only when certain conditions are met.

There are two types of conditions you use in markup—simple conditions and alternative conditions.

Marking Simple Conditions

To mark conditional text, you surround the text using opening and closing square brackets. Immediately after the opening bracket, type the word **if** followed by a colon. After the colon, type a condition name followed by another colon. The document text comes after this last colon.

In this example:

```
[if:ClientIsMarried:I give to my spouse all interest in property, both real and
personal, including buildings, fixtures and improvements to the property.]
```

The opening bracket, followed by **if**:, indicates the beginning of the condition. The condition name, **ClientIsMarried**, is a true/false variable that has been defined in a True/False Variables table. The closing square bracket indicates the end of the conditional text.

A condition name is an affirmative statement specifying the conditions under which the conditional text is included in the document. This can be a true/false variable, a computation variable that returns a true/false value, or it can be an expression. If the condition name isn't defined elsewhere in the document (for example, as a variable field inserted elsewhere in the document or in a True/False Variables table), HotDocs will use the condition name to create a simple true/false variable. All of the rules that apply to variable field names apply to condition names.

When conditional text consists of a complete paragraph, it is usually desirable to include the paragraph mark as part of the conditional text. To do this, type the closing bracket after the paragraph mark, such that it appears at the beginning of the following paragraph. The following example, which shows the paragraph marks, demonstrates this:

```
My name is [ClientName;te]. ¶

[if:ClientIsMarried:I am married to [SpouseName;te]. ¶

][if:ClientHasChildren:The names and birth dates of my children are
[ChildrenNamesAndBirthdates;te]. ¶

]Except as specifically noted in this document … ¶
```

Additionally, conditional text may be "nested" within other conditional text. For example:

```
[if:ClientIsMarried:I give to my spouse all interest in property, both real and
personal, including buildings[if:GiftsIncludeFixtures:, fixtures] and
improvements to the property.]
```

Marking Alternative Text

Conditional text may include one or more alternatives, only one of which will be included in the final document. To specify alternatives, introduce each alternative with a forward slash character (/) followed by

elseif to identify a conditional alternative, or by **else** to identify a conditionless alternative that will be included in the document if no other alternative is included.

Here is a simple example showing a conditionless alternative:

```
I nominate my [if:ClientIsMale:wife/else:husband], [SpouseName;te] as my
personal representative.
```

Here is a complex example with several conditional alternatives followed by a conditionless alternative:

```
I direct that my body be [if:ClientToBeBuried:buried in the
[ClientBurialCemetery;te]/elseif:ClientToBeCremated:cremated and that my ashes
be [ClientAshDisposition;te]/elseif:ClientBodyToMedicine:donated to
[ClientBodyInstitution;te]/else:disposed of as directed by my personal
representative].
```

A conditionless alternative (else instruction) is not required, but if you include one, it must appear after all other conditional alternatives (else if instructions).

See Simple Markup Example (Contract), Simple Markup Example (Agreement), and Complex Markup Example with Tables (Last Will and Testament) for examples of how to mark up a HotDocs Model.

Previous Topic Next Topic

Previous Topic Next Topic

Mark Repeated Text

As you mark up the document, you may need to identify sections of the document that need to be repeated. For example, if you're inserting a list of children, you can designate that the variable field, **[ChildName;te]** be repeated.

To do this, you surround the text in the document you want to repeat (including variable fields) with repeat field markers. A repeat field consists of an opening bracket, the repeat instruction (including the name of the dialog you want to repeat) and the text you want repeated. You must also include a closing bracket at the end of the repeatable section of text. In the field itself, you must separate the dialog name from the properties using semicolons. After the last defined property, you include a final colon.

For example:

```
My children are listed as follows: [repeat:Children;format=”a, b, and
c”:[ChildName;te]].
```

Fields between the brackets will automatically be added to the repeated dialog. If you want to change the order they appear, you can either define the order in the Dialogs table, or you can

include a **Contents**= property directly in the repeat field. Separate the field names using forward slashes, like this:

```
[repeat:BequestList;contents=BeneficiaryName/BequestProperty: I give
[BequestProperty;te] to [BeneficiaryName;te]. ...].
```

When repeating a complete paragraph, it is usually desirable to include the paragraph mark as part of the repeated text. To do this, type the closing bracket after the paragraph mark, such that it appears at the beginning of the following paragraph. For example:

```
[repeat:SpecificBequestInformation:I give [BequestProperty;te] to
[BeneficiaryName;te]. If [BeneficiaryName;te] is not living upon my death, the
specific bequest shall lapse and be distributed to my then living children.¶ ]
```

You can assign the following additional properties to a repeat field:

Many of these properties are described in greater detail in the topic Group Variables in Dialogs.

Property Name	Description	Value
name	Shows the name of the dialog you are repeating	A valid dialog name
contents	Lists the variable fields you want repeated	Variable1/Variable2/Variable3/etc.
format	Describes how a series of answers in a sentence will be formatted and punctuated	("")
		"a, b, and c"
		"a; b; and c"
		etc.
ascend	Sorts the list, based on the specified variable, in alphanumeric (A to Z) order	A variable used in the dialog
descend	Sorts the list, based on the specified variable, in reverse alphanumeric (Z to A) order	A variable used in the dialog
filter	Lists a computation variable that limits (or filters) the list of answers you want merged in the final document	A variable used in the dialog
sortfirst	Sorts the list of items before it applies any filters you've specified	(false)
		true

You can sort answers in your list on two different levels by including two ascend or descend properties in the field. The first ascend/descend property will define the first level of sorting,

while the second ascend/descend property will define the second level of sorting. For example, you may want to sort all states in a list in alphabetical order. You can then sort all cities within a state in alphabetical order, too.

The ascend, descend, and filter properties are processed in the order they are encountered. For example, by placing the ascend or descend properties before the filter property, the list of answers will be sorted before the filter is applied.

By default, a repeated dialog with five or fewer variables is set to appear in the interview as a spreadsheet dialog showing only three rows in the spreadsheet.

Repeating Rows in a Table

At times, you need to repeat a row of a table. To do this, you insert an opening field marker before the repeat instruction, but then leave off the closing field marker. This is because the end of the row signifies the end of the repeated field. For example, the following shows how to mark up a repeated table:

Child's Name	**Child's Birth Date**
[repeat:Children:[ChildName;te]	[ChildBirthDate;da]

Note there is no closing bracket after the **ChildBirthDate** field. This is because the end of the table row signifies the end of the repeat field.

Nesting Repeated Fields

You can nest repeat fields inside each other. For example, say you need to create a list of children, and then for each child, list the property items each child will inherit.

There are two steps to doing this: 1) nesting the repeat fields in the HotDocs Model, and 2) listing the nested, repeated dialog as a variable of the main dialog in the Dialogs table.

So, for example, in the document, your text would look like this:

```
[repeat:ChildrenInfo:To [ChildName;te], my child, I give the following items:
[repeat:ItemListInfo;format=”a; b; and c”:[Item;te]].

]
```

The Dialogs table would look like this:

Dialogs

Name	**Title**	**Contents**	**Resource**	**Additional**
ChildrenInfo	Children Information	ChildName ItemListInfo		style=repeated; ascend=ChildName
ItemListInfo	Item Information	Item		style=ssonparent

Note that the dialog **ItemListInfo** appears in the Contents list for **ChildrenInfo**. Both dialogs include a repeat style.

The property, ssonparent, stands for spreadsheet on parent. It means that the dialog will appear as a spreadsheet and be embedded directly on its parent dialog.

See Simple Markup Example (Contract), Simple Markup Example (Agreement), and Complex Markup Example with Tables (Last Will and Testament) for examples of how to mark up a HotDocs Model.

Previous Topic Next Topic

Previous Topic Next Topic

Other Instruction Fields for HotDocs Models

In addition to using IF and REPEAT fields in a HotDocs Model, you can use the following other instructions:

Field Property Name	Title	Contents
ask	When HotDocs creates an interview for the user, it reads through the HotDocs Model and displays dialogs of questions based on the order it encounters markup fields in the document. However, if you want your dialogs to appear in a different order—for instance, if you want a certain dialog to appear first, even though variables that prompt the dialog to be asked appear at the end of the document—you can use an **ask** field to force HotDocs to display the dialog.	[ask:VariableName] [if:ClientIsMarried:[ask:ClientSpouseName]] [if:MONTHS FROM(Date of Previous Filing DA, TODAY) >= 13:[ASK Current Insurance TE]]
default	If a variable field is unanswered, a **default** field will set the value of a variable to a specific value. (If the variable field is already answered, the **default** instruction has no effect.)	[default:VariableName=Value] [default:EmployeeStatus=Temporary]

set	Variable fields normally get their values from the answers users enter during an interview, but sometimes you may want to assign an answer to a variable instead of allowing the user to specify the answer. For example, a document might include the address of the client and, in another place, the address of the client's spouse. Once the client's address has been entered by the user, you could use a **set** instruction to automatically fill in the same address for the spouse, since it will be the same.	[set:VariableName=Value] [set:SpouseAddress=ClientAddress]
increment/ decrement	The **increment** and **decrement** instructions cause HotDocs to increase or decrease a number variable, usually a counter, by the value of 1.	[increment:VariableName] [decrement:VariableName] [increment:TempNum] [decrement:TempNum]
insert	You can insert one template into another by using an **insert** field. For example, you might want to include boilerplate text in multiple documents, or you may want to include a set of related documents in one main HotDocs Model so users can choose the document they want. During document assembly, when HotDocs finds an **insert** field, it stops assembling the main document so it can assemble the inserted document. When it finishes, it continues assembling the main document An optional property of an **insert** field is **keep**, which can have the value of **header**, **footer**, or **both**.	[insert:"FileName.docx"; keep=header] [insert:"Provisions.docx"; keep=both] Both the parent document and any documents referenced in the **insert** field must be saved to the same location.

