Standards and Practices for HotDocs Server Applications in Legal Services

Draft revision as of March 23, 2011

Prepared and maintained by Capstone Practice Systems[1]

This is an evolving document. Corrections and additions are most welcome.
Send them to marc@capstonepractice.com

Contents

[1] Contributors have included Bob Aubin, Kate Bladow, Bart Earle, Sheila Fisher, Marc Lauritsen, Roland Monson, Jim Robertson, Alan Soudakoff, and Christina Stommel.

Purpose and scope of this document

This document is a reference guide – a compendium of know-how and advice – for people building and maintaining HotDocs applications intended for delivery to legal services advocates and clients over the Web. Information, guidance, and suggested standards are included. This is necessarily a work in progress, and it may frequently be updated.

This is *not* a general guide to all aspects of HotDocs template development, some of which (clause libraries, database connections) are not presently relevant to the online mode. This is also *not* a guide to the broader field of document assembly, such as the pros and cons of alternative software platforms and approaches.

While this document is intended for the nonprofit legal services community, many of the topics covered are relevant to people doing HotDocs work in other contexts, and we encourage them to make use of these materials and contribute their own suggestions. This is the first significant effort we know of to establish standards for document assembly applications across multiple organizations and jurisdictions. We hope that some aspects of this work will become de facto standards *outside* legal services, too.

Application standards

General principles

The following notes include a mixture of suggested standards, design principles, and "tips and tricks." It is unrealistic and counterproductive to proliferate standards at too great a level of detail, but developers should be aware of options, considerations, and tradeoffs.

Some diversity is inevitable and healthy. Some coordination is essential. The trick is to find an appropriate tradeoff of rigidity and flexibility. We don't want standards for their own sake, but we also want to minimize needless reinvention and technical inconsistencies.

There are several clear benefits of standards:

- Ease of use and effectiveness of tools for the end user
- Lower cost of template development
- Lower cost of template maintenance
- Data consistency (e.g., so users can use the same answer files with different templates, but also eventually work smoothly with things like electronic court filings)

And there are important questions of *scope* (possibly differing on a standard-by-standard basis). Should your standards apply:

- Across template sets or practice units within one office?
- Across offices?
- Across states and user populations?
- Retroactively?

However you come out on specific points, it is valuable to be consistent at least within and across your own templates!

For an overview of standards and conventions generally, see Training Materials: Developing Templates for LawHelp Interactive: Recommended Conventions and Practices (2010)

All of the points below apply to applications written for use in HotDocs Server. (Some different issues and suggestions obtain for templates intended for classical HotDocs desktop usage.) It is assumed that developers are using HotDocs 10 or later for editing.

User interface standards

In general

- Keep it simple
- Be concise
- Don't ask for the same information more than once
- Don't ask for information that turns out not to be needed

In particular

See notes on prompts, dialog layouts, helps, etc. in below materials.

There are many considerations of *usability*, including some peculiar to specific *kinds* of users (e.g., self-helpers vs. advocates) that are not yet addressed here.

Variables

Naming

See **Naming HotDocs Components** below for recommended variable naming standards.

Choice of variable types

Often there is more than one kind of variable to accomplish a given purpose. Here are some considerations bearing on your choices.

True-False vs. Multiple Choice. A related group of choices can be implemented either way. Using a single MC variable saves on your component count and provides some built-in formatting mechanics. Using TF variables gives you more flexibility in presentation and reuse.

Number vs. Text. Don't use number variables for zip codes or other "numbers" that will never be added or otherwise numerically manipulated.

Multiple choice expression vs. Computation. When frequently testing for a multiple choice option (IF Landlord entity type MC = "corporation"), use a computation (Landlord is corporation CO). That way if your choice language changes, you'll only need to edit one component, not every context the test is used.

Multiple Choice vs. Computation. For pronouns and other simple computed phrases, consider using computations rather than MCs with merge texts, for better readability of template. (Some prefer MC variables even though reading is difficult because it can be hard to name the CO variables in large templates. You can also use the ability to name Merge Text sets to keep the merge text short.)

Expression vs. Computation. If template expressions are composed of more than 2 or 3 conditions, consider using a computation instead to reduce clutter on the face of your template.

Date variables. Most dates should be implemented as true HotDocs date variables. That prevents illegal dates like September 31, enables computations like ages, and gives you the nice pop-up calendar option. But use text variables for date fields that don't clearly require a specific date, so that users can enter approximate dates (like month and year only, or a range). This is pretty common for dates of past employment, etc.

