



THE PATHAGORAS SYSTEM

© 2020 Innovative Software Products of Virginia, LLC

Document Assembly
Document Automation
Document Management

The Pathagoras System

**Document Assembly
Document Automation
Document Management**

by Innovative Software Products of Virginia, LLC

Pathagoras is a different kind of document assembly/document automation system. It shuns hidden fields and codes. It stores source documents in standard folders and files, without encryption and without restrictions which might otherwise limit use of these documents to just the program. It operates purely on plain text -- the kind of text that a typical word processing operator with no programming experience can easily compose and edit.

The program augments what you have and what you know. It does not attempt to replace it. It implements Word concepts with which you are already familiar, but does so in a new and faster way.

There are few new or foreign concepts introduced. Most are simply standard Word concepts elevated to provide document automation not seen before in a plain-text based system.

We hope that you like what you see. And if you have any questions or concerns, we hope that you will contact us and let us know.

© 2020 Innovative Software Products of Virginia, LLC

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

As an exception, each licensed user of Pathagoras may create (for his or her personal or office use) a single printed copy of this Manual for each license owned.

All references in this Manual to "Word" refer to Microsoft® Word, a product of the Microsoft® Corporation. All references to Windows refer to the Microsoft® Windows® operating system, a product of the Microsoft® Corporation.

Other products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: December 2020 in Yorktown, Virginia

Table of Contents

Foreword	ix
Part I Introduction	1
1 Definitions.....	3
2 Pathagoras in 1...2...3 steps.....	7
3 Design & Work Flow Concepts.....	9
4 Processing vs. Personalizing.....	16
5 Moving from basic to advanced.....	19
6 File and Folder 'Pointers'.....	20
7 Advantages of Plain Text Document Assembly.....	22
8 Document Assembly vs. Document Management.....	23
9 Beyond 'Major' Projects.....	24
10 Help Connections.....	25
11 Upgrading Pathagoras.....	25
Part II Getting Started	27
1 Download and Install.....	28
2 Installation Checklist.....	29
3 'Pathagorizing' Your First Form.....	29
4 Notes and hints.....	30
5 Add Simple Optional block.....	31
6 Default Document Display.....	32
Part III Variables - Definition	35
1 Simple Variables.....	36
2 Multiple Choice Variables.....	37
The Final 'And'	39
Aliases variables)	41
3 Modifying Existing Variables.....	41
4 Existing 'Gold Star' Documents.....	41
5 Emphasis (Bold, Italics, CAPS).....	42
6 Titled variables.....	43
7 Default Value of a Variable.....	44
Part IV Instant Database Screen	45
1 The basics.....	46
2 Scan	48
3 Completing a Variable.....	51

4	Data Records.....	52
5	Advanced Array.....	53
6	Power Tools.....	55
7	More Tools and Settings.....	57
8	Other IDB Functions, Features and Tools.....	61
9	Double-Click Tools.....	62
10	Still More IDB Settings.....	63
11	Multiple Choice Selector.....	65
12	Variables, Length of.....	67
13	Other Uses for IDB.....	68
14	Screen Shots Depicting Screen Features.....	69

Part V Instant Database Functions 73

1	Calendar and Date Math Features.....	74
2	Math and Date Math.....	76
3	In-line Math and Date Math.....	81
4	Date Format Arguments.....	84
5	Auto-Create Incrementing Variables.....	85
6	'=' Equivalency Function (single element).....	85
7	'=' Equivalency Function (*Actors* and their Roles).....	87
8	Concatenation of Variables.....	94
9	<S>pell out Function.....	96
10	<\$>Currency Function.....	97
11	<%> Percent Function.....	99
12	<F>ormat Function.....	100
13	<Fr>action Function.....	101
14	<P>aragraph Function.....	102
15	<.x.> Function Reminders.....	103

Part VI Instant Database Records 105

1	Sharing Data: Instant Database Records.....	106
2	Changing Variable Names in Existing Records.....	111

Part VII *Aliases* 113

1	Creating *Aliases* and Lists.....	115
2	Prepared Lists.....	118
3	Other *Alias* List containers.....	119
4	Presenting *Alias* Lists.....	120
5	*Aliases* and !Groups!.....	122
6	*Aliases* and Administrative Text.....	123

7	*Alias* Files Location.....	124
8	*Alias* Table (Embedded).....	125
9	*Aliases* as DropDown Lists.....	126
10	Sharing Your Lists with the World.....	126
11	Miscellaneous *Alias* information.....	127
12	Document Assembly with *Aliases*.....	129
13	'Sentence' Assembly.....	130
14	'Aliases as Actors' List.....	133
Part VIII Groups and !GroupNames!		135
Part IX DropDown Lists		139
1	Creating 'Free Hand'.....	141
2	Adding Content to DropDown Lists.....	144
3	Deleting a DropDown List.....	145
4	Repointing a DropDown List.....	146
5	Using a List.....	148
6	DropDown Control Panel.....	152
7	DropDown 'Other' Settings.....	154
8	DDL Variables List.....	157
9	DDL *Alias* Lists.....	160
10	Community DropDown Lists.....	161
11	Debugging the List.....	163
12	WordPerfect, PDF and Image Assembly.....	163
13	Double Duty.....	165
Part X Excel Spreadsheets as Datasource		167
Part XI Conditional Text (Options/Optional)		171
1	'Optional' Text with Prompts'.....	175
2	'Options' Text with Prompts.....	176
3	Hover-over Text.....	179
4	'Radio' argument.....	179
5	'and' and 'or' arguments.....	180
6	. . .to call documents.....	180
7	'Cumulative' argument.....	182
8	'Connector' argument.....	185
9	Fill-in the blank.....	186
10	Fill-in the blank	188
11	Nesting.....	188
12	Processing Order, Delaying Processing.....	190

13	"Syntax".....	191
14	Tables, Rows & <<Options>>.....	193
15	Create <<*Options*>> Assistant.....	194
16	Negative Optional (/NEGOPT).....	196
17	Testing & Editing.....	197
18	Administrative Text.....	198
Part XII	{Simple Optional} Text Blocks	201
1	Differences.....	205
2	Enabling/Disabling.....	206
3	Options (Mixing Simple with Complex).....	207
4	Negative Simple Optional Text.....	208
5	Converting.....	209
Part XIII	Groups and !GroupNames!	213
Part XIV	Anatomy of Variable and Optional Text Characters	217
Part XV	<<*Repeat* . . .>> Blocks	223
1	Simple Repeats.....	224
2	Incrementing variables.....	225
3	Repeat with !Groups!.....	226
4	Anatomy of a Repeat Block.....	228
5	Arguments ('and', 'or'; default #).....	229
6	Repeating Tables & Rows.....	230
7	Repeating Within Tables.....	232
8	'Return' the Repeat value.....	233
9	Repeat Settings.....	235
10	'AskRepeat'.....	236
11	Using 'Repeat' value for Options.....	237
12	Repeat(merge).....	239
13	Repeats Elsewhere.....	240
14	Repeat Alternatives.....	241
15	Repeat Restrictions.....	242
16	Testing Repeat Blocks.....	242
17	Repeat Examples.....	243
18	Repeat (a 'one page' lesson).....	245
Part XVI	Interviews / *Ask Tables*	251
1	The <<*Ask. . .*>> Commands.....	252

2	<<*Ask*>> Elements.....	254
3	(Arguments).....	256
	Requiring an Answer	256
4	<<*AskOptions*>> additional results.....	257
5	<<*AskValueInRange*>> Command.....	257
6	Hover-over Text.....	260
7	<<*If* . . .>> Command.....	260
	Cascading Logic	263
	Math within <<*If* ...>>	264
	Multiple Comparitors	264
	Multiple 'True' or 'False' Results	266
	<<*If*(arguments)* . . .>>	266
	<<*If* . . .>> formatting	267
8	<<*Set*>> Command.....	268
	Set: Math	270
	Set: Equal	270
9	<<*Get*>> Command.....	271
10	<<*Break*>> Command.....	274
11	<<*Process*>> Command.....	274
12	<<*Remarks* . . .>>.....	276
13	Logic Lesson.....	276
14	Setting Processing Order.....	276
15	Creation of Interviews.....	277
	Automatic Creation of <<*Ask*>> prompts	277
	Logic Assistant	278
16	Saving Interview Answers.....	281
17	Recalling Interview Answers.....	282
18	External source of Interview Answers.....	283
19	Print Interview.....	284
20	'Case' Logic.....	284
21	Hard Values.....	288
22	Requirements.....	288
23	Interview Examples.....	289
	Simple Interview Examples	289
	A Whimsical Example	290
	Explicit and Implicit !GroupName! settings	292
	A 'Legal' Example	295
	Anatomy of an AskTable	297
24	Displaying a Position.....	299

Part XVII Document Calls

301

1	Document Packages.....	303
2	Order of Search.....	304
3	Disabling.....	306

4	Interview Documents.....	306
5	Other DAB functions.....	307
Part XVIII	Document 'Dis'-assembly	311
1	Manually.....	312
2	Using DropDown List.....	312
3	Using Libraries & Books Screen.....	313
4	Last Paragraph Issues.....	314
5	Automatic Paragraph Numbering.....	314
Part XIX	Debugging Variables & Options Text	317
Part XX	QuickLinks	319
Part XXI	Support	323
1	Customer Service.....	324
2	Non-technical Support.....	326
	Index	327

Pythagoras was a 1st Century B.C. Greek philosopher best known for the
mathematical formula named after him.

More importantly to this program, Pythagoras believed that the essence of all things
is number, and that all things can be expressed numerically. Pathagoras, the
program, distills the theory as it might be applied to word processing by assigning
complex DOS paths and file names to simple numbers. The original versions of this
program (the "PathSmart" module in the current version) is a perfect adaptation of
the "Path = Number" concept.

While surely Pythagoras had more global things in mind,
this program nevertheless is proud to have this remarkable thinker as its namesake.

By the way, it is pronounced p'-THA-go-rus.

The Pathagoras System

Introduction

Part



1 Introduction



This is a substantially reduced version of the Main Manual. It contains bare bones, step-by-step guidance for Pathagoras' major systems. It is good tool, but not a replacement for the 'real thing.'

This Mini Manual is updated through release 2021 (updated through November 1, 2020).

Some functions describe herein operate only with release version 2020.5 or higher.

You can check the version installed on your computer via the Pathagoras 'Main Menu' | 'About' tab.

However, almost all primary functions described in this Manual perform similarly in all versions.

We would greatly appreciate your input as to the usefulness and 'usability' of this Help System.

If you have comments, corrections or suggestions for improvement, please send us an email by clicking [here](#).

If you see an error on a specific page, or wish to offer a comment about a page that you are viewing, click the 'envelope' in the upper right-hand corner of that page. That will activate your email program, and address a letter to us with a specific reference to the page you are viewing. Just add your comments and send. Thanks!

Pathagoras™ is a multi-faceted document authoring tool. It is designed to help you create documents of any nature, and to recall them instantly to the editing screen to be personalized for a client or customer. Pathagoras is well suited to any project, whether it is a complex transaction or a simple letter.

This mini-Manual is based on the 'main' Pathagoras Manual. While the main Manual was (and still is) intended as a reference source (it is close to 900 pages in length) and not meant to be read 'cover-to-cover', this mini-Manual is much shorter, contains the essential and most popular elements of Pathagoras, and very much can be read in its entirety if desired. It contains frequent references back to the main Manual for omitted features and for more detail about other features and functions.

Here are a few key features and notes about the program:

- Pathagoras is a Microsoft(tm) Word add-in. As such, it operates wholly within Word.
- The coding that you add to a form to automate it is typed onto and always resides in, and on the face of, each document you prepare. This is as opposed to how other programs handle their coding. With other programs, the coding takes place in ancillary screens, or remote tables. If initially in the document, when compiled, it becomes hidden (as in gray fields), or is moved to an auxiliary document. In most cases, a separate program screen is required to further edit or manipulate.