assemble	You can use an **assemble field** to add templates to the list of documents that should be assembled. Unlike the **insert** field, an **assemble** field adds the model to the list of assemblies (known as the assembly queue) and then waits until the main document is finished assembling before it starts assembling the new, added document.	[assemble:"FileName.docx"] [assemble:"trustdocument.docx"] Both the parent document and any documents referenced in the **assemble** field must be saved to the same location.
languages	You can mark up a HotDocs Model in languages other than English. To do this, you must create a **language** field in the document that allows HotDocs to format dates and numbers in the template so they appear correctly in the assembled document. Optional properties of the **language** field include **decimal**, which identifies the character used as the decimal mark, and **grouping**, which identifies the character used as the thousands separator.	[language:fra;decimal=',';grouping='.'] Valid language codes are: • **dea** (Austrian German) • **des** (Swiss German) • **deu** (German) • **eng** (English) • **fra** (French) • **nld** (Dutch) • **esn** (Spanish) • **ita** (Italian) • **ptb** (Brazilian Portuguese)
span	Frequently, users need to edit document text once a document has been assembled. To allow this, you must mark sections of document text using **span** fields. Inserting **span** fields in a HotDocs Model allows users to edit the text of the assembled document while viewing the **Document Preview** tab of the assembly window. Changes made to the text can be saved in an answer file, which allows users to later reassemble the document and still have access to the changes they made. An optional property of the **span** field is **title**.	[span:spanName;title=SpanTitle:DocumentText] [span:trialPeriod: The length of [Employee Name]'s employment will be an initial term of six months, with the possibility of continuation beyond that period depending on Hobble Creek Publishing's needs and upon the employee's performance.]

Each of the above fields can also specify a **comment** as an additional property, for example:

```
[ask:VariableName;comment=put comment here]
```

Previous Topic Next Topic

Previous Topic Next Topic

Manage Markup Fields in the Document

As you mark up a HotDocs Model, you may want to distinguish between markers and the actual text of the document. For example, you may want to mark fields using specific colors. There are two ways to do this—by using the HotDocs Markup Tools, or by using the built-in features of Word.

Using the HotDocs Markup Authoring Tools

HotDocs Markup Tools are included as part of the HotDocs ribbon, helping you better manage the markup fields in the document. For example, you can apply color to the different fields, as well as hide field properties that aren't essential for reviewing the document. You can also use the toolbar to draft a test document, so you can see what kind of interview the document produces.

You can use the following buttons in the toolbar to complete your different tasks:

Button Name	Description
Apply Color	Applies color to the different fields in a HotDocs Model. Variable fields are colored using blue, If fields using green, and Repeat fields using maroon. Using color like this can help you distinguish between markup and the actual text of the document. (To apply color to a selection of text within the document, select the text before clicking the button.) When you click the **Apply Color** button, the tool also applies Word's Hidden property to the text. This makes it so you can click the **Hide Clutter** button and hide these properties. This can make the document more readable, particularly when you need to review it.
Hide Clutter Show Clutter	Hides the properties of a variable (such as formats and other styles) that aren't necessary for you to review the document. (Click here for an example.) (To hide clutter in a selection of text rather than the entire document, select the text before clicking the button.) To view the full markup language again, click **Show Clutter**.
Select Field	Highlights the block of text merged using an IF field or a REPEAT field. This helps you identify all of the text affected by the instruction. If you click **Select Field** once, it will select just the immediate section of the instruction (for example, the first part of a multi-part condition). If you click the button twice, it will select the entire instruction.

Draft Document	Lets you view the HotDocs interview that is created by the HotDocs Model.
Options	Lets you view whether the HotDocs Model uses an embedded component file. (When you convert a HotDocs template to a HotDocs Markup, HotDocs embeds the component file in the document so that it can retain its properties. Clicking this button helps you know if this is the case. If you want to delete the embedded component file, click **Remove**.) (See Create HotDocs Model from Template)
Help	Opens the HotDocs Model Help file.

Using the Features of Microsoft Word

As you mark up the document, you may find that the number of properties you need to specify for a variable field, conditional field, or repeat field makes the field hard to read and understand, especially for someone unfamiliar with markup. You may also have a difficult time distinguishing between mark-up fields and document text. The following describes some things you can do using tools in Microsoft Word to make this process of managing your markup easier.

Keep Properties from Appearing in Markup Fields

To keep the information appearing in markup fields to a minimum, you have several options:

- **Place the field properties in hidden text**: Once you have assigned all of the properties to the field, apply the hidden text property to the field property. (To do this, highlight the text you want to hide and choose Font (Format menu). At the Font dialog box, select Hidden in the Effects group.) To view the hidden text, click the **Show/Hide ¶** button in the Word toolbar.
- **Place the field properties in footnotes**: Once you have assigned all of the properties to the field, move the text to a footnote. (See the Microsoft Word help for instructions on creating footnotes.)
- **Place the field properties in variable tables at the end of the document**: You can create tables at the end of the document where you define the properties of the variables you use in the document. See Define Field Properties for an explanation of how to do this.

Apply Color to Markup

The markup examples used in this help file use blue to identify variable fields, green to identify conditional text, and maroon to identify repeated text. While color is not mandatory, it is much easier to see markup in a HotDocs Model when color has been applied.

To apply color using the built-in features of Word, select (or highlight) the text to be colored, choose **Font** (**Format** menu), and select the font color. Or, after selecting text, use the **Font Color** button on the **Word Formatting** toolbar.

Previous Topic Next Topic

Previous Topic Next Topic

Rules and Tips for Creating a HotDocs Model

You should keep the following in mind when working with HotDocs Models:

- Sometimes you may want to include the value from another answer in the prompt for a specific variable. To do this, make sure you include the opening and closing brackets around the variable name. For example, the following prompt **Please enter [EmployeeName]'s gender**, would merge the answer for the employee's name in the prompt during the interview, like this: **Please enter Jane Porter's gender**.
- Markup keywords (like **format**, **text**, **style**, and **enddocument**) are case-insensitive. However, variable names (like **ClientName**) are case sensitive.
- Property definitions are cumulative, meaning that if the same property is defined more than once for a given variable or dialog, the last one found in the HotDocs Model takes precedence.
- If you do not define prompts for your variables, by default, HotDocs will attempt to create default prompts based on the field name. For example, the field name **[EmployeeName]** would become the prompt **Employee name**. (HotDocs determines where spaces should be included in the prompt based on capitalization in the field name. Because of this, you should capitalize each individual word in the field name.) See Define Settings for a HotDocs Model for details.
- The same is true for dialog titles. If no title is specified, HotDocs will attempt to create a default title, based on the dialog name. For example, the dialog **[EmployeeInformation]** would use the title **Employee Information**. See Define Settings for a HotDocs Model for details.

Escaping Characters In Your Markup

If you need to use characters in your text that are typically reserved for the Markup specification, you can "escape" them by following these guidelines:

Properties appearing in a variable table column do not need to be escaped. The only exception is when you're listing multiple, different properties in the **Additional** column of the table.

- The opening and closing square bracket ([ ]), forward slash (/), and back slash (\) characters, if used as literal characters anywhere in the HotDocs Model, must always be marked as literal. You can do this by preceding the character with a back slash character.

  For example, this text in a HotDocs Model:

  ```
  For a complete description, see H:\\Publications\\Complete Description.doc
  \[revised 2006\]
  ```

  appears like this in the final document:

  ```
  For a complete description, see H:\Publications\Complete Description.doc
  [revised 2006]
  ```

- A forward slash (/), if used as a literal character in any list properties (such as a list of multiple choice options), must be marked as literal. You can do this by enclosing the text string in quotation marks.

  For example, you would mark the following different options :

  ```
  Cars/Trucks

  Recreational Vehicles

  Towing Trailers
  ```

  Like this:

  ```
  "Cars/Trucks"/Recreational Vehicles/Towing Trailers
  ```

- Paragraph marks or line breaks, if used in any list properties (such as multiple choice options), must be enclosed in quotation marks.

  For example, you would mark the following different multiple choice option prompts:

  ```
  (Option 1) Cars

  Trucks

  (Option 2) Recreational Vehicles

  (Option 3) Towing Trailers
  ```

  Like this:

  ```
  "Cars

  Trucks"/Recreational Vehicles/Towing Trailers
  ```

Previous Topic Next Topic

Previous Topic Go to Section 3. Changing the Appearance of Questions and Dialogs in the Interview

Draft a Test Document

As you are marking up a HotDocs Model, you can use the  **Draft Document** button in the HotDocs Markup Tools to draft a test document. When you click this button, HotDocs starts and displays the interview and assembled document for the HotDocs Model you are working on. Being able to see the interview and document can help you make sure the markup you're using in the model creates a complete and accurate document. It also gives you the chance to review how questions appear in the interview, in case you need to define additional properties, such as formats or prompts.