Also, consider using text variables instead of date variables when users may not know in advance the exact date (e.g. closing date). This may not be possible if the date is used in a date computation (e.g. 10 days after closing date)

Standalone years are probably best implemented as multiple choice when the possible answers are in a short range, since when a 4-digit number is entered via a true number variable, a comma appears non-optionally during entry (even though it can be suppressed in the document via a format). This can alternatively be dealt with by an appropriate Prompt or a two-digit year with a computation to convert to four digit (prompt should disclose what "cut off year" you've programmed to determine the right century).

Format

HotDocs will automatically include formats in inserted variable fields based on text you replace with a variable during template editing. Don't keep the `Like This` format taken from pre-existing text if in the future users might need to enter text in a different format (like capital letters other than at the beginning of words, for example "McDonald").

Use default merge, formatting, example choices where all or the vast majority of instances of the variable will use them, but avoid using otherwise. Rationale: The default does not appear explicitly in the template, so it may complicate testing, debugging and maintenance to use them. On the other hand, if the variable is always going to appear with the default, the efficiency gained by not having to specify the merge, formatting or example may outweigh the cost.

Answer validation

Use minimum and maximum settings for numbers when possible. They reduce user errors.

Answer patterns

Use patterns when a proper answer will always follow a certain pattern, such as for

- social security number – 999-99-9999
- time - 99:99 x.m.

Don't use patterns for things like telephone numbers when you expect many variations with international dialing, extensions, etc. Be aware that answers saved from template versions in which a pattern is present for a variable may not act as expected when the pattern is absent, and vice versa.

Prompts

Do:

- use “complete the sentence” or similar language or provide example answers when the style or syntax of response is not obvious
- use the “$” choice in currency variables so that users understand when dollar amounts are involved

Avoid:

- colons in prompts
- long questions
- redundant words in prompts such as “Plaintiff last name,” “Plaintiff first name,” and “Plaintiff middle initial”

Note that:

- You can type “NONE” in the prompt field when no prompt is desired

Advanced options

Use REQUIREd answers sparingly to maintain flexibility for users. You can always script a custom warning instead.

Specific variable types

true/false

Checking ‘Yes/No on same line’ helps minimize vertical height of dialogs and need to scroll.

multiple choice

Note that the “grid” display options doesn’t display correctly in HotDocs Server.

text

Increase lines to 2 or more if longer answers are anticipated or if intermediate line breaks need to be accommodated (returns entered by user).

Consider instructing the user not to insert a hard return at the end of their answer, as this adds an unwanted line break in the assembled document.

number variables

You can put dollar signs either in a variable's format field or in the body of the template. If the former, it’s easier to get them everywhere needed when replacing a repeatedly used value with variables.

To format fractions you may need to use a computation. Also, be aware of the rounding done on fractions by HotDocs.

In formatting numbers, remember that a 9 will display a digit unless it is a leading zero or a trailing zero after the decimal point. A 0 will display digits even if they are leading or trailing zeros.

Avoid HotDocs's own numbering variables (<PN>) since these are not dynamic in generated documents. Instead use word processor-based auto-numbering, but see caveats below.

telephone numbers

Country code (if needed) and Area code variables should be up to 3-digit numeric fields. Only needed as separate variables in forms that have different text locations for different parts of the number.

You can have a Full telephone number CO computation that combines the above to the extent answered

person name variables

It is usually best to create just a single variable for a person's name, unless components are specifically needed separately, as when preparing a letter salutation (Dear Jim or Dear Mr. Smith) or a form in which parts of the name need to be positioned in separate boxes.[2]

Possible components to be aware of include:

- family, surname(s), last
- given, first

[2] There are times when you need to break out a name into separate pieces, and trying to use a single variable for a name and parse out the components is a huge challenge. The best example is probably a form where the pieces of the name need to be in separate fields. If you want to alphabetize names by last name and first, you also need the component parts. Greetings in letters and like can be handled with a separate variable for just the greeting, but from a user interface perspective, separate variables for name parts may be best.

It is certainly easier for a user to enter a full name in a single field, so user interface considerations may favor a single variable when the component parts are not needed. If you are developing a single template or template set in which only full names are used, it may be tempting to use full name variables. What happens, however, if you later need to add a template or graphical form that requires the separate components of the name? What happens when you want your users to be able to use the same answer file with another template or template set where separate name variables are used? In addition, no matter what you say in your prompt, getting a user to enter a middle initial or middle name can prove challenging unless there is a separate field for it.

Because of these considerations, we favor at least breaking "important" names up into separate variables. By important, we mean client, opposing party, spouse, children, and perhaps others, depending on the context. In our experience, these are the names for which component parts are most frequently needed.

This issue has important ramifications when it comes to sharing templates and trying to develop standardized variable names. Will the standardized variables need to include only separate variables for name parts, full name variables, or both? Our feeling is that it probably needs to include both, which diminishes the usefulness of the standardized names somewhat, but there will be times when a template developer feels compelled to use a single variable for a name.