- Pathagoras self-generates the data input forms it needs to feed its database. You do not have to spend any time setting up databases, assigning inviolate variables or composing data intake sheets. But if you do have other databases that you wish to feed Pathagoras, we provide tools and outline methods to bring that data into Pathagoras.
- There are only a few easy to implement rules that you must observe to create a very powerful system.
- Pathagoras does not require you to abandon your current files or file storage system. Pathagoras adapts to the methods that you currently use to store and retrieve documents. Your current folder(s) of forms can remain intact. Each can be used as a Pathagoras 'DropDown List' or 'book' without any special configuration, programming or changes.
- Pathagoras offers no pre-packaged text. While sample text is provided below, these samples are not intended to be used as client facing documents. If, however, you already have forms on your computer in which variables are indicated between square brackets, those forms are already 'automated documents' as far as Pathagoras is concerned.

Pathagoras, PathSmart, SaveSmart and 'plain-text document assembly' are trade-marks or service marks of

Innovative Software Products of VA, LLC, and Roy Lasris.

Pathagoras, PathSmart and SaveSmart programs, interface screens, help system and all other visible and

non-visible coding Copyright © 2000 -18, and 2019

by Innovative Software Products of Virginia, LLC, and Roy Lasris.

All references in this Manual to "Word" refer to Microsoft® Word, a product of the Microsoft® Corporation. All references to Windows refer to the Microsoft® Windows® operating system, a product of the Microsoft® Corporation.

Other products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

1.1 Definitions

Definitions:

[Document Assembly](#) ⁴:

[library](#) ⁴

[book](#) ⁴

[clause](#) ⁵

[folder of clauses](#) ⁴

package of documents

[glossary](#) ⁵

[source](#) ⁶

[variable](#) ⁶

[Optional text](#) 

[Options text](#) 


[Repeating text](#) 

[Pathagorizing \('neutering'\) a document\)](#) 

[Document Disassembly](#) 

[Package](#) 

['Process' a document](#) 

['Personalize' a document](#) 

[Administrative Text](#) 

[Document Management:](#) 

[Profile](#) 

[SmartPath](#) 

[SuperSmartPath](#) 

Document Assembly:

'Document assembly' is the process by which an operator creates an entire document from a variety of component parts and then personalizes that document to meet the needs of the intended recipient.

Included within the scope of the term 'document assembly' are how source clauses are:

- created
- neutered, and
- assembled.

Pathagoras has adopted a 'library' and 'books' metaphor to describe way in which it organizes documents and forms.

Library: A 'library' is the top-most level. It is a collection of up to 10 'books,' along with rules and settings that control the assembly process. You can create and save an unlimited number of libraries.

Book: The concept of a Pathagoras 'book' is quite simple. A 'book' simply is a pointer to the folder on your computer (or network) where your source clauses are stored. When Pathagoras makes a call on a book, it is simply reading the file names in the folder to which it points. Recognizing the simplicity of 'what is a book' will lead to a much faster learning curve for Pathagoras. The terms 'book' and 'folder' can be used interchangeably.

- A **'folder'** is a standard, ordinary Windows folder containing Word documents. This is the most common type of book. See separate topic called Folder of Terms for more additional information on creating and adding terms to folders.

- A **‘glossary’** is a special, entirely optional, type of book. It is an advanced function of Pathagoras. You do not need to be familiar with this specialty 'book' to fully implement the program. A glossary is a single Word document that contains multiple clauses used to assemble documents. Instead of each clause residing inside separate documents within a folder, all terms reside in the same document. The individual terms are separated from each other by 'bookmarks.' See Glossary for more information.

It is very important that you understand at this point that 'books' are *pointers* to locations, and nothing more. Books are *not* the actual text found at those location. By the same token, a 'library' is not some sort of 'super' folder which contains the actual text of its enclosed books. It is just a text file that contains the (up to 10) references to the books assigned to it.

Therefore, when we say that a library 'contains' books, we mean that a library contains the *pointers* to the various folders or glossaries reflected in the library. A 'library' is a simple 10-line text file that comprises your collection of 'book' pointers.

When you add a book, you simply are adding a pointer to that text file. When you delete a book, you are deleting the pointer from the library text file, not the target folder to which it points. Similarly, when you delete a Library, you merely are deleting the collection of pointers, not any substantive text.

This should explain how a library can 'contain' books, yet those books can reflect so many disparate physical locations.

Attempts to make this definition more 'complex' than "a book is a pointer to a folder" will lead to frustration. Pathagoras really is designed on a simple straightforward model. Please don't try to over-think it.

Clause: A 'clause' is the smallest component of text that you have designated for assembly. It can be anything – a single word, a phrase, a group of sentences, paragraphs, pages, pictures, charts, etc. A clause can even be a complete document.

- Clauses typically represent the component parts of a larger document. It includes 'boilerplate language,' but also includes text and images that will reflect the personality and personal data of the intended recipient.
- A good 'book' will contain many alternative versions of the same topic. You should make a concerted effort to have a wide selection of clauses from which to build documents. The more variations, the more 'powerful' the system, and the better the final document can be. With an appropriate collection of clauses, strategically organized in appropriate books and libraries, you can create an infinite variety of documents to meet all client and customer needs. When you encounter a new variation of an existing clause, Pathagoras makes it easy to add it to your book.

Package: A group of documents typically produced for a customer as part of a single transaction. A real estate transaction 'package' would comprise a deed, deed of trust or mortgage, warranties and guarantees, and other closing documents. An estate planning 'package' would comprise a will, a trust, funding documents, a power of attorney, etc. See this section of the Manual.

Process: To process a document is to reduce it down (if using a template) or build it up (if clause and paragraph assembly) to an initial rough draft for the situation at hand. The result of a 'processed document is a perfect 'rough draft.' Such draft contains the appropriate text, with all options and optional text block questions answered and repeat blocks handled. All that is needed after 'processing' is to personalize the document (defined below). Options and optional text questions are asked, repeat blocks are 'repeated' the appropriate number of times. Processing occurs before 'personalizing' for a very logical reason. A preprocessed document rarely has the variables you want in place. When using a template as your starting point, it may have far more variables than the final document will contain. When assembling clauses and paragraphs from a blank slate, there may be none at the starting stages. Note: Processing automatically occurs in Pathagoras when you call text to the editing screen via Pathagoras' two primary 'document assembly' tools: the (1) Clause Selection Screen and (2) DropDown Lists. The result of a 'processed' document is a perfect 'rough draft.' (You can 'force' processing a document by pressing <Alt-P>.)

Personalize: To 'personalize' a document is to replace variables in the document with personal text. The Instant Database module (Alt-D) is the tool used for this process. It searches for and displays the variables in the left column. The right column is used to provide the personal information.

Variable: A 'variable' is a place holder for personal data. You should strategically place variables within your source clauses where you want that data to appear. Consequently, those variables will also appear in the first draft of any newly assembled document.

- Pathagoras prides itself on its use of plain text variables. Plain text variables are easy to create and insert into the source text. They are easy for operators and data entry clerks to understand when the final product needs to be personalized. See [Variables](#)³⁶.

Source clause: A 'source' clause is the 'original' version of the clause. It is the actual text stored within the books discussed above.

➡ Pathagoras stores source clauses in standard ".doc" or ".docx" files for easiest editing. See '.doc' vs. '.dot'

'Pathagorizing' (more accurately: 'neutering') is the process of preparing a document for use within the Pathagoras document assembly system. The process includes removing / redacting personal information from the original text and designating variables and optional text blocks which will be processed automatically by Pathagoras at document assembly time.

- The goal of 'Pathagorizing' is to make all clauses 'neutral' so that, after you bring it to the screen during a document assembly session, you can quickly and accurately personalize it for the specific client.
- The more neutral the document, the lower the chances will be that you will have a noun, pronoun or verb that is inappropriate in the context, or that you will accidentally leave the name of an earlier client's spouse as the beneficiary of the current client's Will!
- See separate section called Pathagorizing text for more information.

Optional Text: Keep it or delete it text. [Fully discussed beginning at this link](#)¹⁷².

Options Text: Multiple choice text. Select the text you want to keep; the remaining choices will be deleted. [Fully discussed at this link](#)^[172].

Repeating text: Blocks of text that will be repeated 'X' number of times. Useful for listing actors (e.g., 'children,' 'shareholders,' etc.) when the number of these actors is not initially known. [Fully discussed at this link](#)^[223].

Document Disassembly: The process of deconstructing a complete document into its component parts. The result is two (or twenty or two hundred) building blocks of text. The blocks in turn can be augmented by appropriate additional clauses offering variations and alternatives, and further augmented by bracketed variables and by strategically placed option, options and repeat text blocks. The goal is to be able to select from a large variety of available terms so that finely tailored initial draft can be created to quickly address the client's or customer's need. [See this link for full discussion.](#)^[312]

Administrative text: Those portions of Options, Optional and Repeat text blocks that contain the boundary markers, the command term, the (optional) !groupname! of the block, the (optional) multiple choices labels that you may wish to present to the end users, and other processing instructions that are unique to the block being created. The end of administrative text is denoted by a properly placed closing asterisk.

Document Management:

Document management deals with quick and efficient storage and retrieval of documents.

Pathagoras' document *management* module is reflected in the PathSmart and SaveSmart modules. They shed the 'library/books' metaphor used in the document *assembly* module, and adopt the terms 'Profiles' and 'SmartPaths' to describe the groupings.

A '**SmartPath**' is a pointer to a specific folder on your computer or network. Each *SmartPath* is assigned a number and a nickname. The program can access the *SmartPath* by 'clicking' on the desired *SmartPath* or, mouselessly, by typing the *SmartPath* number to the screen and pressing <Alt-G>. (The module is called PathSmart, but each pointer is called a *SmartPath*.) See separate topic called SmartPaths.

A '**profile**' is a collection of up to 10 'SmartPaths.' A profile groups the folders ('paths') that a user is likely to go for non-document assembly purposes. (It can be for document assembly purposes as well, but 'libraries' were specifically create for that purpose. Since it reflects the usage pattern of a specific user, a profile is typically given the name of the current user or the name of an office section. See Also: Settings

A '**SuperSmartPath**' is a regular SmartPath with one additional, but very powerful, feature. A *SuperSmartPath* allows you not only instant access to the folder to which it points, but to all sub-folders beneath it. See Also: *SuperSmartPaths*.

1.2 Pathagoras in 1...2...3 steps

Because of Pathagoras plain text underpinnings, creating and using automated documents really is as simple as One . . . Two . . . Three.

1. Mark up a document by creating variables with simple square brackets. [Client Name] is a variable. So is [Customer Address]. Multiple choice variables are denoted simply by putting slashes between the choices. [chocolate/vanilla/Rocky Road], [he/she/it], [Joanne Doe/Robert Richards/Alex Attorney] are some examples.

Create optional (conditional text) using curly braces. {This is optional text. When encountered by Pathagoras at assembly time, the text will be automatically highlighted you will be asked whether you want to keep or discard the text. }

Save your Pathagorized document it into your library of forms.

2. When you want to prepare your saved document for your client, call up that document to the editing screen using one of the several assembly tools that Pathagoras provides. DropDown Lists are the easiest, but if you want to build a document from building blocks, the Clause Selection screen is the better approach.

3. After the document is brought to the screen, Pathagoras automatically 'processes' the {Optional text} blocks first. That way, if you have variables such as [Child 1], [Child 2] etc. in a document being prepared for one who has no children, those variables won't appear in variables list -- they will have been deleted when you answer the "Do you want to keep this text?" question.

So, with the processed 'rough draft' on your editing screen, press <Alt-D>. This displays the Instant Database screen. Click the Scan button and the plain text variables created in step **1** above will be displayed in the left column. You simply provide personal values at the right. Click the Next button to replace variables with your clients' personal information throughout the document.

That's it! Pathagoras quickly replaces the variables throughout the document, ensuring consistency and saving hours of work! The values used for this document can be used in all other documents you create for the client.

Other features let you move from existing forms to automated forms quickly, easily and intuitively.