To draft a test document

1. Click the  **Draft Document** button in the HotDocs Markup Tools. A HotDocs assembly window appears.
2. Review the questions in the interview, answering them different ways to produce different versions of your document.
3. Review the assembled document for accuracy and completeness.

If you need to make corrections, make them in the HotDocs Model.

Once you have completed a HotDocs Model, you can add it to a HotDocs template library so it can be assembled by other HotDocs users. See Assemble a HotDocs Model for details.

Previous Topic Go to Section 3. Changing the Appearance of Questions and Dialogs in the Interview

3. Changing the Appearance of Questions and Dialogs in the Interview

Previous Topic Next Topic

Group Variables in Dialogs

If you plan to assemble HotDocs Models using HotDocs, you may find that you want variables to be grouped together during the interview. For example, you may want variables asking for information about the client (such as name, gender, birth date, and so forth) to be asked in the same group, rather than asked individually.

To group variables into dialogs, you must create a Dialogs table in the HotDocs Model. This table lists each dialog you want to use. For each dialog, you list the variables you want to include in the dialog as well as other properties you want defined for the dialog. For example, a typical Dialogs table may look like this:

DIALOGS

Name	Title	Contents	Resource	Additional
DependentData	Dependent Information	DependentExplained ChildName ChildBirthDate	Please list children in birth order, from oldest to youngest	style=repeated
ClientData	Client Information	ClientName ClientGender ClientBirthDate		
PropertyData	Property Information	PropertyDescription PropertyLocation PropertyValue		style=repeated

Sometimes you may need to add standalone text to a dialog to help the user better understand the questions that are being asked. You can do this either by typing the text directly in the Contents column (surrounding it in quotation marks) or by creating a dialog text variable.

Just as you assign properties to variables, you can assign properties to a dialog. For example, you can define a title for the dialog as well as choose a style for the dialog if it must be repeated. The following table describes these properties:

Property Name	Description	Value
name (required)	Specifies the name of the dialog.	Any valid variable name. (See Mark Variable Text for rules on naming variables.)
contents (required)	Lists the variables and dialogs that are to be included in the dialog. (The order in which you list the contents will be the order they appear in the dialog.)	Variable1/ Variable2/ Variable 3 Variables can be separated either by hard returns or by a forward slash.
title	Specifies the title of the dialog. The title is what the user will see during the interview. If you do not specify a title, HotDocs will attempt to generate a title, based on the dialog name. It does this by inserting a space wherever it finds a capital letter. So, for example, the dialog name **ClientInformation** would generate the title **Client Information**.	Any text
resource	Provides additional help text for answering all of the questions in the dialog. This text appears in the resource pane of the HotDocs assembly window.	Any text
style	Specifies whether the dialog is asked once, as a series of dialogs, as a spreadsheet, or as a spreadsheet embedded on the parent dialog.	(regular) repeated spreadsheet ssonparent
group	Lets you group True/False variables so they can appear either as option buttons (which let the user choose one of the options) or check boxes (which let the user choose multiple options).	(none) single multiple

none	Provides users with a "none of the above" option when True/False fields in the dialog are grouped.	(no) yes
label	Lets you assign a label to a dialog that is repeated as a series. The label is used to identify the top node of the repeated dialog in the interview outline.	Any text
prompt	Specifies the text that will be used to identify a child dialog on its parent.	Any text
rows	Defines the visible number of rows in a dialog that is repeated as a spreadsheet.	Any number
ask	Causes the dialog to be asked automatically in the interview.	(yes) no
irrelevant	Defines how a dialog that isn't required in the interview should be treated. For example, if a dialog contains variables that aren't used in the interview, this setting will determine whether the dialog is hidden (because it isn't necessary) or shown.	(hide) show

You can control the order dialogs are asked in the interview. See Define the Order of Dialogs in the Interview for details.

Previous Topic Next Topic

Previous Topic Next Topic

Define the Order of Dialogs in the Interview

As HotDocs processes the HotDocs Model, it displays the dialogs you have created in the order the variable fields are merged in the document. For example, if the first field in the HotDocs Model is **EmployeeName**, HotDocs will ask the variable **EmployeeName** first. If this variable is associated with a dialog, it will ask the dialog instead. It will then move to the next field in the HotDocs Model and repeat the process. (If the dialog has already been asked, it won't ask it again.)

For many, this default order for asking dialogs is adequate. However, if you want more control, you can define this sequence by creating an Interview table. The order dialogs are listed in this Interview table is the order they will be asked by HotDocs.

When defining an interview, you must account for all of the dialogs. Failure to list all of the dialogs in the Interview table will result in some questions not being asked in the interview.

An Interview table lists the dialogs in the order you want them asked. For example:

```
INTERVIEW

ClientData
DependentData
PropertyData
```

Previous Topic Next Topic

Previous Topic Go to Section 4. Using the HotDocs Model in HotDocs

Define Settings for a HotDocs Model

When you work with HotDocs Models, you can define several settings that control how parts of the interview look and feel. For example, you can define options that let HotDocs generate default titles, prompts, and resources for variables in the HotDocs Model.

Additionally, you can select an option that allows you to use non-standard names when marking fields in the HotDocs Model. To explain, the Markup requires you to use only alphanumeric characters and underscores when naming markup fields in your HotDocs Model. However, sometimes when you're converting existing HotDocs templates to model format, variables in the template may use spaces and other special characters in the variable name. Selecting this option allows you to continue to use the template as a model without having to rename all of your variables.

The following table describes the different settings you can include in a Settings table, along with descriptions of each setting:

Property Name	Description	Value
StandardNames	Allows authors to assemble HotDocs Models or convert templates to HotDocs Model where field names include spaces and other special characters. These characters are typically not allowed in HotDocs Model authoring. When a template is converted to a HotDocs Model and it uses non-standard field names, this setting is automatically added to the **Settings** table and set to **False**. If this property isn't defined in the **Settings** table, HotDocs will use the property defined in HotDocs Options. (See the HotDocs Model Naming Standards and Rules for Naming Variable Fields for details.)	(True) False
DefaultPrompts	Generates default prompts based on field names. Prompts are used to identify the question in a dialog. For example, if a text field name is **ClientName**, this setting will automatically assign it the prompt of **Client name**.	(True) False

DefaultTitles	Generates default titles based on field names. Titles are used to describe the field when it appears as an icon in the HotDocs interview outline. For example, if a text field name is **ClientName**, this setting will automatically assign it the title of **Client Name**.	(True) False
DefaultResources	Generates resource text for markup fields based on the document text surrounding the field. This provides context for the question in the interview.	(True) False
IrrelDefault	Defines the default value (hidden, grayed, or always showing) for irrelevant variables in the interview. This option is available in HotDocs 2009 and later.	(Gray) hide show

The following is an example of how you can define a Settings table:

SETTINGS

Name	Value
StandardNames	True
DefaultPrompts	True
DefaultTitles	True
DefaultResources	False

You can also define these settings directly in the template. To do this, simply create a Settings field, like this: **`[Settings:StandardNames=true;DefaultPrompts=false]`**.

Previous Topic Go to Section 4. Using the HotDocs Model in HotDocs

4. Using the HotDocs Model in HotDocs

Previous Topic Next Topic

Assemble a HotDocs Model

Once you have created a HotDocs Model, you can add the model to the HotDocs template library and assemble it using HotDocs.

To add the document to the library and assemble it

1. Start HotDocs.

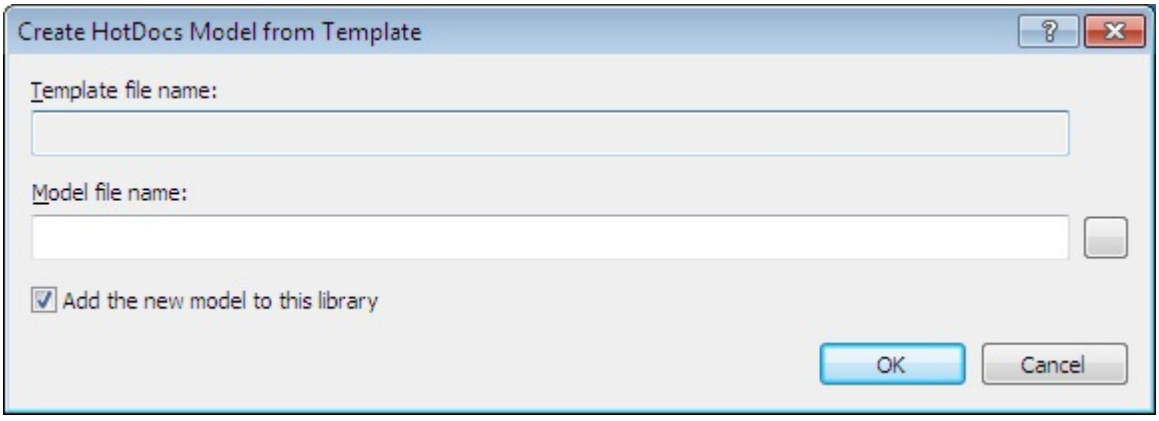
2. At the template library, click the **Add Item** button. The **Add Item** dialog box appears.
3. Click the **Type** drop-down button and choose **HotDocs Model** from the list.
4. Click the **Browse** button next to the **File name** field. The **Add Item File Name** dialog box appears.
5. Locate and select the HotDocs Model and click **OK**.
6. Enter a title in the **Title** field and click **OK**.
7. Once added, select the HotDocs Model in the item list and click **Assemble**.
8. Complete the assembly process.