- middle name(s)
- middle initial(s) (can't always compute easily since person may have multiple middle names)
- given at birth, if different (such as maiden name)
- prefix – Mr, Mrs, Ms, Dr, Hon, etc.
- suffix - Jr, Sr, etc.
- aliases and "other" names

addresses

Addresses should usually be gathered component-wise, and assembled into horizontal strings or vertical lists as required by a particular location or field.

Use two street address variables to accommodate apartment numbers and multi-line street addresses, but check if address line 2 is answered when using it in template or computations.

pronouns

You can implement pronouns as either gender MC variables with merge text, or as computations based on such variables. The latter may take a little more work, but look better on the face of the template.

Dialogs

Naming

See **Naming HotDocs Components** below for recommended dialog naming standards.

Organization

The allocation of variables among dialogs should reflect a logical breakdown of questions that will provide a coherent user experience. Take into account any conditional partitioning of the texts in which they appear, so that irrelevant and unrelated questions are not asked together. Avoid "default dialogs" (single-question windows generated by HotDocs when a variable has not been associated with a dialog).

Consider using "title screens" – opening dialogs that just consist of Additional Text, displaying the template name, credits, and general instructions or scope notes. Closing screens are also useful.

Remember that you can script conditional warning dialogs in case of improper user input.

Layout and content

1. Extra spacing (blank lines) should be used sparingly.
2. Leave prompt alignment in dialogs set to "above fields" in most situations [default]
3. Embedded dialogs may be confusing to user
    a. if used, use ellipsis in prompt
    b. Use @prompt to get a better name in the parent

4. Keep dialogs short enough so they display fully without scrolling when viewed in an IE window at 600x800 resolution.
5. Group True-False variables when possible, to avoid long sets of Yes/No prompts
6. When a T/F variable appears in a group of one, consider alerting users how to clear the selection. (Right click.)

Dialog scripts

LIMIT NUM can be used to prevent users from entering more than a given (fixed, user-supplied, or computed) number of entries in a repeating dialog.

Use HIDE and SHOW rather than GRAY and UNGRAY in most contexts since their use makes for a cleaner interface. But use GRAY and UNGRAY for:

- Cases where the user expects to see subsidiary questions and graying them will add to their understanding of or confidence in the template.
- Cases where HIDE and SHOW cause too much disruptive jumpiness in the dialog.

REQUIREd answers can be powerful, but frustrating for users who may want to temporarily leave something unanswered and return to it later.

Remember that SETting a variable renders it uneditable by a user. To prepopulate but allow user revision, use DEFAULT. But watch out for endless loops caused by defaults in repeated dialogs. (To avoid that, condition the default upon some other variable having been answered.)

Interviews

Use an INTERVIEW computation to speed up moving backwards and forwards in dialogs. The INTERVIEW computation avoids the necessity for putting a computation directly in the template to generate a custom interview and allows you to use the same component file for an interview-only template.

If you create an INTERVIEW computation, be sure to go into the Component File Properties in the Component Manager and check the box for "Use INTERVIEW computation".

Resources

Consider alerting users through prompts of dialog elements to the availability of resources (help).

You can use URLs as helps, and when selected by users their corresponding web pages will pop up.

In the component file settings, under the "Server" tab, you can specify that the resource pane is initially on. You can also disable the ability of the user to hide it. If the resource pane is showing, the help for the dialog will show in the resource pane and the question

mark will not appear. Some developers have felt the need to hide the resource pane, since its size can vary, because of a need for more screen real estate for the interview.

Scripting

Scripts are found in computations, IF expressions, and dialogs. Script-like instructions can also be present in templates.

Negation

Choosing between ! and NOT, both valid syntax for negation, is primarily a matter of readability. For instance, you will probably want to say NOT Landlord is attorney TF, but Landlord entity MC != "Person".

REPEATs

Use whenever there is more than one of a given thing; avoid variables like Child1, Child2, etc. -- even if there are only two possible iterations.

Use LIMIT when you or the user can easily specify a fixed or maximum number of repeats.

IF statements

1. Delete extraneous hard returns (inserted by HotDocs following IF and END instructions) within paragraphs to improve readability. Note that HotDocs will do this automatically for you if you choose the 'Smart' option under Template Development settings.
2. Standardize the location of hard returns and spaces relative to IFs (usually at end of conditional material)
3. Minimize the text included redundantly in each branch
    a. maybe even to the extent of breaking up words, e.g., if first letter capitalized in one branch but not another.
4. Don't use TFvar = TRUE (redundant) or = FALSE (use NOT instead)
5. In general, when dealing with alternate texts for the true and false case of a variable, deal with the true case first (if x ... end; if not x ... end) and use ELSE IF structure in most such cases (if x ... else ... end)
6. Explicitly cover the cases in which variables used in the IF expression may be unanswered, to avoid confusing results in the assembled document.[3]

---

[3] All of the text within the IF statement will be omitted.