- If an existing form you wish to Pathagorize contains many of the same names (e.g., 'John Doe' appears in the title or caption of the document, in the body of the document and in the signature block):
 - highlight the first instance of one of those names
 - press Alt-V to activate the Create Variables wizard. Click 'Create Variables.'
 - provide a variable name (e.g., 'Client Name') to replace the existing name.
 - press 'Next'. All instances of that name are replaced with the new variable. (Even the upper and lower-case attributes of the original text are preserved.)
- If you want a choice made in one section to carry forward to another:
 - augment the variable or optional text block with a 'group name'.
 - just inside the opening boundary mark, type an ! (exclamation mark), a group name (from 1 to 30 characters), and another !. E.g.:

On this day came [Client Name], [!sex!Testator/Testatrix] and declared that this is [!sex!his/her] Last Will and that [!sex!he/she] signed it as [!sex!his/her] free and voluntary act.

➤ All members of the same group will change in tandem. A group name is noted by a word or short phrase surrounded by exclamation marks.

- You can insert building blocks of text in two clicks from a DropDown List
- You can insert building blocks of text mouselessly by just typing the name of the document and pressing Alt-G).
- You can easily disassemble entire documents into those building blocks. Just highlight the text you want broken out and press Alt-G. Select the target folder and provide a name. That's it.

Hundreds of other features permeate this feature rich program. They are discussed at length in this Manual. (You can view a bullet pointed listing of the major program features -- over 200 of them -- at [this link](#).

At Pathagoras, we back everything up with extensive support seven days a week. And when you contact us, instead of getting an automated service, you're connected with a developer—who's there to answer any question you may have.

Pathagoras is available by outright purchase (you own the license forever; no annual renewals required) or by subscription.

1.3 Design & Work Flow Concepts

"Pathagoras is different from its competitors. Any similarity is purely coincidental."

Document assembly is, according to one definition, the process of creating documents from precedent clauses. When first introduced, it was thought by many to be the next revolution in legal technology. In actuality, it has been a bit of a bust.

Why would this be the case? After all, the envisioned cost and time savings that document assembly can bring to an office was unfathomable. But when reality sets in, it was found that most of those who attempt to implement a document assembly system have found the process of preparing the text, the interviews and the coding needed to make it all work to be overly complex and time consuming. Many a well-intentioned soul began the process. Few finished it, and many of those who succeeded did so only with the (paid) help of consultants and experts specially trained in the particular program.

Enter Pathagoras. Our goal in developing Pathagoras was to create a dynamic, full-featured document assembly program that was simple, intuitive and actually enjoyable to use. We have designed Pathagoras so that every person in every office would want to use it, from 'Pathagorizing' a document (preparing it for use by the end user), to setting up the libraries and books of clauses (telling Pathagoras where the Will clauses, the Real Estate clauses, etc., are stored so they can be instantly displayed for selection) to personalizing the final document (including saving the variable to personal value match-ups so that data need only be entered once).

When we began to design Pathagoras, we began with no assumptions. This included an affirmative rejection of the concept that the work flow of existing programs was necessarily correct. We observed work flows in 'real' law offices. We heard the complaints and suggestions from active users of other document assembly programs. And we set to work and wrote the program around what we saw and what we heard. Here is a short list of some of the design requirements:

- One comment that we heard often during Pathagoras' initial design was an insistence that the program run within and be a part of the underlying word processing system. Learning a new program just to make an existing one work faster was not acceptable. To achieve that, we started writing the program using VBA, Microsoft® Word's macro language.
- There should be no dependency on hidden or 'out of document' fields. Creating fields (those gray blocks of text that changed values depending upon the embedded coding) was often confusing. Even if the document assembly program would create the fields for the end user, they were difficult to modify. It was not always obvious why a document assembled the way that it did because the coding for the document was stored 'somewhere else.'
- Creating menus was the most difficult part. Drawing meaningful screens needed to 'interview' the end user was tough. Writing the logic so that appropriate clauses would be selected based on the interview answers was even tougher.
- Were the tedious steps needed to create these interviews really necessary? "Why can't all potential clauses simply be displayed in front of us and we can pick and choose the one's we want?" The fancy stuff can come later. For now, "all I want is a quick way to get to my clauses."
- Many users wanted to be in total charge of what clauses would be inserted. They did not want a computer program telling them what clauses to use based on an interview. On the other hand, some offices use paralegals and other staff to create the base documents and the attorneys don't want the initial decisions made by a non-attorney. Use to power of computers to remember a specific pattern based on specific criteria.
- The program 'basics' should operate with little set up beyond installation. Significant parts of the program must be usable right 'out of the box.'
- The reason what a document works (or doesn't work) the way it does should be 'facial' to the source document. It should be a simple matter of looking at the source text, not ancillary files, to see why an assembled document looks the way that it does, and the source text should be easy to access so that any corrections can be quickly make and quickly tested.

We concluded that 'document assembly' is initially about getting a reasonable rough draft of a desired document(s) to the screen. What was 'reasonable' varied among users. Some have very low demands. Any text bearing any resemblance to the intended end product would do. Others wanted an almost 'perfect' first draft. (Those wanting the former far outweighed those wanting the latter. This clearly indicated that the complexities involved in getting a 'close-to-perfect' initial draft is what dissuaded even techno-savvy users from using the program more -- or at all.)

So here is work flow concept #1. *The quicker the end-user can get an acceptable version of the final desired text onto an editing screen, the happier that end-user will be. And the happier the end user, the more likely that document assembly would actually be implemented. This is where we set our focus, and this is where Pathagoras shines.*

We made the creation of document variables as simple as possible. Square brackets around placeholder words was to be the primary shape of a variable. And all text was to be 'plain text' (i.e., only regular keyboard characters would be used).

We practically eliminated the need to navigate up, down, across, further up, further down, your folder/network trees to find documents. But we did so while at the same time not requiring you to change a single thing about the way you currently organize your documents. And this is where Pathagoras begins to separate itself from the pack.

With Pathagoras, you avoid navigation by setting pointers to your existing folders, and assigning meaningful names to those pointers (typically the current folder name). Once set, you never manually navigate to that folder again. You use 'on-screen' cues and simple clicks of your mouse. The productivity savings with no-more-navigation is immeasurable.

Personalizing the assembled document to reflect the proper client or customer information was obviously important as well. And that is discussed next. But our users made it clear that just getting something on-screen to manipulate was the most important first step to document assembly nirvana.

Here are Pathagoras' two primary document assembly tools:

- **DropDown Lists:** This was mentioned in the introduction. Point up to 10 folders (or parent folders if you want to use the 'tree service') to Drop Down Lists and never navigate to them again. So, if you have highly 'Pathagorized' documents or document which you have never prepared for automation, you can quickly bring an exact copy of any item in that folder to your screen by just clicking on the item. Nothing more.
- **Libraries and Books:** More clicks for a single document (so DDLs are preferred in that setting), but when you want to assembly multiple clauses or multiple documents, this is the way to go. An unlimited number of pointers to you most needed folders, arranged by topic (Estate Planning or Family Law) and specific subject within the library (Wills, Trusts, etc. or Divorce, Separation Agreements, etc.)

To call a document you wish to prepare for a client or customer, you would either call it from the DropDown List (2 clicks) or select the library then book then specific document(s) from the Clause Selection Screen. A rough draft of the document(s) you selected is instantly brought to the editing screen.

Once getting text to the screen became 'second nature', adding 'automation elements' to the source documents comes to the fore.

Work Flow Concept #2: Adding 'automation' elements to source documents must be easy and intuitive. With Pathagoras, automation is accomplished using plain text and remains intact on the face of the source document. Original documents need not be converted, renamed or moved.

Everything in and about Pathagoras is 'plain text' and 'facial'. Let's take these concepts one at a time:

Plain text:

- Variables are holding places for words you want to add when you are ready to personalize your document. Variables are created simply by surrounding words with square brackets. [Client Name] is a variable.

- Optional text blocks are 'keep it or leave it' blocks of text that may or may not remain in the final document. The decision is made at document assembly time. They can be created simply by surrounding words, sentences or entire paragraphs with curly braces.
- Options text blocks is commonly known as 'multiple choice text. They are text blocks from which you want to select at least one choice from several that are provided. Options text blocks likewise are words, sentences or entire paragraphs surrounded by curly braces, with the choices be separated by "/OR".

(We describe the simplest iteration of Options and Optional text blocks in the above sections. More robust alternatives are available for the advanced user, allowing complex selections and promptings for the end user.)

Just a note here. Some competitive programs also claim that their editing for automation is done in 'plain text.' There is a half-truth to their assertions. Initial automation markups can be done in plain text. The problem is that when you have finished editing and press the buttons to make it part of their systems, they convert your work into a new document, replacing your typing with their hidden fields and links. They sometimes move the document to a new location and almost always rename it, assigning a foreign extension and creating a tag-along document which contains additional instructions for processing the file.

None of that happens in Pathagoras. When you have finished editing your document, just save it. When you need to edit it again, you recall it from its same location. It starts as a Word document. All intermediate steps are done in Word and the final product is a Word document. We call this "All Word, all the time."

Facial text: The culmination of the above 'rant' is Pathagoras' concept of faciality. What you type is what you keep. No conversions. No fields. Document automation triggers are typed on the document's face and remain there until the document is automated. If your document does not process correctly, you can 'read' on the face of the document what went wrong. You do not have to consult the ancillary files (where ever they may be), edit the fields, convert and then reconvert. So 'Word-like' (here we are referring to Microsoft Word) is the process that when something goes wrong, you can simply press the Undo button on the assembled document. This lets you step backwards through the process to see what Pathagoras did with the variables and options text you provided. This makes it very easy to see (albeit in reverse) what Pathagoras is doing with your markups so you can correct the original if the need be.

Once you have edited and saved your document, refer to Work Flow Concept #1 to access you document.

Here is BIG WARNING #1:

Never use an original document to serve as your base for a document assembly session.

So long as you abide by the above rule, you will not accidentally overwrite your source documents. (We are sure that it has never happened to you, but some people forget to save the edited documents under a new name. The original document is therefore lost. When you heed BIG WARNING #1, and follow the document assembly steps stated above, you minimize the chances of that occurring.

When a document is brought onto the screen using Pathagoras tools you will always be working on an exact **copy** of the original. How do you know it is a copy? Look for the 'name' of the document in the upper left-hand corner of the screen. It will be named "Document1", "Document5", etc.

If you find yourself working on the original for other than source editing purposes, you should rethink your process, and implement the tools provided by the program. If you do nothing else in your early days with the program you should at least "Create your First Library" and "Shelve your First Book" and call up copies of documents from your various books. If you find yourself working primarily on complete documents or templates, consider our exceptionally popular [DropDown Lists](#)¹⁴⁰.

Adding New source text (Pathagoras Work Flow Concept #3).

Once you begin using Pathagoras, you will begin to muse, 'Gee, I sure wish I could easily and quickly add additional documents to the folders I assigned to my books or to my DropDown Lists.' (Well, maybe you won't state it quite that way, but the thought will nevertheless cross your mind.)

The process is (or course) simple, and no navigation is ever required.

Depending upon the source of the text you want to add, you have several choices.

Copying existing documents from 'other' folders:

- **To DropDown List folder:** If you are moving multiple files, copy files from the source folder by highlighting them, right clicking and "Copy". Then, click Display Folder from the appropriate DropDown List. When the folder displays, right click and select "Paste." (Ctrl-C and Ctrl-V work as well.) If the text you want to move is 'on-screen," simply highlight it (if you only want a portion). Drop down the appropriate DropDown List and select the "Save to Folder" element.
- **To Libraries and Books folder:** If you are moving multiple files, highlight their names in the folder. (By folder, we mean simply the folder into which you navigated using regular Word/Windows navigation tools. You are simply locating files you want to add to a folder which you have identified to Pathagoras as a book.) Right click on your selection and click "Copy". Close that menu. The selected documents are now in your 'clipboard.' Next, using the Document Assembly button, select the library and then the book into which you wish to add the new documents. Click "Display Folder" option from the expanded screen and click Next. A standard Word 'file open' dialog displays. After the folder displays, right click in

any 'blank' area of the screen and click "Paste." (Ctrl-C and Ctrl-V work as well to copy and paste the files.)