By default, when HotDocs generates an interview from a HotDocs Model and displays the questions, it shows the surrounding text in the resource pane of the assembly window. This can help provide context for the question being answered. (To view the resource pane, click the **Dialog Resource Pane** button. (You can keep resources from appearing in your interview by defining a setting in the Settings table. See Define Settings for a HotDocs Model for details.)

When you add a HotDocs Model to the template library, HotDocs appends the **HotDocs Model** command-line option (**/mo**) to the file path. This indicates to HotDocs that this is a HotDocs Model and not just a regular Word document.

Previous Topic Next Topic

At a Glance: The Create HotDocs Model from Template Dialog Box



A

B

C

Illustrations used throughout the help file depict HotDocs Developer and may include features not present in HotDocs Developer LE Player User.

After clicking on **Create HotDocs Model from Template** in the **Template** menu of your HotDocs Library you will see the **Create HotDocs Model from Template** dialog box.

In field A you need to select the template you wish to export. The easiest way to do this is to click on the  **Browse** button to the right of field A and navigate to the template you want to export.

HotDocs will automatically fill field B with the same file name as your chosen template. If you would like to use a different file name for the HotDocs Model you can type it into field B.

If you check box C then HotDocs will put a reference to the newly created HotDocs Model into the HotDocs Library you have open. If you would not like to add the HotDocs Model to this library then make sure this box is unchecked.

Go here for more information on creating a HotDocs Model from a template.

Previous Topic Next Topic

Create HotDocs Model from Template

You can create a HotDocs Model from an existing HotDocs template. Saving templates as models allows you to share your templates with subject matter experts unfamiliar with HotDocs. As long as those experts understand (and use) the markup rules specified in this help file, they can further automate the template and use it with HotDocs User to assemble documents from it.

When you create a model from a template, HotDocs converts variable and instruction fields to markup fields. Variable and instruction properties are stored in tables at the end of the document. Component file properties are saved in an embedded component file.

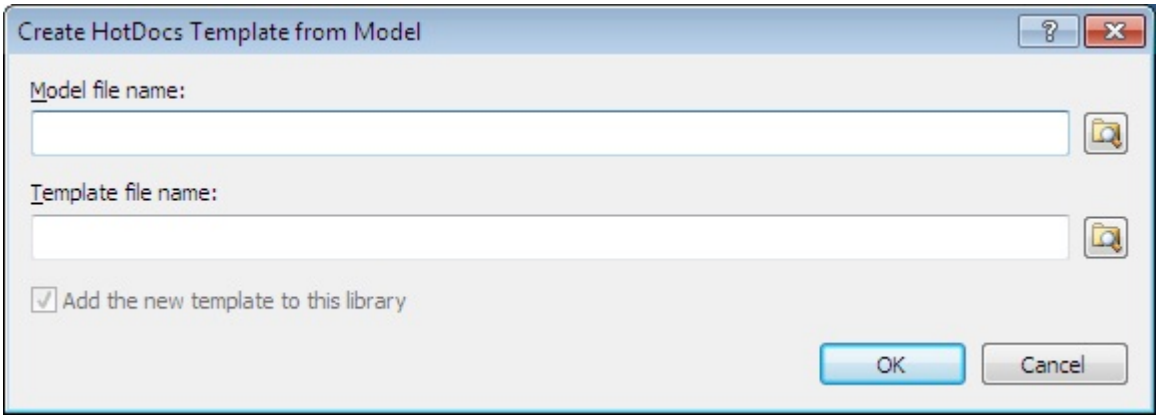
Some template development features are not supported in HotDocs Models. This includes pointed component files and clause libraries.

To Create a HotDocs Model from a Template

1. At the template library, select the template you want to export.
2. Click **Create Model from Template** (**Template** menu). The **Create HotDocs Model from Template** dialog box appears.
3. Optionally, change the name of the HotDocs Model in the **HotDocs Model file name** field. (By default, HotDocs appends the text ***.model*** to the file name. This helps distinguish between it and the original template file.)
4. Optionally, select **Add the new model to this library** to add a reference to the model to the library. This allows you to more easily assemble the HotDocs Model using the **Assemble** command.
5. Click **OK**. HotDocs exports the template as a model. You can further edit the model or distribute copies of it to others for review or additional markup.

Previous Topic Next Topic

At a Glance: The Create HotDocs Template from Model Dialog Box



A

B

C

Illustrations used throughout the help file depict HotDocs Developer and may include features not present in HotDocs Developer LE Player User.

After clicking on **Create HotDocs Template from Model** in the **Template** menu of your HotDocs Library you will see the **Create HotDocs Template from Model** dialog box.

In field A you need to select the HotDocs Model you wish to import. The easiest way to do this is to click on the **Browse** button to the right of field A and navigate to the document you want to import.

HotDocs will automatically fill field B with the same file name as your chosen HotDocs Model. If you would like to use a different file name for the template you can type it into field B.

If you check box C then HotDocs will put a reference to the newly created template into the HotDocs Library you have open. If you would not like to add the template to this library then make sure this box is unchecked.

Go here for more information on creating a HotDocs template from a model.

Previous Topic Next Topic

Create HotDocs Template from Model

Once you have marked up a document, you can create a template from a HotDocs Model. Markup fields in the model are converted to variable and instruction fields. Once created, you can edit the template or assemble it.

To Create a HotDocs Template from a Model

1. At the HotDocs library, choose **Create Template from Model** (**Template** menu). The **Create HotDocs Template from Model** dialog box appears.
2. Click the **Browse** button next to the HotDocs Model **file name** field and locate the HotDocs Model.
3. Enter a file path and name for the newly created template in the **HotDocs template file name** field. (HotDocs automatically suggests the same name as the HotDocs Model, with the text ***.template*** appended to it.)
4. Optionally, select **Add the new template to this library**. This places a reference to the template in the HotDocs library.
5. Click **OK**. The template is created from the marked up HotDocs Model.

Previous Topic Next Topic

Previous Topic Go to Section 5. Including HotDocs Scripting In a HotDocs Model

HotDocs Model Naming Standards

When marking up HotDocs Models, you must adhere to the Markup rules, which requires you to follow certain standards when naming variables and other fields in the document. Failure to follow these standards, particularly #1, may result in errors or other problems when users attempt to assemble the HotDocs Model.

Specifically:

1. You must use only letters, digits, and underscores. You cannot use spaces or other special characters.
2. You should use names that are at least two words.
3. You should make one of the words in the name the "subject" word (Client, Child, etc.).
4. You should capitalize the first letter of each word, regardless of part of speech.
5. You must never use names that are all capitals.

If you must include spaces or special characters in your variable and other field names, you can select a HotDocs option that allows this.

Previous Topic Go to Section 5. Including HotDocs Scripting In a HotDocs Model

5. Including HotDocs Scripting in a HotDocs Model

Previous Topic Next Topic

Including HotDocs Scripting In a HotDocs Model

As you mark up your HotDocs Models, you may need create scripts that perform a particular task, such as insert one model into another or perform some action based on an answer the user provides. Similarly, you may need to add up several dollar amounts, or find the number of years between two given dates. Or, you may need to search a user's answer for a certain string of text.

To accomplish this, you must use the HotDocs scripting language, which consists of instructions, expressions, operators, and values—such as text, numbers, dates, or answers users enter. You can include scripts in your markup to help you accomplish these different tasks.

Specifically, an instruction tells HotDocs to perform some sort of function, while an expression retrieves a special value. Most instructions and expressions also include placeholders, which you must replace with a value. Possible values include text strings, number amounts, other expressions, or variables. An instruction or expression will not work until all of its placeholders are replaced.

In addition to using instructions and expressions, you can use operators to control how a script is processed. Most operators are common mathematical signs, but there are also Boolean operators such as; AND and OR, as well as comparison operators.

When creating scripts, instruction and expression keywords can either be entered in the Script column of a Computation Variables table, or as an in-line property of a computation field. Keyword names are not case sensitive, but it is recommended you enter them in lowercase. Multi-word keywords are entered as one word, capitalizing the second (and any remaining) words in the expression or instruction. For example, to determine the absolute value of a number, you would enter the keyword as absoluteValue.

Template developers automating templates in HotDocs are required to use all uppercase letters when using scripting keywords. To accommodate this difference between template development and creating HotDocs Models, you can specify a setting in the HotDocs Model that allows for uppercase letters in expression and instruction keywords. See Define Settings for a HotDocs Model.