This may be an oversimplification, but whenever and wherever you instruct HotDocs to do something if X is the case, and the user fails to give it an answer on the basis of which it can determine whether or not X is true, HotDocs will insert asterisks, underlines, or your other preferred unanswered indication in the document. Both for variables (including computations) and for IF statements.

If you want to generate clean documents even for users who fail to answer key questions (and most of us do), it's thus important to be sure that HotDocs is never totally stumped for what to do.

For computations, a handy way is to initialize them to a value by putting an empty string, a zero, or a FALSE on the first line (for text, number, and TF vars, respectively).

Inserted templates

Inserted templates are useful for modularizing large pieces of templates, or those that are used in more than one template.

There are issues with subtemplates. Section breaks require special care, and can cause problems with page numbering, paragraph numbering, and styles.

If you create an inserted template by selecting a portion of text and have HotDocs automatically create an inserted template, HotDocs will automatically point the inserted template's component file to the master template's component file. If you insert an existing template into another, no automatic pointing occurs.

At a minimum, the inserted template's component file must either contain the variables, dialogs etc., OR be pointed to another component file that contains them. If you do not use the pointing approach, and need to refer to the same components in the master template, then you will need to maintain copies of all shared components in both template's component files.

Here's a recommended approach:

1. If your inserted template is a) inserted only in one master template, AND b) not designed to be assembled separately: Point the inserted template's component file. Having only one place to manage components is simpler.

2. If your inserted template needs to a) be assembled separately, OR b) will be inserted in multiple master templates: Maintain the components in both component files. We recommend this because: 1. Pointing to a component file other than the master or inserted template's component file does not currently work in HotDocs Server. 2. In HDS, when dealing with inserted templates, we recommend using INTERVIEW (not feasible if the inserted component file points and also has to be assembled as a separate document).

For IF statements, you can add an ANSWERED(var) test, ensuring that the expression has a value (false) even if the var is not answered. For example, <<IF ANSWERED(Dependent other TF) AND NOT Dependent other TF>>(None)<<END IF>>

There are other ways to avoid unansweredness, e.g. by defaulting variables to certain values absent user input, or by using the ZERO(num var) instruction for potentially unanswered numbers. But a good place to start is just to put yourself in HotDocs's place and ask "will I always know enough to know what to do here?"

Of course, sometimes some questions are so important that you *want* HotDocs to flag the absence of an answer, not only in the interview, but in the document.

Default answers

There are many ways to prepopulate answers, e.g.

- overlay answer files (not yet usable in HDS)
- variable specific defaults [in the advanced tab]
- logical pre-sets on the face of the template or in computations
- DEFAULT instructions in dialog scripts
- default option for multiple choice variables

Consider including all global defaults in one computation to minimize clutter in your template.

Commenting scripts and computations

It's good practice to include comments in scripts that are particularly tricky. You can use // to place a comment on any line, and the word QUIT to have all subsequent text treated as a comment.

Comments following "//" in IF statements can also be useful in templates.

Word processing issues

Unless your user community is purely WordPerfect based (in which case WPT templates make sense), use RTF for templates, which both Word and WordPerfect can handle. RTF is not rendered exactly the same for all versions of Word and WordPerfect, so if you have a mixed user base you'll want to do a lot of testing.

Use word processing level automation for the following features:

- paragraph numbers
- cross references
- tables of contents
- block protect
- tables for captions, signature lines
- general document format preferences
- fonts
- headers
- margins
- styles

For consistent production of templates from HDS the following should be observed for word processing features:

- Use styles as much as possible
- Only use fonts that are known to ship with all versions of Windows and are installed by default

Be aware that certain word processing features (auto numbering, cross references, automatically generated TOCs, indices, etc.) cannot easily be automatically updated in

documents generated from HotDocs Server. (In desktop mode you can PLAY a macro to do the job.) Since some legal services clients will be unsophisticated word processor users, who might not understand about updating references, it may be better to use HD coding and auto-numbering in such situations (they would be presumably few in number and limited in scope). Or alert the user in a closing dialog that they will need to Update their documents after assembly.

Note that there are also lots of standards issues for the actual forms (apart from their automation). These include document page layout formats, whether names are capitalized or not, where the caption begins, page numbers, etc. The development of these kinds of standards generally springs from a desire to have a consistent "look and feel" among documents originating with the organization. Even outside that context, there are good reasons for adopting such standards. There are at least perceived benefits in having families of documents that typically are prepared together (e.g. multiple documents filed in the same court case, or prepared for the same client) have a similar appearance. If, in the case of litigation-related documents, there are local customs and practices with regard to what documents look like, standards can help insure adherence to those practices.