Moving 'on-screen' text into source folder:

- **To DropDown List folder:** Highlight the portion of text you want to add to the folder associated with your DropDown List. Drop down the appropriate List and select the "Save to Folder" element. Pathagoras will copy the highlighted text into a new document and save it with a name that you provide into the target folder.
- **To Libraries and Books folder:** Highlight the portion of text you want to add to the folder associated with you book. Select the appropriate Library and then Book and select the "Save to Folder" from the expanded screen. Click Next. Pathagoras will copy the highlighted text into a new document and save it with a name that you provide into the target folder.

(Of course, you can always move files into a target folder the 'old fashioned' way, and if you have several documents to move around, you can use standard file manipulation tools. But if you have 'free text' or a single document, the steps listed above cannot be beat for speed.)

Notice that all the above is irrespective of the nature of the documents being moved.

Workflow Concept #3: There is no such thing as a 'Pathagoras document.' Pathagoras owns no document. There are only documents, and they carry the definition of documents according to Word and Windows, not Pathagoras. (Don't get us wrong. You can 'Pathagorize' a document by adding brackets around variables, and by adding Options and Optional text blocks, but the result is still Word document with the same name (unless you decide to change it) and extension (.doc or docx) as the one you began with.

Workflow Concept #4: Pathagoras goal is to make your document creation tasks easier, not different.

There is a lot more to the program, of course, than discussed above, but this should get you started with a solid understanding of Pathagoras' approach to document assembly.

Workflow Concept #5: It should be easy to add new text or to access for edit existing text to any document assembly system.

Marketing Concepts:

While we were at it, we decided to include our philosophy regarding marketing.

Marketing Concept #1: Demonstrate the program on the potential customers computer. No tricks up our sleeves. We show you how easy it is, and how to do it, live using your documents.

Marketing Concept #2: Show prices on the website.

Marketing Concept #3: Don't ask for life history to allow the viewing of our websites and videos.

Marketing Concept #4: Don't nag. Maybe every once in a while send a newsletter or mailing to ask how things are going, but don't nag.

Marketing Concept #5: Be as available before the sale as after it.

Marketing Concept #6: Be as available after the sale as before it.

Marketing Concept #7: Leave the user better off for having tried the program (or at least no worse off.) Example: Many customers 'coded' their variables like this: ***** and/or _____. During a demo, we may replace those 'variables' with much more meaningful variables like "[Date of Letter]", "[Client Name]" and "[Child@1]". The customer may decide not to buy the program, but doesn't have to undo the changes made. Indeed, he or she has a better document in exchange for the trial. So your efforts at the trial are not wasted.

Marketing Concept #8: Offer features that are not dependent upon document coding. Allow the user to use the program when purchased, not just after the program has been fully configured. We can brag about two of these features: "[DropDown Lists](#)"¹⁴⁰ and the "Names and Subjects Editor". These are spin off programs, potentially stand alone. By the same token, they are also at the heart and soul of Pathagoras.

If it is challenging to locate the source text to edit it, or to add more source text, a typical user doesn't even try in many case. So we had to make it easy and consistent with the way the typical user already functions.

When you want to create or edit source text, that too can be done without navigation. ('Source text' means the documents and other files that you want to store in the folders targeted by your books and DropDown List. It is from 'source text' that a copy is called when you take the above document assembly steps.)

Pathagoras' allows you quick access to your source text:

- **From DropDown Lists:** Drop down the list. Locate and click on the element called "Display Folder". A standard Word 'file open' dialog displays. Simply select the original file you wish to edit.
- **From Libraries and Books:** Click the document assembly button. Select the library and then book. Click the "Display Folder" option from the expanded screen and click Next. A standard Word 'file open' dialog displays. Simply select the original file you wish to edit.

- **Using existing Word tools:** Don't forget that you are in Word. Word maintains a Recently Used Files list. Use that list when you need to edit a file you just worked on. If you are editing, testing, reediting, retesting a document, this likely is the fastest way to access your work. And it is a perfectly acceptable (indeed, preferred) technique.

Pathagoras saves you time even if you don't 'Pathagorize' a single document. Even if you have inserted no brackets, no optional text, etc., into a single document, the fact that you can quickly call up a document onto the editing screen is a meaningful time saver. Some customers have bought the program for the DropDown List feature alone.

When we demo, we demo when possible on your computer.

1.4 Processing vs. Personalizing

Pathagoras is programmed to handle two 'types' of text. The first type we call 'process text', typically large blocks of text in a template marked out with '<<' and '>>' boundaries that either stays or goes, depending upon the answer to a question. The second type is 'variable' text, typically short answers, like a name, location, pronoun, etc., represented in the document with generic words surrounded by square brackets. Variables are typically saved as a permanent record. Variables can be recalled for use in other documents that contain those variables. 'Process' text choices are not preserved (except of course in the document that you created) for reuse on a repetitive basis).

Summary:

Document assembly typically involves 2 steps.

- **Processing** -- Selecting or discarding the text blocks that will make up the initial draft of the document.
- **Personalizing** -- Assigning specific values to the variables. If 'processing' occurs first, only the relevant variables will be present in the 'processed' document.

To become most efficient in using Pathagoras, it is imperative that you understand the difference between 'processing' and 'personalizing' a document and the order in which these two modules are optimally called.

Pathagoras functions are directed at two 'types' of text. The first type we call 'process text', typically large blocks of text in a template that either stays or goes, depending upon the answer to a question; and variable text, typically short answers, like a name, location, pronoun, etc.. The latter (variables) can be saved as a permanent record and recalled for use in other documents that contain those variables. In the former the choices are not preserved (except of course in the document that you created, but not for reuse on a repetitive basis).

=====

Processing

'Document processing' includes those steps you take to arrive at the first perfect rough draft of your document. Processing a document involves

1. assembling the document building blocks from source text (discussed below) and
2. answering Options/Optional/Repeats questions that appear in the source text.

NOTE: Options/Optional and Repeats questions will be automatically presented when you assemble document using the tools provided by Pathagoras. This happens because Pathagoras is programmed not only to display the requested text but to hunt for text inside of '<<' and '>>' boundaries and act upon the text found therein. This is the 'secret sauce' to Pathagoras operations.

Many customers who are not yet familiar with the 'flow' of document assembly with Pathagoras may begin a document assembly project by recalling the original source text onto the editing screen. That, of course, is the way they undertook a project in 'pre-Pathagoras' times. We need to 'break' you of that habit.

Of course, when you simply recall source text to an editing screen, it will just 'sit there' until you do something to cause a change. Pathagoras thinks you want to edit the source text to correct a typo, add new text or add additional automation features. But it never assumes you want to *process* that text. Pathagoras knows that an accidental save of a processed original will wipe out that original document. So, Pathagoras requires that you 'assemble' your document to get the benefit of the program's automated features.

Note: if you are composing source text and want to process it for testing purposes, that is perfectly fine. Read [this section about testing](#)^[197].)

There are two primary methods to **assemble** documents using Pathagoras:

- (1) Selecting documents or clauses from the **Clause Selection Screen**.

This is the 'classic' assembly process involving a click of the Document Assembly button, the selection of the proper library and then book from the library. When you click the Next button, the available clauses appear in the left panel. Select the desired documents/clauses and move them to the right ("Selected") panel. Another click of the Next button and the selected items are instantly assembled.

- (2) **DropDown Lists**

If you have assigned a folder to one of your up-to-10 DropDown Lists, just click the list, point to the desired clause or document and click. The selected item is instantly brought into your editing screen.

When a document or clause is 'assembled' via any of the above techniques, Pathagoras is not done. Pathagoras searches for the existence of any <<*Options*>>, <<*Optional*>>, <<*Repeat*>> or <<document call>> blocks that may have been brought onto the editing screen. If found, they are processed.

The various commands can cause existing text to be deleted, as would be the case of 'No' to an Optional command, or a 'non'-selection of an Options choice, voiding the document of certain irrelevant variables. A command may call in new text, and that new text may contain variables that were not present in the originally called block.

If the text processing involves responding to Repeats that contains variables within the scope of the <<*Repeat*>> block, Pathagoras will add new variables, each incremented by the designated number of repeats: [Buyer@1], [Buyer@2], etc.

Processing is iterative. If the commands bring in more text, that new text is processed as well. It continues unabated until Pathagoras has determined that there are no <<*Options*>>, <<*Optional*>>, <<*Repeats*>> or <<text*>> blocks to process.

Personalizing

Only when the document is fully 'processed' (i.e., you have a 'perfect' rough draft of the document), does it then make sense to 'personalize' the document. It's at that stage that your document contains the proper variables (including variables incremented by the Repeat processing) for completion.

If you are satisfied that you have a great base document, press <Alt-D> to bring up the Instant Database screen. Click the Scan button (or a Mask or an existing record) and Pathagoras will hunt for and display in the left-hand column all bracketed variables contained in the document. Provide replacement values at the right. Click the Next button and all variables are quickly replaced with their respective 'personal' text. (You will also be asked if you want to save the variable-to-values pairings to an Instant Database record. You typically should say 'Yes.')

=====

What if I want to replace my variables first? Am I allowed?

Of course you are allowed. Pathagoras lets you process your document in any way you wish. But if you follow the advice in this article, you will likely save time and effort.

Can the two steps be combined in a single keystroke?

Yes, and many have done this. To set this up, press <Alt-D> to display the Instant Database. Press the red Power Tools button and then More Tools. Scan the resulting screen for various options (including the one called 'Process Document before completing variables.' So, what's the drawback? Not much, except that you may forget that <Alt-P> exists when you just want to process a document (or when you are teaching Pathagoras to others who may never know that the <Alt-P> function ever existed).

So when do I set up text for 'processing' and when for 'personalizing'?

Generally, the larger the block of text, and the less 'personal' it is, the more likely it is that you are going to set it up as process (i.e., <<*Options*/*Optional* . . .) text. Large swaths of text don't make very good variables, and you don't typically intend to save large blocks of text for reuse in other documents in the same way you want to save a name, address, etc. (i.e., variables) in a record for reuse across a wide variety of document.

But there are 'cross-over' terms that you can set up as either process text or as variables. The most 'classic' examples are those words and phrases that deal with sex and number. Should 'he/she/they' be set up as the variable [he/she/they] or as process text <<*Options*he/she/they*>>. The answer to the former is stored as a recallable record (i.e., as a variable) and can be recalled for any number of future documents for that client; the latter is not saved outside the document and cannot be used for future documents. The answer to the question, therefore, lies in that explanation. If you plan to reuse the information, present it as a variable. Otherwise, you can just set it up as 'process' text for the particular document. And keep in mind, whatever you decide, if you later change your mind, it's a simple switch.

1.5 Moving from basic to advanced

Pathagoras allows you to approach document assembly with a methodical, measured pace. It allows you to implement as much or as little of the program as you wish, as fast or as slowly as you wish. You can use the simplest of setups to get you started and then take advantage of the automation provided by Pathagoras to elevate the sophistication of your source text to any level you desire.

Unlike so many other programs out there, Pathagoras is not an 'all or nothing' type program. Start as simply as you like (or -- and we are not dissing anyone here -- as your skill level allows). Build on your successes on your own or using automation tools provided by Pathagoras.

Not a single step is wasted by starting 'simple.' You will not have to 'start over' if you decide to take a different approach. Pathagoras is replete with conversion tools so that you can experiment with different document assembly techniques. If you decide to try a different approach, use Pathagoras conversion tools to make the conversions for you.