Previous Topic Next Topic

Full List of Expression Models

Expression	Description
absoluteValue(num)	Using the absolute value expression, you can find the absolute value of a given number. You can calculate a negative number, but have it appear as a positive number.
age(date)	The age(date) expression produces an age, in years, by calculating the number of years between the current date (as determined by your computer's system clock) and a date you provide in the computation script.

answered (dialog)	HotDocs can determine whether a dialog has been answered using the answered expression. Even if only one variable in the dialog is answered, the expression returns a value of true.
answered (var)	You can use the answered expression to determine whether a HotDocs variable has been assigned a value. If so, the expression receives the value of true.
count(dialog)	You can find out how many sets of answers a user provides for a repeated dialog. A repeated dialog is any dialog used in a repeat instruction. This expression produces a number, based on each answered dialog.
count(mult_choice_var)	This expression counts how many options a user chooses when answering a Multiple Choice variable. The result it produces is a number.
counter	HotDocs uses counter as a way to compare two incrementing number values. For example, perhaps you want to list the last child named in a repeated dialog. You could create the following computation script:
date - num days	You can subtract any number of days from a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
date - num months	You can subtract a certain number of months from a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
date - num years	You can subtract a certain number of years from a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
date + num days	You can add any number of days to a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
date + num months	You can add any number of days to a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
date + num years	You can add a certain number of years to a Date variable. The result of this computation is a new date value, which can be merged into the assembled document.
dateOf(num, num, num)	This expression finds a date value based on day, month, and year values.
dayOf(date)	This expression returns the day portion (1 to 31) of a given date.
dayOfWeek(date)	This expression determines on which day of the week a specific date falls and converts that value to an integer.
daysFrom(date, date)	This expression allows you to find the number of days between two dates.
first(text, num)	Using this expression, you can return any number of characters starting with the first character in an answer value.
format(value,	Sometimes you may need to add a date, number, or true/false value to a text value.

"example")	You can do this by formatting the date, number, or true/false value as text.
integer(text)	Sometimes you may have a text value that contains number characters, as in the case of a time of day value. The integer expression allows you to convert those number characters into numeric values so you can perform calculations or compare them with other values.
last(text, num)	The last expression finds and returns a certain number of characters from the end of a text string.
length(text)	The length expression counts the number of characters—including spaces and punctuation—in a text value, such as a Text variable.
max(num, num)	The max expression compares two number values and returns the greater of the two.
mid(text, num, num)	Like the first and last expressions, this expression extracts a specified number of characters from within a text string.
min(num, num)	The MIN expression compares two number values and returns the lesser of the two.
monthOf(date)	This expression returns the month portion of a given date.
monthsFrom(date, date)	The months from expression calculates the number of months between two given dates.
mult_choice = text; mult_choice != text	The mult_choice = text expression returns true when the user chooses a Multiple Choice option that is equal to (=) a given text value. If it is not equal (!=), the expression returns false. The mult_choice != text expression functions in the opposite way—testing instead to see if an answer is not equal to (!=) a given text value.
not true_false	You can use the not true_false expression to find out if a True/False variable is false.
other(mult_choice_var)	This expression determines whether the user has chosen the Other option of a Multiple Choice variable and, if so, returns the text entered in the Other field. It can also be used to test whether the user has selected the None of the Above option.
position(text, text)	The position expression finds the position of a certain character or character string in a given text value. It is useful if you need to find a character you know will be in an answer but are not sure where it will appear. It returns a number value, which represents the first character.
power(num, num)	The power expression generates a numeric value, based on a given exponent.
remainder(num, num)	The remainder expression returns the remainder of a division. If the denominator is a zero, HotDocs generates a divide by zero error.
replace(text, text, text, num)	This expression lets you search a string of text for a given character string and replace the results with new text.
result	As you write computations, you often need HotDocs to acknowledge what the result would be at that point in the script. You can update this answer by using the result expression.

round(num, num)	You can round a number value to a specified number of places.
selection(mult_choice_var, num)	This expression lets you retrieve individual options (answers) selected in a Multiple Choice variable. It returns a text value that corresponds to the defined answer (as designated by the num placeholder).
space(text, text)	This expression tests whether the variable is answered. If it is, it merges the answer, followed by a space character. If the variable is unanswered, it merges nothing ("").
strip(text, text, true_false, true_false)	This expression removes a specified character or characters from the beginning or end of a text answer. By default, HotDocs removes the characters from both the beginning and the end of the text. If you want to specify just one or the other, you must use the true_false parameters.
sum(computation_var)	Using the sum expression, you can add computation values that have been repeated.
sum(num_var)	Using the sum expression, you can add repeated number values.
text contains text	The text contains text expression determines whether the first text value contains the same text as the second value. If it does, it returns the value of true.
today	This expression returns the current date, according to your computer's system clock.
truncate(num, num)	You can truncate a decimal number a specified number of places after a decimal point.
unanswered	This expression removes an assigned value from a variable. It is used most often with the set instruction:
union(mult_choice, mult_choice)	This expression creates a single list of all unique options (answers) that have been selected across two or more Multiple Choice variables.
value(var, expression)	This expression returns a default value for the variable type if the variable is unanswered. If the variable is answered, the value is the answer the user specifies.
yearOf(date)	You can use this expression model to find the year portion of a given date.
yearsFrom(date, date)	This expression calculates the number of years between two given dates.
zero(num_var)	This expression returns the value of zero only if a Number variable is unanswered. If the Number variable is answered, the value is the answer the user specifies.

Previous Topic Next Topic

Full List of Instruction Models

Instruction	Description
ascend var; descend var	The ascend instruction sorts lists of answers (gathered using a repeat instruction) in alphanumeric order, from 1 to 9, and from A to Z. The descend instruction sorts lists of answers from 9 to 1, and from Z to A.
ask dialog	The ask dialog instruction allows you to control the order in which dialogs appear in an interview.
ask var	Sometimes a variable needs to be asked by itself. You can use the ask var instruction so that during the interview, HotDocs displays the variable in its own default dialog.
default var to value	This instruction suggests a value for a variable if the variable is unanswered.
filter computation_var	The filter instruction filters out certain entries from a repeated list, based on conditions you specify.
format "list_format"	The format "list_format" instruction allows you to create a sentence-style list within a computation.
if expression; elseIf; else; endIf	You can make sections of a HotDocs Model conditional by using if instructions. You can also make instructions or expressions in a script conditional. A conditional section will be included only if a condition you specify is true. The elseIf instruction allows two or more conditions to be included in an if instruction. The else instruction establishes a final condition for an if instruction, specifying that if all preceding conditions are false, the following information should be included. It must be the last item of the if instruction.
increment num_var; decrement num_var	The increment and decrement instructions cause HotDocs to increase or decrease a number variable, usually a counter, by the value of 1.
insert "filename"	This instruction inserts another HotDocs Model into the document currently being assembled. When HotDocs encounters an insert instruction, it immediately processes the instruction and inserts the model into the current document. If there are variables to be answered, HotDocs presents them before finishing the interview of the main document.
language code	This instruction tells HotDocs to format numbers and dates in a particular language.
repeat dialog; endRepeat	A repeat instruction gathers lists of answers and merges them into a document.
set var to value	This instruction lets you specify a given value for a variable's answer automatically, rather than allow the user to specify an answer. With the set instruction, you can transfer names and other values from one variable to another.
while expression; endWhile	The while expression instruction allows you to repeatedly process (or loop through) an answer or set of answers until a certain condition is met, such as a certain answer is found or a limit is reached.

Previous Topic Next Topic

Use Operators When Scripting

An operator is a symbol or word that causes an operation such as addition or a comparison to be performed in a computation or expression. Most operators can be used when working with both number and text values.

There are three types of operators:

- **Comparison operators**: These compare two values of the same type (text, number, date, multiple choice, or true/false). They return values of true or false depending on whether the comparison is true or not.
- **Arithmetic operators**: These calculate new values. Operands used in the script must be the same type. You can use the Add (+) operator to string together (concatenate) two text values.
- **Logical operators**: These return a true/false value based on a logical comparison of their operands, both of which must be true or false.

The following tables explain how each operator works:

Comparison Operator	Description
=	The two items in the comparison are of equal value. For example: BirthDate = 17 Dec 1989 Employee Name = "Louisa Gehrig"
!=	The two items in the comparison are not of equal value. For example: if ExhibitA != true if PlaintiffGender != "Male"
<	The first item in the comparison has a lesser value than the second item. For example: AccountBalance < 9000 counter < 10
>	The first item in the comparison has a greater value than the second item. For example: DependentAge > 18

<=	The first item in the comparison is less than or equal to the second item. For example: ClientAge <= 65 counter <= 2
>=	The first item in the comparison is greater than or equal to the second item. For example: TaxedIncome >= 75000

Arithmetic Operator	Description
+	Add the different components of the script together. For example: Value1 + Value2 ClientStreet + ", " + ClientCity + ", " + ClientState
-	Subtract the different components of the script from each other. For example: MonthlyIncome - AmountOfOwedChildSupport
*****	Multiply the different components of the script. For example: PurchasePrice * 0.625
/	Divide the different components of the script. For example: YearlySalary / 12

Logical Operator	Description
AND	The statement to the left and the statement to the right must both be true. For example: if ClientIsMarried and ClientHasChildren
OR	The statement to the left or the statement to the right must be true. For example: if ClientIsSingle or ClientIsWidowed

NOT	The two items in the comparison must not be equal to each other.

The final operator, the parentheses ( ), instructs HotDocs to perform the operation inside the parentheses first.

Example HotDocs Models

Simple Markup Example (Contract)

This example contains simple text, number, and date markup fields. Where an answer needs a special format, it has been marked as such. The example also contains a simple conditional statement. (To see a slightly more complex markup example, see Simple Markup Example (Agreement).)