Here's an example:

FORMATTING FOR NORTH PENN LEGAL SERVICES TEMPLATES:
As of 1/21/2004

Word 98 or more recent

1≅ wide margins all sides (unless necessary to do otherwise)

Time New Roman, 12 point for general text, 16 point for titles (no bold or underline)

Left justification

1-1/2 line spacing for pleadings

Spacing and underlining by using tabs, whenever possible

No automatic paragraphing, unless necessary

All captions in tables

Paragraph formatting: no indentation, .5≅ tabs

Numbered paragraphs B **number at left margin, one tab indent for text**

Non-breaking spaces

HotDocs answers can be formatted as nonbreaking (via check box on text variable editor.)

To insert a "non-breaking space," in Word, hold down [Ctrl] + [Shift] and press [Space] - then entire phrase (e.g. "Mr. Jones") will stay on one line.

You insert a non-breaking space directly in a text computation where, for example, you are concatenating a title with a last name, by using the «.ns» dot code.

Graphical forms

[This is a placeholder for material to be added on standards peculiar to PDF templates, e.g., fonts, answer positioning, overflow handling]

Naming HotDocs Components

Introduction

HotDocs template developers have great freedom in naming the components that make up their applications. There are good reasons, however, to follow standards in naming. This document reviews some of the main choices you have, the considerations in favor of each, and some recommendations. While there are minor issues with other component types[4], this document focuses on *variables* and *dialogs*. And of course there are many other dimensions of standardization that are addressed elsewhere, such as choosing what variables to *have* in the first place, and how best to organize them into data structures. (You can have great component names and still have a programming mess!)

Note: While many of the considerations and recommendations here apply to all versions and modes of HotDocs, this document is intended for an audience of developers who are building templates intended for delivery through HotDocs Server. Clauses and other features not presently supported in HotDocs Server are not addressed.

Why follow standards in component naming?

Standards are useful both to developers and to users.

For *users*, since each answer in an answer file is stored with the name of the associated variable at the time the template is used, consistency of naming across templates and across time will spare them from having to enter the same information more than once, and from having previously entered information suddenly seem to disappear.[5]

For *developers*, standardized naming will improve efficiency and quality. If you get in the habit of following certain naming conventions, you will spend less time thinking up the name of each new variable, and be able to type in the names of existing ones with reasonable confidence, rather than having to look them up all the time or running the danger of creating two slightly different ones for the same underlying data element. You will also be able to copy and re-use components from prior templates or from a "lending library" of components that you or others have created.

---

[4] For instance, you can affirmatively name a Merge Text component, which can be useful to make it less ugly in the template, more intelligible, and more easily found in component lists.

[5] For example, if you use Name of plaintiff in a complaint template and Plaintiff's name in a motion template, a user who assembles a complaint and then tries to generate a motion with the same answer file, will not see the plaintiff name previously provided. The user will be forced to re-enter the name. And if, in a belated effort to be consistent and follow these standards, you then rename both variables to Plaintiff name, the user, by now very puzzled and frustrated, who simply tries to re-generate either document using the saved answer file will again have to re-enter the name. Hence it is not only important to be consistent across templates but to think about a naming scheme early in the development process to avoid renaming variables after answer files have been created. It is possible to write remedial scripts to set previous answers to newly named variables, but that should generally be a last resort.

A good overall naming scheme is one that

- is not too complex,
- is easily followed, and
- can be flexibly applied.

You may understandably depart from standards when

- you've already created a lot of templates with different naming conventions
- you want to share answers with published forms that use different names

Some variables are less important than others to standardize:

- those that are not actually stored in answer files, e.g. computation variables
- those that are idiosyncratic and unlikely to be used in other templates

What you *can* do

For the most part, developers are limited only by their imaginations in naming components. Some people have flexible and ad hoc naming practices. Others have very strong opinions about their methods and follow them compulsively.

For instance, you could name the variable designed to gather a child's name in any of the following ways (and endless others):

- Name of this child
- Child's name
- Name of child
- Child Name
- Child name
- ChildName-t
- Child name TE
- ChNamet
- child_name_txt
- txt Child Name
- tChildName
- strChildName [str for "string"]
- Child TE ["name" assumed]
- tChild
- CHILD Name TE

And you *could* randomly mix different styles for different variables in the same application. E.g., Child Name, StreetAddress TE, birthDate-d, etc. (But please don't!)