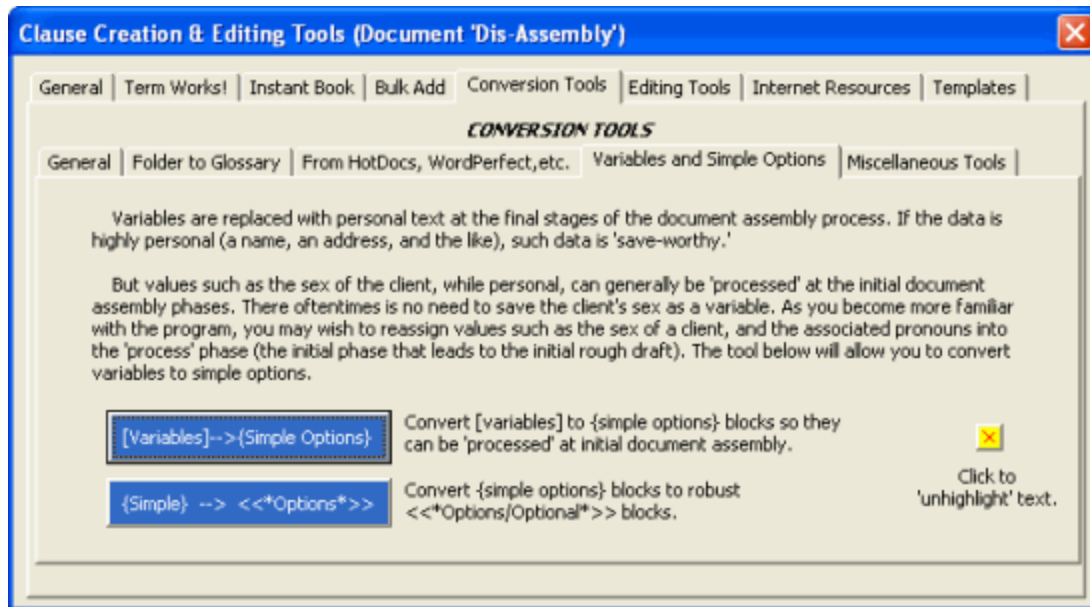
As you begin creating your source text, don't try to 'eat the whole elephant' at the beginning. It can lead to frustration. Our suggestion is that you start 'Pathagorizing' your first form by creating 'simple' variables and 'simple' optional text blocks in a single relatively simple document. Play with that document to see and understand the action of Pathagoras. A 'simple' variable is a word surrounded by square brackets: [Client Name] is a simple variable. A simple optional text block is just a word, sentence, paragraph or group of paragraphs surrounded by curly braces. {This is optional text.}

Once you appreciate the easy stuff, then move on to augment and advance the document to incorporate some of the more advanced features of Pathagoras.

Pathagoras conversion tools allow you to move from 'basic' to 'advanced' in an automated fashion. These tools include:

- Convert variable to simple options. Most of the time, a variable should remain a variable. But sometimes the variables are more in the nature of the document's structure. Examples, the sex of the client; the number of children. These items can be handled as variables (and for the beginner, they should), but the gender of the client and (perhaps more obviously) the presence or absence of children are more 'structural' in nature. Huge chunks of the document's rough draft would be eliminated if there are no children.
- Convert simple options to robust Options/Optional blocks. With these more robust blocks in place, you can present to the user more meaningful prompts that lead the user to the right choice. So instead of just showing the first 100 characters of the text of each of the options in the 'select' screen, you can present meaningful questions for the user to answer. Plus, with these robust blocks in place, you are just one step away from 'AskOptions.' Read more below.
- Convert 'robust blocks' to an <<AskTable>>. When Options/Optional and Repeats blocks are in place, with each augmented with a !groupname!, Pathagoras' wizard can collect all such blocks and place them at the 'top' of the document. When the document is processed, all questions are asked at the start and at one time. This avoids the repetitive ask one question/give one answer that the other processes use. This becomes the 'ultimate' in document assembly ease and speed.

Conversion tools can be found under the Wizards and Assistants section of the Pathagoras Features list. When clicked you will be taken to this screen:



Note the various tabs under the title line. You can convert from Folder to Glossary, from Glossary to folder; if you have HotDocs® documents, you can convert the variables contained therein to simple variables recognized by Pathagoras. And more germane to the topic of this page, you can convert simple variables to simple options and simple options to robust options, all with just the click of a mouse.

Click on the appropriate button and follow the prompts that Pathagoras will provide.

1.6 File and Folder 'Pointers'

The 'plain text' operations that define Pathagoras are made possible because of its equally simple programming design. Pathagoras doesn't create a new subset of your documents, or create a database of your files requiring calls to retrieve or save a document to be intercepted by the program for initial processing. Pathagoras merely points to where your documents currently reside (or to the location where you save or move them using normal Word and Windows techniques).

An understanding of this incredibly simple 'pointer concept' is helpful if you wish to fully implement the program and integrate it into your practice.

➔ **A book is simply a pointer.** A Pathagoras book is a pointer to a folder. And a folder is just a standard Word/Windows folder, just like the kind you used before Pathagoras. It is a folder into which you can manually enter and from which you can manually retrieve clauses whether Pathagoras is in use or not. (It is nice to know this latter fact because if Pathagoras should ever 'crash', you are not stranded. Just navigate to the folder to retrieve the desired document.) When you assign a folder to a book shelf (let's say #8 represents the physical folder "c:\my documents\contract clauses"), Pathagoras lets you refer to the physical location by the number 8, but otherwise the folder is 100% intact.

By extension, a book is not a collection of the documents it points to (although to the end user, it definitely feels that way). Rather it is simply a pointer to those documents. With that definition in mind, let's explore the positive consequences of this arrangement:

- You technically cannot add text to a 'book' (since the book is simply a pointer). You add text to a folder, just like you have always done in the past. It becomes part of the book simply because you added it to the folder. When Pathagoras rereads the folder's contents at the next document assembly session, the new document will be automatically added to the display.
- When you delete a document from the folder, the book is diminished simply by that action. When Pathagoras rereads the folder's contents at the next document assembly session, the deleted document won't display due to the mere fact that it is not present in the folder. You (as operator) don't need to perform any other task to keep your books in sync with your folders.
- When you delete a book, you are deleting just the pointer reference, not the physical folder. So, you can add and remove books from shelves in your library with abandon.
- A book cannot exist independently from a library. (To be useful to you, these 'books/pointers' have to be stored somewhere, and the concept adopted by Pathagoras is that these books are collected within libraries.) Which leads us to the next main topic . . .

➔ **A library is a collection of pointers.** Following the same line as above, a library is not a large collection of files. The libraries refer to files, but are not the files themselves. (This differs from other programs where the files that make up the library are moved, encrypted or compressed into a single container. Rather, a library is no more than a collection of up to 10 books/pointers. Indeed, a library is simply 10 lines in a simple text file. Each line in that text file will be either a pointer to a folder in the style of "c:\my documents\contract clauses", etc.), or it will be a blank line, representing an as-yet-unassigned book.

When you delete a book, you are not deleting the folder. You are deleting only the pointer which references the folder. Likewise, when you delete a library, you are merely deleting a collection of pointers, not any text in the actual folder to which the book points.

By way of illustration (and possibly by way of overkill, but it is important that we drive home this point): Let's say that you wrote the name of a folder on a piece of paper. You then took an eraser and scrubbed that folder name off the piece of paper. You have merely erased a bit of text, not the content of the folder that the words represented. If you want to physically delete or move a folder, you can do so, but must do so using standard Word/Windows techniques.

To see where a specific book is pointing, just hover your mouse over the book in the Libraries & Books screen. A small 'tip' box will appear telling you where it points.

➔ **DropDown Lists**, similarly, point to collection of files (typically documents, but can be anything) in a designated folder. The top-most and bottom-most elements of each List contain the information as to the whereabouts of that folder. When you click on a file name from the list, Pathagoras checks for the folder name and then quickly finds and inserts the appropriate file into your document.

➔ **Instant Database records** are individual text files typically saved on the user's computer. By default, Pathagoras can store them/find them because the initial IDB pointer is set to "C:\program

files (x86)\Pathagoras\IDBs". But your Instant Database files can be stored anywhere and shared with anyone. To move them, just re-point the program (via the Instant Database settings screen) to the folder where you want your records stored. If you want others on the network to use those same files, just point them to the same location.

[Click here to learn how to set a new location](#)^[106] to which Pathagoras should point to find you Instant Database records.

Don't overthink this. Setting a location for your Instant Database records is merely 'pointing' Pathagoras to the location where those files are, or where you want them to be, stored. You are *not* activating a proprietary database where the records are encrypted, compressed and stored in a fashion that is inaccessible unless you have Pathagoras. To the contrary, your Instant Database files are stored in normal Word/Windows folder and readily accessible to any program.

See:

[Pointing/Repointing a Book](#)

[Pointing/Repointing to the Instant Database Records](#)^[106]

[Pointing/Repointing to your *Aliases*](#)^[124]

[Pointing/Repointing to DropDown Lists](#)^[146]

1.7 Advantages of Plain Text Document Assembly

There are several advantages to our plain text approach to document assembly. Here are a few:

- The learning curve is much shallower. While there is syntax with which you must become familiar, it is minimal and easy to learn. Pathagoras approach is undeniably simpler than the kind of field coding or setup required by others.
- Because it is 'all Word, all the time', you don't have to worry about [Client Name LIKE THIS] signals the other programs demand. Instead of the additional coding, just directly make variable look like how you want it to appear in the document. [CLIENT NAME] at the top (because you want the client's name spelled out in ALL CAPS) and [Client Name] in the body (because you want in Upper Lower case style) is just fine.
- The initial and the final 'coding' is facial. If your Pathagorized document does not process correctly, you can 'see' on the face of the document what went wrong. You do not have to consult the remote tables or ancillary files (where ever they may be) to edit the fields, convert and then reconvert the source text.
- Because it is 'all Word, all the time', if something goes wrong, you can simply press the Undo button on the assembled document. This lets you step backwards through what Pathagoras did in assembling the document. This makes it very easy to uncover (albeit in reverse) what happened to your markups so you can correct the original if needed.
- 'Plain text' means that you can 'Pathagorize' your source documents using a computer on which Pathagoras has not been installed! (Of course, to *test* your work, you will need a computer on which Pathagoras has been installed.)

- Any commercial document, document that you find on-line, or document that you pick up at a continuing professional education class, that presents variables within brackets is automatically a Pathagorized document.
- Perhaps most importantly, if you happen not to have the identical job in the office one year (or 10 years) from now, your successor has a fighting chance of figuring out what you have done.

All references to "Word" refer to Microsoft® Word, a product of the Microsoft® Corporation.

1.8 Document Assembly vs. Document Management

Pathagoras contains both Document Assembly and Document Management modules. As used by Pathagoras, these two modules are distinct, but integrally related. The access to the two functions is controlled by different screens, but the underlying programming code frequently overlap. It is important to understand, and distinguish between, these two concepts.

- The two modules are not mutually exclusive. They are not polar opposites. Rather, they are distinct components of the same system, and are designed complement and support each other quite nicely.
- Typically, a user will create and personalize a new document by using the Document Assembly module. After the initial editing is done, the user will store the new document, and recall it for later editing or printing, using PathSmart/SaveSmart (document management) tools.
- A folder mapped to a PathSmart 'profile' (collection of SmartPaths) can be the same folder as a book mapped in the Document Assembly 'library' (collection of Books). For example, the path to 'Office Forms' might be both a Book in a library and a SmartPath in profile. But they are used for different purposes.
- However, a folder containing personalized client documents (mailed letters, completed contracts) would *never* be part of a Document Assembly library. Why? Because personal documents would never be directly used to create future documents. (At least we hope that you never directly use a form created for one person as the source of one you intend to create for another. Doing so can lead to embarrassing, if not disastrous, consequences. 'Pathagorize' it and save it to a forms book.)
- A document retrieved from a SmartPath will typically be the original document. When you use the document management tools, Pathagoras presumes you want to edit the original text (modify content, correct errors, etc.).
- A document retrieved from a Book will typically be a copy of the original. When you use the document assembly module, Pathagoras presumes you are creating new content using original sources, and protect you from accidentally overwriting the original.
- Folders containing personal client letters, contracts awaiting execution, pleadings for a specific case and other personalized documents are precisely the kind that would be mapped within a PathSmart profile. Folders containing form documents, clauses, templates, glossaries, etc., that you view as 'source' text for creating new content are what would be mapped to a Book.

- There are other differences among the modules, some subtle, some more blatant. Only actual use of the program will bring into sharp focus the differences between them. Just take comfort in knowing that both systems work seamlessly together.

The following further illustrates the differences between document assembly and document management. We will stick with the libraries and books metaphor described in previous sections.