If you're creating a text field, you do not need to include a variable property (te) in the field as HotDocs will automatically assume text as the default. With all other field types, however, you must define a field type property.

```
PUBLISHING CONTRACT

This Publishing Contract, by and between [EmployeeName], whose address is
[EmployeeAddress], and Hobble Creek Publishing, is entered into on
[AgreementDate;da].

Hobble Creek Publishing is in the business of publishing books and desires to
publish [EmployeeName]'s novel entitled [NovelTitle]. [EmployeeName] desires to
have [NovelTitle] published by Hobble Creek Publishing.

[EmployeeName] hereby grants to Hobble Creek Publishing the right to print,
publish, distribute, and sell [NovelTitle] in the United States and Canada in
various editions, including hardbound, trade paperbound, mass market
paperbound, and special edition.

Hobble Creek Publishing agrees to publish [NumberOfCopies;nu] hardback copies
of [NovelTitle] during its initial printing. Numbers for additional
publications will be determined at a later date, based on the profitability of
the initial printing.

[EmployeeName] and Hobble Creek Publishing agree that Hobble Creek Publishing
shall be under no obligation to publish [NovelTitle] provided that this
Contract shall be terminated. All rights granted herein to Hobble Creek
Publishing shall revert to [EmployeeName] if [NovelTitle;te] is not published
by [PublishingDeadline;da].

[if:NoObligationToPublish:In addition, if Hobble Creek Publishing does not
publish [NovelTitle] by [PublishingDeadline;da], Hobble Creek Publishing agrees
to pay [EmployeeName] $[KillFee;nu].

]

_____

James Tyson

Hobble Creek Publishing
```

```
_____________________________

[EmployeeName]
```

Simple Markup Example (Agreement)

This markup example is slightly more complex than Simple Markup Example (Contract). It contains all of the variable types (text, number, date, true/false, multiple choice, and computation). All variables are fully marked using their field type.

Because this example doesn't incorporate variable tables, you'll note additional field properties for some variables, such as multiple choice options and merge text as well as a simple script for a computed field.

```
EMPLOYMENT AGREEMENT

This Employment Agreement, by and between Hobble Creek Publishing and
[EmployeeName;te], is entered into this [AgreementDate;da;format="dth Day of
Mn, YYYY"].

As of [HireDate;da], Hobble Creek Publishing employs [EmployeeName;te], and
[EmployeeName;te] accepts employment, as a full-time [JobTitle;te]. Job duties
shall include [JobDuties;te]. [EmployeeName;te] shall be paid
$[MonthlySalary;nu] per month, which is equivalent to $[Yearly
Salary;co;script=MonthlySalary * 12] per year. Salaries are paid monthly, on
the last business day of the month.

[EmployeeName;te] shall be entitled to a paid annual vacation of
[NumberOfVacationDays;nu;format=alpha] ([NumberOfVacationDays;nu]) days each
year during the continuation of this agreement. Vacation time must be taken in
the year earned. In addition to vacation time, [EmployeeName;te] may take the
following paid holidays: New Year's Day, Martin Luther King, Jr., Day,
President's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving (plus
one additional), and Christmas (plus one additional).

[if:PaidSeminarDays:In addition, [EmployeeName;te] shall be allowed
[NumberOfSeminarDays;nu] days each year to attend professional meetings or
seminars, provided that [EmployeeGender;mc;options=male/female;merge=he/she]
plans attendance at such meetings or seminars for minimum interference with the
business of Hobble Creek Publishing.

][if:TrialPeriod:The length of [EmployeeName;te]'s employment will be an
initial term of six months, with the possibility of continuation beyond that
period depending on Hobble Creek Publishing's needs and upon
[EmployeeGender;mc;options=male/female;merge=his/her] performance.

][EmployeeName;te]'s employment with Hobble Creek Publishing is "at will." The
terms of employment are subject to change at Hobble Creek Publishing's
discretion with advance written notice.
```

```
_____________________________

[CompanyRepresentative;te]

Hobble Creek Publishing

_____________________________

[EmployeeName;te]
```

Complex Markup Example with Tables (Last Will and Testament)

The following example shows a marked up HotDocs Model that incorporates variable tables.

Colors have been used to distinguish variable fields, If fields, and Repeat fields from the rest of the document text.

To see an example of a HotDocs Model that does not use variable tables, click here.

LAST WILL AND TESTAMENT

OF

[ClientName;format=upper]

```
I, [ClientName;format=upper], of [ClientCityOrCounty], Washington, being of
sound and disposing mind, memory, and understanding, do hereby make, publish
and declare this to be my Last Will and Testament (this “Will”), hereby
revoking all prior wills, codicils, or other testamentary dispositions made by
me.

[if:ClientIsMarriedOrHasChildren:

ITEM ONE: IDENTIFICATION

[if:ClientIsMarried:As used in this Will, the word “spouse” refers to
[SpouseName;format=upper]. ][if:ClientHasChildren:I have
[NumberOfChildren;format=alpha][ClientHasOneChild;merge=child/children] living
on the date of execution of this will, [repeat:ChildList;format="a, b, and
c":[ChildName;format=upper]]. As used herein, the terms “child” and “children”
shall refer to any of my children named herein as such and any children
subsequently born to or adopted by me. As used herein, the term “my
descendants” shall include all children. A person who has a relationship by or
through legal adoption shall take under this will as if the person had the
relationship by or through birth.]]

ITEM TWO: FUNERAL PROVISION
```

I direct my Personal Representative to pay my funeral expenses, including the cost of a suitable marker for my grave, the cost of cremation by a reputable funeral director, and/or the expenses of any memorial service as my Personal Representative may deem appropriate, to be paid from the principal of my residuary estate, free of any limitation or restriction imposed by law with respect to the amount thereof and without the necessity of an order of court.

ITEM THREE: PAYMENT OF TAXES

I direct that all taxes due by me, all federal and state inheritance and estate taxes due and payable by reason of my death, and all expenses of administration of my estate, shall be payable out of the principal of the rest and residue of my estate, and no one shall be required or called upon by my Personal Representative to contribute to the payment of any such taxes.

ITEM FOUR: TANGIBLE PERSONAL PROPERTY

(A) I give and bequeath all automobiles, equipment and machinery, furniture, chinaware, silverware, household furnishings, books, pictures and other similar objects, all clothing, jewelry and all other tangible personal property which I may own at the time of my death in accordance with the Personal Property Memorandum. To the extent this property is not disposed of by such writing, this property shall pass as part of my Residuary Estate.

(B) I direct that any and all expenses related to the maintenance and storage of such tangible personal property after my death, and the transportation and delivery of such property to the beneficiary or beneficiaries, shall be borne by my estate.

ITEM FIVE: DISPOSITION OF RESIDUARY ESTATE

[if:ClientIsMarried: (A) Disposition to Spouse. After the payment of all expenses of administration and other charges payable from my estate, I hereby give, devise and bequeath all the rest, residue and remainder of my estate, real, personal or mixed, wheresoever situated and howsoever acquired (my “Residuary Estate”) unto my spouse, if my spouse survives me.

(B) Disposition to Children and Descendants. If my spouse predeceases me, then my Residuary Estate shall be divided into a sufficient number of equal shares, if more than one (1), so that there shall be set apart therefrom one (1) such share for each of my children as shall survive me, and one (1) such share for the descendants (as a group) of each child of mine who predeceases me, but of whom one (1) or more descendants shall survive me. Any share of my Residuary Estate to which the descendants (as a group) of a deceased child of mine shall be entitled as hereinabove provided shall be further divided into equal shares, so that there shall be set apart therefrom one (1) such share for each child of such deceased child of mine. Subject to ITEM SEVEN of this Will, each share of my Residuary Estate shall be distributed to my respective beneficiaries free of trust.

(C) Alternate Disposition. If none of my spouse, children, or other descendants survive me, then my Residuary Estate shall be distributed, free of trust, to [repeat:BeneficiaryList;format="a, b and c":[BeneficiaryName;format=upper]].

/else: (A) After the payment of all expenses of administration and other charges payable from my estate, I hereby give, devise and bequeath all the rest, residue and remainder of my estate, real, personal or mixed, wheresoever situated and howsoever acquired (my “Residuary Estate”) [if:ClientHasChildren:to my descendants who survive me. If none of my descendants survive me, I give the remainder of my real and personal estate (hereinafter referred to as my “Residuary Estate”)][if:BeneficiaryIsIndividual:[if:MoreThanOneBeneficiary:in equal shares]to [repeat:BeneficiaryList;format="a, b and c":[BeneficiaryName;format=upper]]. But if [if:OneBeneficiary:[BeneficiaryGender;merge=he/she]/elseif:TwoBeneficiaries:either /else:any such individual] predeceases me leaving descendants who survive me, [if:OneBeneficiary:my Residuary Estate/else:that individual’s share] shall pass to such descendants[if:MoreThanOneBeneficiary:. If [if:TwoBeneficiaries:either/else:any] such individual predeceases me leaving no descendants who survive me, then that individual’s share shall pass [if:TwoBeneficiaries:to the other individual who survives me or, if the other individual predeceases me, then to the descendants who survive me, collectively, of that other individual/else:in equal shares to each such individual who survives me and to the descendants who survive me, collectively, of each such individual who predeceases me leaving descendants who survive me]]/else: to [BeneficiaryCharityName;format=upper], or its successor in interest].