What you *can't* do

There are *some* built-in limits to names. For instance,

- Only 50 characters are permitted
- The first character must be a letter
- Certain characters are not permitted at all, such as:
  - . (period)
  - ( )
  - %
  - , (comma)
  - [ ]
  - " (double quote)
  - : (colon)
  - $
- The following characters can't be used when there is a space immediately to their right or left:
  - +
  - >
  - =
  - - (hyphen)
  - <
  - !=
  - *
  - <=
  - /
  - >=
- The following non-alphanumeric characters evidently *can* be used without restriction: ` ~ ! @ # ^ { } | \ ; ? '
- Certain upper case *words* are not permitted, such as those used in HotDocs instructions and expressions (IF, ASK, OR, AND, LIMIT, MONTH, etc.)

Capitalization

Issue

What if any letters in your component names should you capitalize?

Considerations

- Component names are case sensitive, so you need to be consistent to avoid errors and inadvertent duplication.
- Unless you're going to write a separate prompt, use capitalization that looks good on a dialog. (Recall that, by default, the name of a variable becomes the prompt unless you affirmatively specify a prompt.)

Recommendations

- Use "sentence case" for variables (only first word and proper nouns capitalized). E.g., Landlord is Massachusetts corporation TF.

- Use "title case" for dialog names (each significant word initially capitalized). E.g., Terms of the Tenancy.

- Do not use all uppercase words, both to avoid "shouting" and because HotDocs may someday use the word as a new instruction or expression model.

You may want to leave even the first word uncapitalized if it represents actual lowercase text to be used. (E.g., is/are for plaintiff CO.) Some developers also find it useful to distinguish between "behind-the-scenes" variables (e.g., temporary or disposable ones used to handle conversions, parsing, data import) from those that actually appear in documents by beginning the former with a lowercase letter. For instance, parse name CO (behind-the-scenes) and Full name CO (appears in documents).

Spaces

Issue

Should you use spaces between the words in your component names?

Reasons to use spaces

- Names with spaces are easier for people to read in most contexts.
- Names with spaces are less likely to trigger spell check errors.
- You'll want spaces if your variable name will serve as the default prompt.

Reasons not to use spaces

- It is arguably easier to read scripts when each variable is one continuous string of characters. (Makes it easier to distinguish them from instructions. *But* colorization in current HotDocs mostly obviates this point.)
- Some external programs and databases don't allow spaces in variable names, and not having them in HotDocs makes it easier to use consistent names with such programs.

Using *underscores* to separate words arguably achieves decent readability of both templates and scripts, but still requires an affirmative prompt to be written for every variable and is more awkward to type.

Recommendation

- Use spaces, and don't use underscores.

Data type indicators

Issue

Should you include some code at the beginning or end of variable components to indicate their data type? E.g., tChildname, Child name TE, Maturity Date_txt, Temp-n.

Reasons to use indicators

- You can tell instantly what kind of variable one is. [*But* in some developmental contexts an icon or other clue is present to indicate data type, and in the body of a template the name or context usually makes it clear, and if not you can quickly find out.]
- They can make a template set easier to maintain, especially when taking over from another developer.
- They are useful when linking HotDocs answers to fields in an external database because the naming convention helps to match types and avoid errors in HotDocs and provides backward traceability to fields in database tables.
- They may be useful in some contexts when HotDocs answers are referenced in XML files. [Even though HotDocs's own XML answer file format includes data type tags right in context.]
- LexisNexis uses the two letter indicators in most of its many published template sets and recommends the practice in its trainings.

Reasons not to use indicators

- They require entry of a distinct prompt or title, even if the rest of a variable could serve fine as a default prompt. [*But* you can simply copy and paste the name and delete the indicator. And if you use a "lending library" component file, you only need to do this once per variable.]
- They can interfere with readability, especially for non-programmers (e.g., substantive experts reviewing a template.)
- They add to the length of the variable.
- They require a couple extra keystrokes every time you create a variable or type its name.

Recommendations

- Use the following standard abbreviations at the end of variable names, preceded by a space:

    - TE for text
    - DA for date
    - NU for number
    - TF for true/false
    - MC for multiple choice
    - CO for computation
    - Exception: INTERVIEW (the specially named computation for controlling dialog flow)

- Don't use a type indicator like "DI" in dialog names. (You can then distinguish them by their *lack* of one, and their title case capitalization.)

Length

Recommendations

- Make component names only as long as necessary to be meaningful, unambiguous, and reasonably recognizable in various contexts.

Word choice

Issue

What words or symbols and how many of them should you use?

Considerations

You probably want to use names that

- make a decent *prompt* for the question (minus the data type indicator)
- use *common* words
- are as *universal* as possible, so they can be used consistently in other templates, present and future
- *unambiguously* describe their variable to avoid it being used in other contexts with different meanings
- are likely to be *stable*, so as not to orphan existing answers and require template revision

Recommendations

- Avoid plurals and possessives. (Apostrophes in variable names can cause problems with Word's "smart quotes" feature. Ditto for pairs of dashes that Word may turn into an "em" symbol.)