- Let's assume that you have been assigned to research and prepare a report on flying buttresses for an architecture project. To accomplish your task, you would go to the appropriate architecture library to find the information you need. You would look in the various books there. You would extract text and copy other source materials for the final project. And you would assemble your research into what ultimately becomes the first draft of your final report.
- After you have completed the report, you would not return to the library to store your composition. Rather, you would store your work in a location appropriate to a specific 'personal' project. You would also (mentally) use a methodology for storing client/customer projects that is distinctly different from the methodology used by the library from which you gathered the source material.

Likely, you would place your final research product inside the customer's folder or a client's file, organized by customer/client name. (If it is a general office project, you might place it in a general-purpose filing cabinet.)

But one place you *don't* store it is back in the library.

The same concepts pertain to the primary systems that make up Pathagoras. The document *assembly* system provides the tools needed to create documents from source clauses. But once created, the document *management* system provides the tools by which you can store and easily retrieve the completed documents from personalized storage containers.

'Nuff said?


1.9 Beyond 'Major' Projects

Pathagoras is a multi-dimensional program. It can easily be used in every aspect of office operations. Pathagoras can assemble complex trusts, lengthy bids and detailed requests for proposals. It can be used to put together highly personalized pleadings and contracts.

But Pathagoras should be used with everyday cover letters and memos, too. Its simplicity and speed lend itself to such use.

Think broadly and creatively when it comes to implementing Pathagoras. It can save you an incredible amount of production time. There is no reason why every form and letter in your office should not be Pathagorized.

- Call up your frequently used documents and forms and put [brackets] around every potential variable.
- Assign folders containing frequently used forms and letter to a [DropDown List](#)¹⁴⁰ and 'point and click' instead of navigating.

- Use the [Instant Database](#)  module to quickly identify and replace each variable.
- <Alt-G> can call in not only the ultra-sophisticated blocks of text into your work screen, but also simple signature blocks, letterheads and other text frequently used in everyday office operations.

Unlike other programs that require extensive setups that would dissuade their use on your 'simple' forms, Pathagoras is the opposite. Its plain-text roots make 'Pathagorizing' your entire office quick and easy. A little bit everyday is all that it takes.

1.10 Help Connections

Other Built-in or On-line Help Systems


- ➔ Check out our [FAQs](#) found on our site. Read what others are asking; pick up some good hints and tips; read what others have posted as 'challenges' and our (or others') solutions to those challenges. We think that the FAQs are as important a resource for the efficient use of Pathagoras as is this Manual.
- ➔ Every overlay screen shown by Pathagoras contains at least one box containing a question mark ("?") or globe. These symbols are fully integrated with the Pathagoras Help System. Whenever you click one of the '?'s or globes on a screen, you will be taken to the appropriate topic in this system.
- ➔ A downloadable, printable PDF version of this Help System is available. Click here for the details.
- ➔ Two shorter guides designed to help the newcomer become familiar with the scope and features of the program are available. Check out:
 - ['Beginner's Guide to Pathagoras'](#) This Guide contains illustrated, step-by-step instructions for the program's basic features.
 - [The 7-day Plan](#): Spend 15 to 20 minutes each day leaning the program basics. Specifically written for those who "don't have enough time to learn another program." This is by far the most popular of our beginner's guides.
- ➔ An 'in-line' help system can be activated by clicking the "How Do I . . . ?" element found either in the Main Menu or at the bottom of the Pathagoras dropdown features list. This context sensitive help tool provides short tips and 'how to' steps to accomplish a given function.
- ➔ The Pathagoras menu at the top of the Word screen makes Guides (labelled, 'Beginner', 'Intermediate' and 'Advanced'. Refer to these often to learn new things or brush up on old one.

1.11 Upgrading Pathagoras


Pathagoras is always changing. It is upgraded on a fairly regular basis as users offer suggestions to improve current features or to add new ones. It is this reason why Pathagoras is distributed only over the Internet.

To upgrade Pathagoras,

1. Display the Utilities/Settings screen from the Pathagoras dropdown features list.
2. Select the Upgrade tab. (You can check for the latest release version number by clicking the 'Check for Upgrade' button. Compare it to the information on the Upgrade tab to see if an upgrade is indicated.)
3. Click the button that reads "Download complete program from the Internet" and follow the prompts.


 Upgrading Pathagoras is a simple, painless process:

- Pathagoras preserves all libraries, books, settings, Instant Database records, etc. You will lose nothing when you upgrade.
- You do not have to uninstall a previous version of Pathagoras to upgrade.
- The transition from a demo version to a licensed version is seamless.

 You are entitled to the latest version of Pathagoras while you are within the period your initial Annual Support Agreement or subsequent renewals.

If you are not sure of the status of your Support Agreement, you can check by [clicking here](#).

Read more about the Annual Support Agreement in this section of the Manual.

And don't forget: if you ever have questions, [contact us](#)  ³²⁴.

The Pathagoras System

Getting Started

Part



2 Getting Started

2.1 Download and Install

Downloading Pathagoras

Pathagoras is distributed exclusively over the Internet. There is no 'boxed' version of the program.

Retail Version: When you purchase a retail license from the Pathagoras website, you will be directed to a site from which to download the full Pathagoras program. You will also be provided a serial number with your purchase receipt or confirmation by which you can unlock the program.

Demo version: Pathagoras also offers a free 90-day trial version. Download the Pathagoras Demo from the Pathagoras website at www.pathagoras.com.

You should save the downloaded file (called Pathagoras Setup.exe) to a location on your computer where you can readily find it. We recommend the desktop, at least until you have installed Pathagoras. Once installed, you can erase the Setup file.

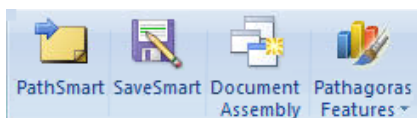
Installing Pathagoras

After downloading Pathagoras from the appropriate source, double-click on the Pathagoras Setup.exe file and follow the prompts. All files required by the program will be moved to their proper locations.

After you have installed Pathagoras, it will automatically open and automatically close alongside Word.

Pathagoras tools reside in a separate tab in the ribbon area of your screen:

The Pathagoras Toolbar sits behind the Pathagoras tab and looks like this:



Setting up Pathagoras

Pathagoras requires very little preparation to use its basic features. So long as you can see the Pathagoras tab, you are set for right now.'

Running Pathagoras

Actually, you don't 'run' or 'start' Pathagoras. Being a Word add-on, Pathagoras 'runs' whenever you run Word, and closes whenever you close Word. You don't have to do anything special beyond the installation discussed above to cause Pathagoras to run.

But what about the Pathagoras icon on my desktop?

That's just the installer. It's called "Pathagoras Setup.exe". Once Pathagoras has been installed, you can delete it.

2.2 Installation Checklist

All:

- ☐ Install 'full license' version of Pathagoras to the new computer.

Shortly after you have placed your order (within seconds, not hours), you should receive an email confirming your order. This email also contains the link to download Pathagoras from the server and your serial number needed to register your program. To install, it's a simple 'download and run' process. Pathagoras offers a distinctive 'deep purple' screen to denote that you are in the install mode. You will insert your serial number on the 4th screen of the install sequence. It goes in the lower text box in place of the word Trial.

You can use any User Name you wish in the upper box..

With just the install, you can immediately begin to create variables, scan the 'Pathagorized' and produce personalized text.

Multi-user licenses:

- ☐ Check pointers to your Instant Database records. If you want to share Instant Database records with others on the network, you should create a folder on a computer to which all users can point. Then, point the IDB path on your computer to that location and point the IDB path on all other computers to that same location.
- ☐ Check pointers to your *Alias* file. (By default, but not by necessity, the *Alias* file moves in tandem with your Instant Database records. But if you elect to store them in separate locations, make sure that all users have pointed their *Alias* file to the same folder.

Additional Network tools:

Keep in mind the following: Pathagoras can span your network with no setup beyond the initial installation. It is not necessary to take any of the below steps for Pathagoras to use and assemble document across the network. But if you, as administrator, wish to share Library and Profile setups that you have created on one computer with the other computers on the network, then you should create a mirror path.

- ☐ **Create a Mirror Path.**
- ☐ **Check pointers to your various books in your Libraries.**

With all settings checked and confirmed, you should be ready to roll.

2.3 'Pathagorizing' Your First Form

This exercise takes place without using a single tool or button on the Pathagoras toolbar. This exercise takes place by just using your keyboard. (What other program can say that?!?)

1. Open a document that you want to 'Pathagorize' (neuter). It doesn't matter from where this document comes*
2. Study the document. Look at all places where 'personal' or customer- or client-specific data presently exists. We are talking about names, addresses, dates, quantities, colors, etc. Anything that is likely to be different from one completed form to the next.

3. At the beginning of each of those 'personal places,' type a "[" (a square bracket next to the P on your keyboard. No quotes, tho').
4. At the end of each 'personal place' type a "]" (a square 'closing' bracket; again no quotes).
5. Replace the stuff in between the two facing brackets with generic sounding, not 'real people,' terms. E.g., "[Customer Name]"; "[Names of Children]"; "[product description]"; "[shipping date]"; etc. Because it is so easy to change anything in Pathagoras, it is not critical right now what terms you use.
6. After completing the above steps, "Jonathan E. Doeberg" might become "[Client Name]" and "4,342,654 widgets" might become "[quantity] [item ordered]". If "Mr. Jonathan E. Doeberg" appears in the address line, make the address line something like this: [Title] [Client Name].
7. Save this new document. For the absolute beginner, maybe save it in the same folder from which you drew it. Or, and we do recommend this, think about saving it in a new Forms folder (with an appropriate descriptive name). If you are moving and/or renaming, just do a simple 'SaveAs'.

You have now placed your first book onto a shelf in your library. This makes all documents in that folder available for document assembly. You have also 'Pathagorized' your first form. Repeat as appropriate to build your document assembly foundation.

The next sections of this Guide will take you through the actual processes of document assembly. When asked to select a book or navigate to a folder, you can use the books, folders and forms created in this section.

*Despite this statement, it will definitely matter pretty soon the locations you ultimately decide to store your documents. They should definitely be organized in one fashion or another by topic. But right now, that is a secondary concern.

2.4 Notes and hints

Couple of Notes here:

Compare these two

[Date]	[Date of Letter]
[Title] [Name]	
[Address]	[Client Title] [Client Name]
[City, ST ZIP]	[Client Address]
Dear [Salutation]	[Client City, ST ZIP]
	Dear [Client Salutation]

When the Instant Database (<Alt-D> is called this is that the user will see.

Compare the 'descriptiveness' of the various variables in the left side of the table to that on the right. Single word variables may not be of sufficient help to the person who will actually be completing the form. Likewise, single word variables like [Title] or [Name] jsut are not descriptive enough for the end use. You should always keep in mind the end user as you name variables.

Along these same lines, you should begin to develop early on a variable naming pattern and stick to it. Whether you choose 'Client Name' or 'ClientName' (no space) or 'Name of Client' doesn't matter to Pathagoras. Only 'consistency" matters. But we do have a suggestion-- don't use apostrophes. [Client's Name] is legal, but just adds a needless complexity. We recommend 'no apostrophes. [Client Name is just as descriptive., no?

2.5 Add Simple Optional block

Enter topic text here. Create an 'optional text' block. This is 'take or leave it' text. When encountered, Pathagoras will ask if the text should remain in the document.

- Choose (or create) a sentence or paragraph that you want to be 'optional.'
- At the beginning of the text, type "{" (no quotes).
- At the end of the optional text block, type "}" (again, no quotes).

Example:

[Date of Letter]

[ClientTitle] [Client Name

[Client Address]

[Client City, ST ZIP]

Dear [Client Salutation]

Thanks for your purchase of [quantiy] [product]. They should be delivered shortly.

{Please call me as soon as possible so that we can discuss the various options that [product] offers. }

Sincerely,

Roy H. Lasris

When the document is recalled, the text in the curly braces is processed first. (Press Alt-P to process the document manually.)

Once processed, then you would complete the variables --Alt-D.