(B) Whenever property is to be distributed to the descendants of a person (the “ancestor”), such property shall be divided into equal shares, one share for each then living descendant in the first generation below the ancestor in which at least one descendant is living, and one share for each deceased descendant in such generation who has a descendant then living. Each share created for a living descendant shall be distributed to such descendant. Each share created for a deceased descendant shall be divided and distributed according to the directions in the two preceding sentences until no property remains undistributed.

(C) A person who has a relationship by or through legal adoption shall take under this will as if the person had the relationship by or through birth.

ITEM SIX: FAILURE OF BENEFICIARIES

Any property of my estate maturing for ultimate and absolute distribution in respect to which there is no one then living and qualified to take under the foregoing provisions hereof shall be distributed to such person or persons as would have been entitled to receive my estate, and in the same proportions as they would have taken, had I died immediately following the time as of which there is no taker, intestate, unmarried, domiciled in the State of Washington, and the absolute owner of such share, portion of a share or other property then to be disposed of, as the case may be.

ITEM SEVEN: TRANSFERS TO MINORS

If any beneficiary entitled to a share of my estate under the foregoing provisions of this Will shall not have attained the age of twenty-one (21) years at the time for outright distribution of such share, then my Personal Representative shall distribute the share which such beneficiary would have

been entitled to receive to such person (including any Personal Representative of mine) as my Personal Representative shall determine, to be held by such person as custodian for such beneficiary under the Washington Uniform Transfers to Minors Act until such beneficiary attains the age of twenty-one (21) years.

ITEM EIGHT: APPOINTMENT OF PERSONAL REPRESENTATIVE

(A) I hereby nominate and appoint [PersonalRepresentativeName;format=upper] to serve as Personal Representative of my estate under this, my Last Will and Testament.[if:AlternatePersonalRepresentativeNominated: If my above-named Personal Representative predeceases me, or survives me and is unable or unwilling for any reason to serve as my Personal Representative, then I hereby nominate and appoint [AlternatePersonalRepresentativeName;format=upper] to serve as my Personal Representative in his/her place and stead.]

(B) I direct that my Personal Representative herein named be excused from the necessity of giving bond.

(C) I confer upon my Personal Representative all powers necessary, proper or convenient for the preservation, management and direction of my estate, and I authorize my Personal Representative to buy, sell, assign, transfer, convey, exchange, divide, invest, reinvest, hypothecate, pledge, mortgage, borrow, lend (with or without security), lease, release, deed, grant options, compromise, arbitrate, consent to or oppose reorganizations, consolidations, mergers or partitions, select depositories for the funds of my estate and from time to time to deposit therein the funds of the estate, employ and pay counsel, and otherwise to deal with the whole or any portion of my estate, real and personal, as my said Personal Representatives may deem to be proper and advantageous to my estate, and to that end, to make contracts, deeds, conveyances, leases, releases, transfers and other instruments in writing, and to receive payment and to do all other acts and things incident thereto, all of which powers shall be exercised without the necessity of prior application to or subsequent ratification by any court.

(D) I further confer upon my Personal Representative all powers necessary, proper or convenient, without filing reports with any court, to continue, incorporate, enter into, or operate any business, whether as a stockholder, general or limited partner, sole or joint owner, or otherwise; to invest whatever assets may be needed in the business; to employ agents to operate the business; to serve in any capacity with the business; to receive reasonable compensation for such services, in addition to compensation for services as a fiduciary; and to reorganize, liquidate, merge, consolidate, or transfer the business or any part of it.

(E) It is my intention, and I hereby declare, that the mention of the above powers is not intended to be a limitation upon the exercise of other powers by my Personal Representative, but that my Personal Representative shall have all powers which my Personal Representative may deem to be necessary, proper or convenient for the advantageous administration of my estate and to carry out the purposes of this Will.

(F) My Personal Representative, while acting in good faith, shall not be liable or held responsible for any loss.

ITEM NINE: MISCELLANEOUS

(A) Governing Law. This Will shall be interpreted pursuant to the laws of the State of Washington, without regard to its conflict of laws principles.

(B) Headings. The item and paragraph headings are for convenience only, and shall have no bearing on the construction or interpretation of this Will.

(C) Severability. If any court of competent jurisdiction or any regulatory agency shall at any time invalidate any of the separate provisions of this Will, such invalidation shall not be construed as invalidating the whole of this Will, but only the separate provision or provisions in controversy. All of the remaining provisions shall be undisturbed as to their legal force and effect.

(D) Context. Gender, singular, or plural shall be interpreted as the context requires.

[if:GuardianNominated:

ITEM TEN: GUARDIAN FOR MINOR CHILDREN

In the event it becomes necessary for a guardian of person and property to be appointed for any child of mine, then I constitute and appoint [GuardianName;format=upper] to serve as guardian of the person and property of any child of mine. [if:AlternateGuardianNominated:Should [GuardianName;format=upper] fail or cease to serve, I name [AlternateGuardianName;format=upper] to be the guardian of the person and property of any child of mine.]I request that any guardian designated herein or pursuant to the provisions hereof be allowed to qualify as such without being required to give security on any required fiduciary bond.

]

IN TESTIMONY WHEREOF, I have hereunto subscribed my name and affixed my seal this [WillExecutionDate;format="dth day of Mn, y";unans="_____ day of _____, _____"].

_____(SEAL)

[ClientName;format=upper]

The foregoing instrument was SIGNED, SEALED, PUBLISHED and DECLARED by [ClientName;format=upper], as and for [ClientGender;merge=his/her] Last Will and Testament, in the presence of the undersigned, who, at [ClientGender;merge=his/her] request, in [ClientGender;merge=his/her] presence and in the presence of each other, hereunto subscribe our names as attesting witnesses.

_____

Signature of First Witness

[WillWitness1Name]

Address:

[WillWitness1Address]

_____

Signature of Second Witness

[WillWitness2Name]

Address:

[WillWitness2Address][enddocument]

TEXT VARIABLES

Name	Prompt	Resource	Additional
ClientName	Client full name		
ClientCityOrCounty	City or county of residence (e.g., Falls City, Emory County, Sanpete County)		
SpouseName	Spouse full name		irrel=hide
ChildName	Name of child		
PersonalRepresentativeName	Name of personal representative		
AlternatePersonalRepresentativeName	Name of alternate personal representative		
BeneficiaryName	Name of beneficiary		
BeneficiaryCharityName	Name of beneficiary		irrel=hide
GuardianName	Name of guardian		irrel=hide
AlternateGuardianName	Name of guardian		irrel=hide
WillWitness1Name	Name of first witness		warn=no
WillWitness1Address	Address		warn=no;height=4
WillWitness2Name	Name of second witness		warn=no
WillWitness2Address	Address		warn=no;height=4

DATE VARIABLES

Name	Prompt	Resource	Additional
WillExecutionDate	Date will is executed		warn=no

TRUE/FALSE VARIABLES

Name	Prompt	Resource	Additiona
ClientIsMarried	Is the client married?		
ClientHasChildren	Does the client have children?		
GuardianNominated	Does the client want to nominate a guardian for minor children?		
AlternateGuardianNominated	Does the client want to nominate an alternate guardian?		
AlternatePersonalRepresentativeNominated	Does the client want to nominate an alternate personal representative?		

MULTIPLE CHOICE VARIABLES

Name	Prompt	Resource	Options
ClientGender	Gender of Client		ClientIsMale/ClientIsFemale
BeneficiaryType			BeneficiaryIsIndividual/BeneficiaryIsChar
BeneficiaryGender			BeneficiaryIsMale/BeneficiaryIsFemale

COMPUTATION VARIABLES

Name	Script

ClientIsMarriedOrHasChildren	ClientIsMarried or ClientHasChildren
ClientIsMarriedAndHasChildren	ClientIsMarried and ClientHasChildren
ClientHasOneChild	count(ChildList) = 1
NumberOfChildren	count(ChildList)
BeneficiaryIsIndividual	BeneficiaryType = "BeneficiaryIsIndividual"
OneBeneficiary	count(BeneficiaryList) = 1
TwoBeneficiaries	count(BeneficiaryList) = 2
MoreThanOneBeneficiary	count(BeneficiaryList) = 1

DIALOGS

Name	Contents
ClientInformation	ClientName
	ClientGender
	ClientCityOrCounty
	ClientIsMarried
	SpouseName
	ClientHasChildren
ChildList	ChildName
PersonalRepresentativeInformation	PersonalRepresentativeName
	AlternatePersonalRepresentativeNominated
	AlternatePersonalRepresentativeName
GuardianInformation	GuardianNominated
	GuardianName
	AlternateGuardianNominated

BeneficiaryInformation

BeneficiaryList

AlternateGuardianName

"Identify here the beneficiaries who are to receive the residue of the estate"

BeneficiaryType

BeneficiaryCharityName

" "

BeneficiaryList

" "

BeneficiaryGender

BeneficiaryName

INTERVIEW

ClientInformation
ChildList
BeneficiaryInformation
PersonalRepresentativeInformation
GuardianInformation
ExecutionInformation

Quick Reference Guides

Quick Reference—Field Types

The following table describes the different types of fields you can mark in a HotDocs Model.