- Avoid prepositions and articles. Use Plaintiff name TE rather than Name of plaintiff TE; use Petitioner is attorney TF rather than The petitioner is an attorney TF

- Use abbreviations sparingly and consistently (unless you are *really* short on characters to name a variable unambiguously, or are creating a temporary variable like Temp NU.)

- Use form names in a variable name only when similar information might be used elsewhere with a different meaning (i.e., not merely because the variable appears to be unique to the form you are currently automating). For example, use Complaint filing date DA, instead of Filing date DA if it is possible in additional forms to have other filing dates.

- Name True/False variables after the optional or conditional section of text they trigger (Breach of warranty of habitability TF), or after the condition or choice they represent (Representative is attorney TF, Impound client address TF).

- Don't use "?" in a True/False variable name.

- Don't waste more time than you save by being *too* compulsive about naming nuances!

Some people like to name computation variables used to produce short alternative texts with those texts separated by "/" followed by a hyphen and a word or phrase that shows what the alternatives depend on. For example:

he/she/it/they-plaintiff CO
is/are-one or several properties CO
s-only one plaintiff CO [for verb suffix]

Often nothing besides the alternatives is needed:

subsidiary/subsidiaries CO

For dialogs specifically,

- Use meaningful names that serve well as titles for display. (Even though you can now specify a separate title for dialogs.)

- Take care not to overuse the word "information".

- Use single words in title of dialogs that repeat.. E.g., Creditor Details

Word sequence

Issue

In what *order* should the words in your name go?

Considerations

Since variables are listed alphabetically in component manager, it is highly recommended to make sure related variables are listed together by using names like Plaintiff phone TE and Plaintiff city MC rather than Phone plaintiff TE or City plaintiff MC.

On the other hand, slavish observation of this rule can result in names that are difficult to read. E.g., Spouse residence mortgage principal NU.

Recommendation

- Put the most significant word(s) first, while paying attention to readability.

Examples of recommended names

Client name TE	Use "client" for the main person to which an answer file relates, even if the variable may sometimes be used in templates intended for pro se use (in which case you can appropriately vary the prompt).
Plaintiff name TE	Note that variables like this may often be set in logic based upon the client's or other parties' roles in a case, rather than directly asked, since a person may be one kind of party in one context and a different one in another.
Landlord is Massachusetts corporation TF	Proper noun is capitalized.
is/are for Plaintiff CO	Exception to rule of capitalizing first letter of first word, because the computed words here will always be all lower case.
Monthly salary NU	Entered directly by the user
Monthly salary CO	E.g., computed from weekly amount
Spouse full name CO	Computed from first, middle, last, prefix, suffix, etc.
Representative is attorney TF	Form of a statement
Creditor Details	A dialog

Standard HotDocs variables for templates that may be invoked from a case management system

Summary

This section defines a set of standardized HotDocs variable name and types for common data elements that play a role both in case management and in document assembly systems. Background and rationale for these standards can be found in *Best practices for templates that will be invoked from case management systems* (March 2010).

Apart from several utility variables noted at the end, variables under this standard observe this pattern: [Primary role, and optionally a Secondary role] [attribute] [variable type designator]. E.g., Client name first TE, Opponent Attorney address street TE.
Lists of supported roles and attributes follow. All combinations are "legal," but certain ones (like Judge Child SSN TE) are unlikely ever to be used.

Roles

Primary roles

A primary role is one played with respect to the factual situation as a whole, not with respect to some other particular role. There can be one or more instances in an answer set; thus the variables will typically be singly indexed in HotDocs.

Role	***Notes***
Client	A person whose situation is the focus of the data in the answer set
Opponent	A person in a proceeding or relationship whose interests are adverse to or at least not necessarily aligned with a client
Applicant	
Plaintiff	
Defendant	
Petitioner	
Respondent	
Judge	

Note – It is a good practice to use variable names that reflect as specific as possible a role (or procedural posture) that an person plays in a template. Thus, even if the plaintiff in a divorce petition template is almost always a client, it is best for variables referring to him/her to use the word Petitioner.

Secondary roles

Secondary roles are to be used only in combination with a primary role, because they always pose the question *whose* _____? For instance, Opponent Child. There can be one or more instances *per primary role* in an answer set; thus the variables will typically be doubly indexed in HotDocs.