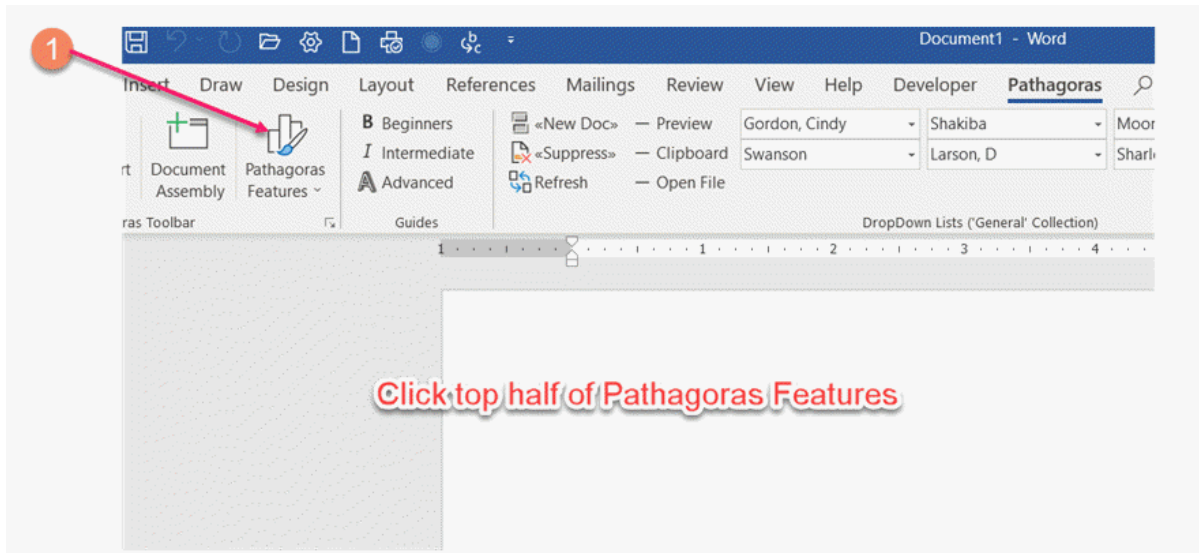
2.6 Default Document Display

Some like 'Show highlighted text' always 'on' so that text highlighting always displayed in the final draft; others don't. Some like their 'Comments' to always show; some don't.

To insure that your document will always be returned to a precise display you like, set your initial display defaults by following these simple steps:

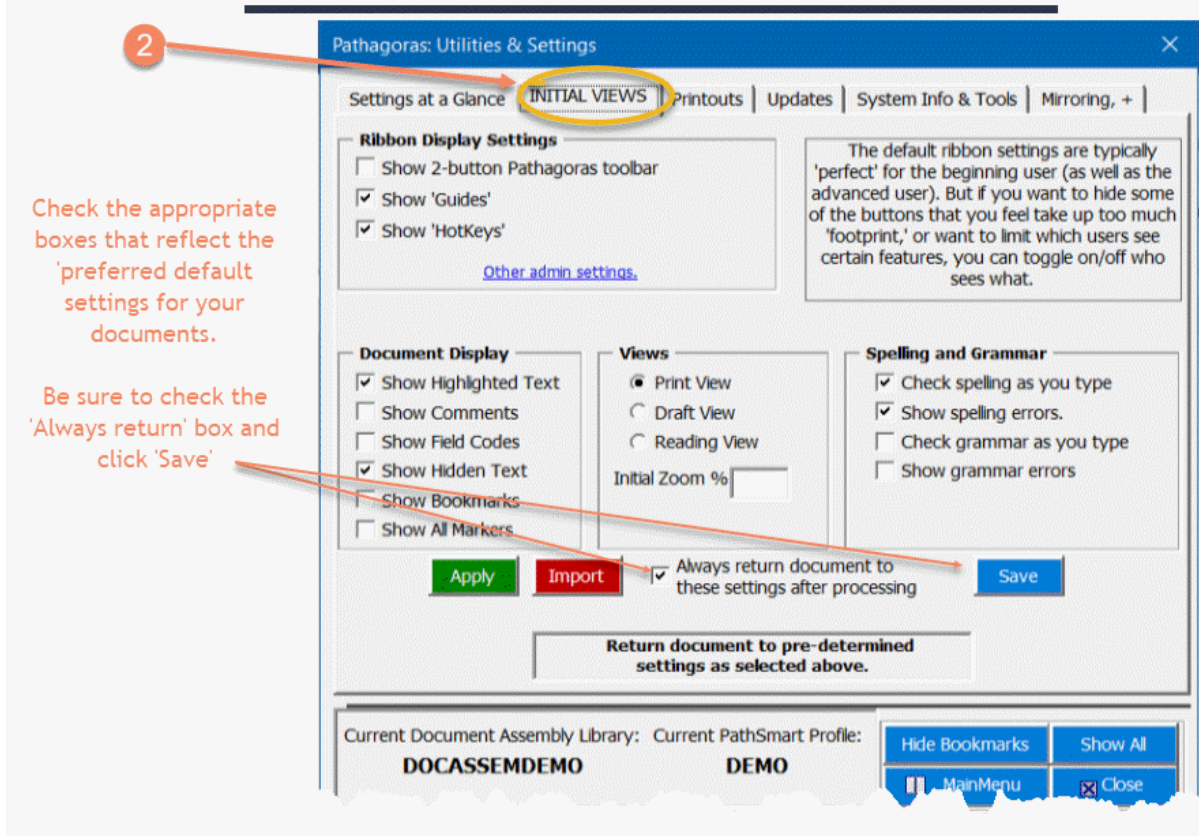
From the Pathagoras tab, click the top half of the Pathagoras features button and then click 'Initial Views'.

1



Click top half of Pathagoras Features

2



Check the appropriate boxes that reflect the 'preferred default settings for your documents.

Be sure to check the 'Always return' box and click 'Save'

Pathagoras: Utilities & Settings

Settings at a Glance **INITIAL VIEWS** Printouts Updates System Info & Tools Mirroring, +

Ribbon Display Settings

- ☐ Show 2-button Pathagoras toolbar
- ☒ Show 'Guides'
- ☒ Show 'HotKeys'

[Other admin settings.](#)

The default ribbon settings are typically 'perfect' for the beginning user (as well as the advanced user). But if you want to hide some of the buttons that you feel take up too much 'footprint,' or want to limit which users see certain features, you can toggle on/off who sees what.

Document Display

- ☒ Show Highlighted Text
- ☐ Show Comments
- ☐ Show Field Codes
- ☒ Show Hidden Text
- ☐ Show Bookmarks
- ☐ Show All Markers

Views

- ☒ Print View
- ☐ Draft View
- ☐ Reading View

Initial Zoom %

Spelling and Grammar

- ☒ Check spelling as you type
- ☒ Show spelling errors.
- ☐ Check grammar as you type
- ☐ Show grammar errors

Apply Import ☒ Always return document to these settings after processing Save

Return document to pre-determined settings as selected above.

Current Document Assembly Library: **DOCASSEMBLY** Current PathSmart Profile: **DEMO**

Hide Bookmarks Show All MainMenu Close

Make appropriate selections. Be sure to click 'Save' to retain settings for all documents.

The Pathagoras System

Variables - Definition

Part



3 Variables - Definition

A 'variable' is a place holder for personal information.

Variables are placed in strategic locations throughout a document to indicate where personal information ultimately will be placed in the final version of the document.

A '*simple variable*' is a single word or short phrase that serves as a place holder for something else. Typically that 'something else' is personal data such as a name or an address, but it can be much more complex than that. In Pathagoras, a simple variable is a word or phrase all enclosed between square brackets. E.g.: [Client Name]

A '*multiple choice variable*' is a series of pre-determined choices from which the end-user may choose. In Pathagoras, the various values of a multiple choice variable are separated by slashes. E.g.: [chocolate/vanilla/strawberry].

The following pages describe in greater detail how to creating and use variables.

3.1 Simple Variables

Creating variables with Pathagoras could not be easier. Standard keyboard characters are all you need..

If you are starting from scratch, with your cursor at a spot where you want to place a variable, type an opening square bracket. Next, type a 'variable sounding term'. Then type closing square bracket. Your end product should look like this: [Client Name].

If you are Pathagorizing an existing document, peruse the document for places where personal information currently exists. If the text is currently a personal name, change it to a variable sounding word and then put brackets around it. So, change "Abraham Lincoln" to "[Client Name]" and you have done it.

Note the following:

- You can manually add variables by hand typing them into your source documents.
- If you work with documents that you obtained from a third-party which already contain bracketed variables, you actually have a 'Pathagorized' document. (And you can add even more variables to those documents if you wish.)
- When deciding on variable names, keep the end user's reaction to the variable in mind: "[Date Contract Signed]" is probably better than "[Date]."
- Type the variable the way you want it to appear in the final product. If you want the variable to be in ALL CAPS, type the variable that way at that particular location. So, at the top of the document, you might have [CLIENT NAME] and in the document body, you might have [Client Name]. Pathagoras will replace variables following the style it 'sees' in the document. [Read more about that here](#)⁴².

Check out this video on creating and using simple variables: [Creating Simple Variables](#)

Modifying variable names: Sometimes the name you initially select for a variable turns out not to be the 'best' choice. Not to worry. You can easily change them to a more appropriate value. If you are just getting started, and have only created a few variables, manually changing

them is probably the easiest course. If you have a lot variables and you need to change them *en masse*, Pathagoras has tools to help.

3.2 Multiple Choice Variables

A multiple choice variable is a simple variable containing a series of words or phrases that you want to offer to the end user as separate choices. Each choice is separated from the others by a plain text slash ("/").

Examples of Multiple Choice Variables:

[he/she/it/they] [his/hers/its/their]

[red/white/blue]

[chocolate/rocky road/vanilla cream swirl/triple fudge brownie delight]

[Poughkeepsie/Patchogue/Irondequoit/North Tonawanda/Skaneateles]

A multiple choice variable is typically used when the author wants to provide the end user with a limited range of possible 'answers.' But, as the last example above illustrates, spelling challenges can be another reason to provide a multiple choice variable.

Usage examples:

When the entire range of colors is a possible answer, a simple variable like "[color of widgets]" would work fine.

However, if you wish to limit the selection of colors to just a few choices, you can list those colors within the variable itself as a multiple choice variable. That way, the end user understands the limitation, and will be led to provide an acceptable answer:

Thank you for your order. We will deliver [number] dozen of [yellow/green/blue/orange] widgets within 2 weeks.

When Pathagoras encounters a multiple choice variable, it parses out the individual choices and presents them in an easy to select drop down list (in the [Instant Database](#)⁴⁶ module) or selectable buttons (in GotForms?). Try it out. Type a simple multiple choice list onto your editing screen. (You can even copy and paste the examples above.) Press <Alt-D> to activate the Instant Database. Press the <Scan> button. The variables appear at the left, the dropdown lists at the right.

What if I have a 'lot' of choices?

Pathagoras allows you to create an unlimited number of choices within a standard multiple-choice variable. However, if you need to list more than 5 or 6 we recommend that you use [*Aliases*](#)¹¹⁴, a powerful feature discussed in the next section that allows you an unlimited number of choices to be presented in your document as a single word. (Think *States* as the alias for the 50 United States, *Countries* for all the countries in the world, and of course, *Flavors* to contain all of the possibilities for Ben & Jerry's® Ice Cream.)

What if I want to use a multiple-choice variable in different sections of my document, but be able to select different values?

Making unique names for each variable is easy to do for a simple variable. Just use a different name, or at the very least, append a number or other character at the end. E.g., "[Variable1]", "[Variable2]", "[Variable3]", etc.

To make otherwise identical multiple choice variables 'unique', append the '@' sign and a distinguishing *single* character or *single digit* number at the end of the list of choices. E.g., "[Red/Blue/Green@A]", "[Red/Blue/Green@B]", "[Red/Blue/Green@C]", etc. (Pathagoras knows not to include the distinguishing character or number as part of the last element of the multiple choices when it encounters the '@' sign.)

An alternative approach is to [add a title](#)⁴³ to the variable. A title is simply a descriptive word or phrase typed at the beginning of the variable and separated by a colon. E.g., [custodian:mother/father], [grantor:he/she]. If you have multiple multiple-choice variables that you want to change in tandem, use a groupname.

What if a slash is an integral part of the variable?