Field Type	Parts of the Field	Examples
Variable Field	Variable name Variable type Variable properties (optional)	[ClientName;te;format=upper][ClientAge;nu;format=alpha][ClientBirthDate;da;format=dd Mn, YYYY]
Conditional Text Field	If / Else If / Else keyword Variable type Variable properties (optional)	[if:PaidSeminarDays:Document Text][if:ClientIsMarried:Document text/elseif:ClientIsDivorced: Document text/else:Document text]
Repeated Text Field	Repeat instruction Repeat field properties (optional) Document text you want repeated	[repeat:ChildrenInformation; format="a, b, and c";ascend=ChildAge; filter=Minors:Document text]
Comment Field	Comment instruction Comment text	[comment:Calculates the monthly average]
Ask Field	Ask instruction Variable name	[ask:EmployeeStartDate]
Default Field	Default instruction Variable name	[default:HireDate="2 Nov 2008"]

Set Field	Set instruction Variable name	[set:MaritalStatus=Single]
Increment Field / Decrement Field	Increment or Decrement instruction Number variable name	[increment:TempNum][decrement:TempNum]
Insert Field	Insert instruction Template file name Keep property with header or footer value (optional)	[insert:"healthcaredirective.docx"; keep=header]
Assemble Field	Assemble instruction Template file name	[assemble: "healthcaredirective.docx"]
Language Field	Language Instruction Language Code • eng (for English) • deu (for German) • des (for Swiss German) • dea (for Austrian German) • fra (for French) • nld (for Dutch) • esn (for Spanish) • ita (for Italian) • ptb (for Brazilian Portuguese) • Decimal (optional) Grouping (optional)	[language:fra; decimal=','; grouping='.']
Span Field	Span instruction	[span:ExclusionaryClause; title=Exclusions: Document Text]

Title (optional)

To apply colors to the fields in your HotDocs Model, click the **Apply Color** button on the HotDocs Markup Tools.

You can assign a comment property to any of the above-named fields.

Quick Reference—Field Properties

The following tables describe properties you can assign to the different field types in your HotDocs Models. For a full description of each property, see Define Field Properties. Default values are in parentheses.

Text Variable Properties

Property Name	Value	Where It Can Be Used
name	text	Variable field
title	text	Variable field
		Variable table
prompt	text	Variable field
		Variable table
resource	text	Variable field
		Variable table
format	(none)	Variable field
	upper	Variable table
	lower	
	title	
	sentence	

nonbreak	(no)	Variable field
	yes	Variable table
height	(0)	Variable field
	Number between 1 and 12	Variable table
	By default, HotDocs displays one-line answer fields	
max	(0)	Variable field
	Number between 1 and 15,000	Variable table
	By default, HotDocs lets you enter any number of characters between 1 and 15,000.	
pattern	(none)	Variable field
	(999) 999-9999	Variable table
	999-99-9999	
	99:99	
	99:99am	
enter	(break)	Variable field
	paragraph	Variable table
ask	(yes)	Variable field
	no	Variable table
warn	(yes)	Variable field
	no	Variable table
save	(yes)	Variable field
	no	Variable table

irrelevant	(gray)	Variable field
	hide	Variable table
	show	
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field
	Font Name	
comment	text	Variable field
		Variable table

Number Variables

Property Name	Value	Where It Can Be Used
name	text	Variable field
title	text	Variable field
		Variable table
prompt	text	Variable field
		Variable table
resource	text	Variable field
		Variable table
format	alpha	Variable field
	ordinal	Variable table
	09	

	9 1/8	
	9,999.00	
	9.9	
	9999	
	9th	
	IX	
	Nine Dollars and Twelve Cents	
nonbreak	(no)	Variable field
	yes	Variable table
minimum	(0)	Variable field
	any number	Variable table
maximum	(0)	Variable field
	any number	Variable table
decimal	(0)	Variable field
	any number between 1 and 7	Variable table
currency	($)	Variable field
	£	Variable table
	€	
	DM	
	Any currency (3 characters or less)	
ask	(yes)	Variable field
	no	Variable table

warn	(yes)	Variable field
	no	Variable table
save	(yes)	Variable field
	no	Variable table
irrelevant	(gray)	Variable field
	hide	Variable table
	show	
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field
	Font Name	
comment	text	Variable field
		Variable table

Date Variables

Property Name	Value	Where It Can Be Used
name	text	Variable field
title	text	Variable field
		Variable table
prompt	text	Variable field
		Variable table
resource	text	Variable field

		Variable table
format	d	Variable field
	dd	Variable table
	dth	
	dy	
	m	
	mm	
	mn	
	mnt	
	y	
	yy	
	yyyy	
	yr	
	wd	
	wdy	
nonbreak	(no)	Variable field
	yes	Variable table
ask	(yes)	Variable field
	no	Variable table
warn	(yes)	Variable field
	no	Variable table
save	(yes)	Variable field

	no	Variable table
irrelevant	(gray)	Variable field
	hide	Variable table
	show	
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field
	Font Name	
comment	text	Variable field
		Variable table

True/False Variables

Property Name	Value	Where It Can Be Used
name	text	Variable field
		Variable table
title	text	Variable field
		Variable table
prompt	text	Variable field
		Variable table
resource	text	Variable field
		Variable table

format	/x	Variable field
	true/false	Variable table
	x/	
	yes/no	
	truetext/falsetext	
style	(row)	Variable field
	column	Variable table
nonbreak	(no)	Variable field
	yes	Variable table
ask	(yes)	Variable field
	no	Variable table
warn	(yes)	Variable field
	no	Variable table
save	(yes)	Variable field
	no	Variable table
irrelevant	(gray)	Variable field
	hide	Variable table
	show	
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field

	Font Name	
comment	text	Variable field
		Variable table

Multiple Choice Variables

Property Name	Value	Where It Can Be Used
name	text	Variable field
title	text	Variable field
		Variable table
prompt	text	Variable field
		Variable table
resource	text	Variable field
		Variable table
format (multiple-select variables)	a, and b	Variable field
	a, b	Variable table
	a, b and c	
	A, b and c	
	A, B AND C	
	A, B and C	
	a, b or c	
	a, b, and c	
	a; b; and c	

format (single-select variables)	upper	Variable field
	lower	Variable table
	title	
	sentence	
options	text	Variable field
		Variable table
optionprompts	text	Variable field
		Variable table
optionresources	text	Variable field
		Variable table
merge	text	Variable field
		Variable table
select	(single)	Variable field
	multiple	Variable table
other	(no)	Variable field
	yes	Variable table
none	(no)	Variable field
	yes	Variable table
style	(dropdown)	Variable field
	(grid)	Variable table
	column	

	list	
nonbreak	(no)	Variable field
	yes	Variable table
ask	(yes)	Variable field
	no	Variable table
warn	(yes)	Variable field
	no	Variable table
save	(yes)	Variable field
	no	Variable table
irrelevant	(gray)	Variable field
	hide	Variable table
	show	
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field
	Font Name	
comment	text	Variable field
		Variable table

Computation Variables

Property Name	Value	Where It Can Be Used
name	text	Variable field

script	A valid HotDocs script	Variable field
		Variable table
merge	text	Variable field
		Variable table
		This property can be used when the Computation field generates a true/false value.
nonbreak	(no)	Variable field
	yes	Variable table
unanswered	Any text you enter	Variable field
		Variable table
font	(font used in document)	Variable field
	Font Name	
comment	text	Variable field
		Variable table

Dialogs

Property Name	Value	Where It Can Be Used
name	text	Dialog table
title	text	Dialog table
resource	text	Dialog table
contents	names of variables, dialogs, and/or dialog text variables	Dialog table
group	(none)	Dialog table
	single	
	multiple	

style	(regular)	Dialog table
	repeated	
	spreadsheet	
	ssonparent	
label	text	Dialog table
prompt	text	Dialog table
rows	(3)	Dialog table
	number	
ask	(yes)	Dialog table
	no	
irrelevant	(hide)	Dialog table
	show	
none	(no)	Dialog table
	yes	

Repeat Fields

Property Name	Value	Where It Can Be Used
name	text	Repeat field
		Dialog table
format	a, and b	Repeat field
	a, b	Dialog table
	a, b and c	
	A, b and c	

	A, B AND C	
	A, B and C	
	a, b or c	
	a, b, and c	
	a; b; and c	
title	text	Repeat field
		Dialog table
resource	text	Repeat field
		Dialog table
contents	names of variables, dialogs, and/or dialog text variables	Repeat field
		Dialog table
group	(none)	Repeat field
	single	Dialog table
	multiple	
style	(regular)	Dialog table
	repeated	
	spreadsheet	
	ssonparent	
label	text	Repeat field
		Dialog table
prompt	text	Repeat field

		Dialog table
rows	(3)	Repeat field
	number	Dialog table
ask	(yes)	Repeat field
	no	Dialog table
irrelevant	(hide)	Repeat field
	show	Dialog table
none	(no)	Repeat field
	yes	Dialog table
ascend	VariableName	Repeat field
		Dialog table
descend	VariableName	Repeat field
		Dialog table
filter	ComputationVariableName	Repeat field
		Dialog table
comment	text	Repeat field
		Dialog table

Insert Fields

Property Name	Value	Where It Can Be Used
keep	header	Insert field

	footer	
	both	
comment	text	Insert field

Span Fields

Property Name	Value	Where It Can Be Used
title	text	Span field
comment	text	Span field

Language Fields

Property Name	Value	Where It Can Be Used
decimal	decimal separator character	Language field
grouping	thousands separator character	Language field