Role	***Notes***
Attorney	'Attorney' mostly refers to a lawyer, but might be used for a law student or other representative
Child	
Spouse	
Parent	
Employer	

Attributes

Attribute/Type	***Notes***
address city TE	
address county TE	Some developers may use MC, but could default it to a TE if prepopulated. (Note that US post office has standard code for every county.)
address state MC	Options are those for fifty states, District of Columbia, and territories. Can be used for province in other countries.
address country TE	
address street TE	
address street2 TE	Note no space between 'street' and '2'. Used for a second line between the addressee and the city/state/zip in an address.

address zip TE	Can be used for postal code in other countries.
birth date DA	
email primary TE	
email secondary TE	
employed TF	
gender MC	Options are male and female
health TE	
in military service TF	
name full TE	When name is not gathered in pieces
name alias TE	
name birth TE	One's given name at birth, before any name change.
name former TE	One's immediately prior name, whether or not the birth name.
name first TE	
name last TE	
name middle TE	
name prefix TE	e.g. Mr., Ms., Dr., Hon.
name salutation TE	Dear ____
name suffix TE	e.g. Jr., Sr. This variable can also be used for 'second last names' in Spanish and other cultures.
occupation TE	
SSN TE	
telephone home TE	
telephone mobile TE	
telephone work TE	

Utility variables

Name	***Notes***
Answer set includes answers from a CMS TF	Can be used by a developer to control behavior of HotDocs interview when some data has been passed from a case management system
Variable standard TE	To store the name of the standard being followed (This one can be referenced as 'LHI CMS.')

An illustrative name set

HotDocs variable name	Notes
Client name first TE	
Client name last TE	
Client name middle TE	
Client name salutation TE	
Client name alias TE	
Client name birth TE	
Client address city TE	
Client address county TE	
Client address state MC	

Client address country TE	
Client address street TE	
Client address street2 TE	
Client child type MC	natural, step, or adoptive
Client child name first TE	
Client child name last MC	
Client child name middle TE	
Client child SSN TE	Number in format 999-99-9999
Plaintiff address impound TF	
Plaintiff county residence months NU	
Plaintiff county residence years NU	
Plaintiff marriage number NU	
Plaintiff minor children by another relationship TF	
Plaintiff state residence months NU	
Plaintiff state residence years NU	
Plaintiff employer name TE	
Plaintiff employer telephone TE	
Plaintiff attorney address city TE	
Plaintiff attorney address state MC	
Plaintiff attorney address street2 TE	
Plaintiff attorney address street TE	
Plaintiff attorney address zip TE	
Plaintiff attorney bar number TE	
Plaintiff attorney firm name TE	
Plaintiff attorney name TE	
Plaintiff attorney salutation TE	
Plaintiff attorney telephone business TE	
Plaintiff attorney telephone fax TE	
Plaintiff attorney type MC	

Development processes

Selecting, designing, and planning applications

See *Keys to a Successful Document Assembly Project*, www.capstonepractice.com/keys.pdf, for an overview of considerations involving in the selection, staffing, design, and implementation of a document assembly application.

Particularly for applications intended for widespread use over the Internet, it is important to be clear about intellectual property rights. (Who owns the work product? Do you have clear rights to use the source materials?)

Some design challenges

One basic design question is how much to modularize your templates into computations and inserted subtemplates. There are tradeoffs between embedding complexity in computations or inserts, versus having logic evident on the face of the template. High modularity can increase the cost of development but reduce the cost of maintenance.

Use good names or comments to describe your logic and isolate it. That way people can understand what is happening when reviewing the template and can burrow in if they want to verify the details.

Because HotDocs Server does not support clauses and clause libraries, there is no easy way to allow users to *sequence* selected text modules in online mode. Where such functionality is critical, though, you can prompt for explicit orderings (by number) and use WHILE loops to properly sort selected items.

Separate vs. master component files

Consider the many advantages and disadvantages of "pointed" or master component files. Only a few are mentioned here.

Advantages: One set of variables, dialogs, and other components can be shared across multiple templates, promoting consistency and efficiency. You can change things in one place, and all templates will reflect the change.

Disadvantages: Two people cannot simultaneously edit templates pointing to the same component file. All your eggs are in one basket, and if the master file becomes corrupt, you better have a good backup!

The relatively new Template Manager makes it easier to manage multiple component files, obviating some of the prior advantages of master files.

Documentation

Give a lot of thought to the various kinds of documentation that can be maintained in a project.

- A written "action plan and design notes" is a useful way to get a project team on the same page and record design decisions for posterity. It can clarify purposes, expectations, and specifications.
- A "master variable list" can often usefully be created in advance of actual coding, e.g. using a spreadsheet.
- Template printouts are particularly useful in color since that will show HotDocs codes in blue.
- HotDocs Developer lets you easily create component printouts.
- Short user guides are good places to supply basic guidance, answer frequently asked questions, and pass on tips & tricks. Grab screen shots with Alt-PrintScreen and paste them into your guide.

Other resources

http://www.probono.net/dasupport/ is the official repository of LHI support materials.