If you need to choose among variables that themselves contain slashes, e.g., "1/2" and "and/or" either:

Put the specific choice in quotes (the quotes will be stripped from the final copy). For example: ["1/4"/"1/2"/"3/4"/1 inch]; ["Renovation and/or repair."/"New construction."]

OR

Use the 'slash OR' ('/OR') separator (upper case OR mandatory). E.g., [1/4/OR1/2/OR3/4/OR1 inch]; [Renovation and/or repair./ORNew construction.]

Can I include a picture or a Word 'field' (those 'gray' embedded things) as one of my choices?

No. Variables are 'plain text' in nature. Therefore Pathagoras can handle only choices that are typed as plain text and which can be stored as a permanent record in plain text. HOWEVER (and probably better for most purposes) you can present pictures, Word fields and a wide range of other non-text objects within an <<*Options*>> or <<*Optional*>> text block. This powerful alternative should accomplish what you are seeking. See [<<*Options*/*Optional*>> text](#)¹⁷².

What if I need to repeat the same variable selection in a different section of my document, or make a choice in another section that is based on my initial selection?

Not a problem. That is what [!Groups!](#)¹³⁶ (two sections down) is all about.

Can I set one of my values to the 'default' selection?

Yes. Just precede the default value with the '#' (hash tag) sign. E.g., "[chocolate/rocky road/vanilla/#triple fudge brownie delight]"

See Also: [*Aliases*](#)¹¹⁴

[!Groups!](#)¹³⁶

[Titling](#)⁴³

3.2.1 The Final 'And'

When you need to present lists of items in a stacked format, you may wish to include to the right of the next to last item an 'and' or an 'or' to signal that the list is about to end.

Manual Approach: When the list is static, all you need to do is type the appropriate conjunction between the next to last and last item. But when the list is fluid, a much more challenging issue is presented, one that is particularly difficult to program for.

1. Here is my list of jobs:
 - a. Take out the garbage;
 - b. Mow the lawn;
 - c. Wash the dishes;
 - d. Vacuum the floor.

Viewing the example above, you can see some of the complexities. The first issue deals with the punctuation of the intermediate and final choices. Item d. ends with a period, whereas the others with a semicolon.

You could append an 'and' at the end of a-c and then remove (manually or with the help of a program) the unnecessary 'and' when the document is fully assembled.

But that necessarily assumes that d. will always be the last choice. It doesn't work if you choose a and b (and not c and d) because the final punctuation mark is a semicolon, and you want a period.

Automated Approach:

Other programs make setting up the handling of the 'final and' a programming nightmare.

It's easy *and all plain text and plainly visible* in Pathagoras.

If the last item is always the last item (example, the last item is "Other _____."):

Since you are not worried about interim punctuation. Just add <&FINAND> (the case doesn't matter. ,<&FinAnd> works just as well.)

```
Here is my list of jobs:
  a. Take out the garbage;<&FINAND>
  b. Mow the lawn;<&FINAND>
  c. Wash the dishes;<&FINAND>
  d. Vacuum the floor;<&FINAND>
  e. Other _____.
```

There are two part to this code. The first part is the '<&', and and text you want (here we use FIN for final. The second part is 'AND' and a final '>'

If the last item can be any item in the list, including the first item,

There are 3 parts to this code. The first part is '<&' and any identifying text (here we use FIN for final. The second part is the final lines punctuation (which can only be a period) and the third part, 'AND' and a final '>'. The code must be appended to each line that could be the final line.

```
Here is my list of jobs:
  a. Take out the garbage;<&Fin.And>
  b. Mow the lawn;<&Fin.And>
  c. Wash the dishes;<&Fin.And>
  d. Vacuum the floor.
```

NOTE: The 'identifying text' must be unique for each list. This is how Pathagoras knows how to process multiple lists in your document as individual units. But they can be similar. For example <&FIN1AND> and <&FIN2AND> are unique enough to identify the two lists below.

```
Here is my list of jobs:
  a. Take out the garbage;<&FIN1And>
  b. Mow the lawn;<&FIN1And>
  c. Wash the dishes;<&FIN1And>
  d. Vacuum the floor.

This is a list of your jobs:
  a. Do the laundry;<&Fin2And>
  b. Edge the yard;<&Fin2And>
  c. Empty the dishwasher;<&Fin2And>
  d. Mop the floors.
```

OR conjunction:

For the Or conjunction, just substitute 'Or' for 'And' in any of the above setups.

Notes:

Before adding the <&. . .> text, make sure the paragraphs have all the punctuation you want using the assumption that all elements will be chosen. Pathagoras doesn't 'add' the punctuation to the last line; it backspaces and replaces it.

The 'And' or 'AND' or 'and' (or 'OR' or 'Or' or 'or') that you type in the list will control the replacement.

The above lists suggest that the user will manually remove the items that are not to remain on the list. And if you do that, press Alt-P to process the <&Fin. . .> blocks.

The below example applies '<<*Options*' blocks to the last list shown above. When this list is processed, the '<&Fin. .>' blocks are processed automatically with the rest of the document. (When processing a list with a 'Final And/Or' tag, you are presented 'connectors' ('and', 'or', 'space', etc.) after you select the second item. Choose 'None' because the connector is already selected.)

```
Here is my list of jobs:
<<*Options*!MyJobs!Garbage/Lawn/Dishes/Vacuum*
  a. Take out the garbage;<&Fin1And>/
  b. Mow the lawn;<&Fin1And>/
  c. Wash the dishes;<&Fin1And>/
  d. Vacuum the floor.
>>
This is a list of your jobs:
<<*Options*!YourJobs!Laundry/Yard/Dishwasher/Mop*
  a. Do the laundry;<&Fin2And>/
  b. Edge the yard;<&Fin2And>/
  c. Empty the dishwasher;<&Fin2And>/
  d. Mop the floors.
>>
```

3.2.2 *Aliases* variables)

You can reference a multiple choice list of values using an *alias* ('starred') variable.

An alias is simply a word (or two) bounded by asterisks (*) (that's what makes it an alias term) enclosed within brackets (that's what makes it a variable).

Read more about [*Aliases*](#)¹¹⁴ at this link.

3.3 Modifying Existing Variables

Enter topic text here.

3.4 Existing 'Gold Star' Documents

You likely have great documents, perhaps the perfect form you just prepared for an existing client, that you want to serve as a model when you are creating forms in Pathagoras. We call these 'Gold Star' documents. They are replete with 'John Does' and 'Mary Smiths,' and you anticipate that it will be a pain to have to replace each name, address, date, etc., one at a time.

One approach that will save you time is to use Microsoft's 'Find and Replace' tools to replace John Doe with [Client Name]. Because Pathagoras is a plain text document automation program, that approach is perfectly okay.

But Pathagoras offers another (superior) way to accomplish this. We call it our 'Variable Creation Wizard' and you activate it by pressing <Alt-V>. Here are the steps to using it:

- o Highlight a 'real name' or other value in the document that you want to convert to a variable.

- Press <Alt-V> to call the Wizard. Type in the desired variable in text box beneath the 'Replace with?' question the desired variable name, e.g. "Client Name". (No need to add brackets.) Press the Next>> button.
- All instances of 'John Doe' in the document are replaced with [Client Name].
- Note that all case and emphasis attributes in the original document are preserved. (I.e., if original name was JOHN DOE at top of document and John Doe in body, the names are replaced as [CLIENT NAME] at top and [Client Name] in body, exactly the same style as the original.) .

3.5 Emphasis (Bold, Italics, CAPS)

As text is replaced via Instant Database, the replacement text inherits the formatting of the text it replaces. This allows you to provide a single replacement value to replace essentially identical variables but with different 'looks' (i.e., different case and emphasis attributes). (If you have tried other programs, you may recall that they require you to add case and emphasis attributes to the grey fields tied to the variable. No so with Pathagoras. Just type the source document with the look and feel you want and Pathagoras will follow that lead. So:

if the variable in the source document is ALL CAPS, the variable will be replaced by ALL CAPS characters, even if the operator types the replacement text in lower case letters.

if the variable in the source document is BOLD and ITALICIZED, when the variable is replaced, it will appear with BOLD and ITALICIZED characters.

The color or emphasis attributes (bold, italics, underline) of the variable are.

EXCEPTION: If you type the replacement text in ALL CAPS, Pathagoras assumes that you mean it, and that you intend all replacements throughout the document to be in ALL CAPS. Therefore, when you type replacement text into the right column of the Instant Database screen, unless you want all caps, you should type the replacement text in regular upper and lower case style ('John Q. Doe', not 'JOHN Q. DOE').

Bottom line: do not worry about capitalization, bold, italics or underlines as you are completing the IDB screen. Type text in a normal fashion. Typically you will type "Upper And Lower" case for names and titles, lower case for most everything else.

Override exception: In some documents, you don't want the sophisticated replacement protocol offered by Word to control. You want the replacement text to be identical to that typed in as the 'replacement text.' That is possible. To enable 'Exact Replacement,' click the 3rd element of the Advanced Array check-boxes just above the Next button.



Advanced Array. Click 3rd button to replace text exactly as typed.

Don't worry if you cannot remember the "third checkbox." Simply hover the mouse over any box in the advanced array. A prompt will appear describing the function of the box.

[Click here to read more about the Advanced Array.](#)

3.6 Titled variables

If you have variables that, standing alone, don't give the end user enough context for an intelligent response, you can add a 'title' to the variable. A title is simply a word or two (no more than 20 characters total) at the 'front' of the variable term, separated from the term with a colon. E.g., [\[Title:variable\]](#).

Titles are especially helpful with multiple choice variables that could apply to different actors. For example, the multiple choice variable [he/she/they] appearing in the Instant Database screen gives the end user no idea as to who the pronouns refer. But add a title, and the selection is much clearer. E.g., [\[beneficiary:he/she/they\]](#).

Each separately titled variable will appear as 'stand-alone' to Pathagoras. The title will appear at the left of the IDB screen as discovered by the scan. It also shows at the top of the multiple-choice list at the right.

Illustration:

[\[Special Order:Yes/No/Not Applicable\]](#)

[\[Priority Mail:Yes/No/Not Applicable\]](#)

[\[Health insurance provider:Husband/Wife\]](#)

[\[Custodian:Mother/Father\]](#)

[\[Special Order:Yes/No/#Not Applicable\]](#)

Notes:

'Title' vs. 'GroupName': A title is typically used to modify and distinguish otherwise identical variables from each other. They are also used to offer guidance to the end-user as to what information is being sought. So, a title is 'informational' in nature. A 'group name' is very much functional. It is used to select the same relative item from separate lists. Think pronouns: 'he/she/it' and 'his/hers/its'. Think 'Attorneys' and their respective 'bar numbers.' Read more about [GroupNames](#)^[136] in the preceding page.

A groupname can be used as a title, but the better practice is not to do so. A groupname takes a bit more computer time to process as it searches for matching group entries. (So, groupnames should be used only when you wish the other member of the group to 'behave' in the same manner as the first member of the group, use titles when you just need a label.)

You cannot combine titles and groupnames.

Every variable potentially could be 'titled.' After all, more information is better than less information. But the better practice would be that you not add a title a variable unless it's really needed to guide the end user. A title takes up space in your document. If it doesn't add to the understandability or functionality of the variable, don't use it.

3.7 Default Value of a Variable

Default values:

A variable containing a title and a single value can be used to provide a 'default value' in the right side of the Instant Database screen. Examples of such variables: [product:#widget] and [flavor:#strawberry].

You can set a default value for a variable in these ways:

- If a simple variable, in the first instance of the variable in the document, add a colon and a # sign (colon-hashtag) after the variable name and provide the default value. E.g., [Favorite fruit:#Oranges].

If you use the variable later in the document, just use [Favorite fruit]

- If the variable is a multiple choice variable, add a # (hashtag) just before the default value in the listing. E.g., [chocolate/vanilla/#coffee mocha delight]
- If the variable is an *alias*, open the alias list spreadsheet and add a hashtag in front of the default value.

When the document is scanned, the complete variable appears in the left side of the Instant Database screen. The default value is placed in the right column.

For other variables in the document which you wish to have the same value, you need only use the variable (the text before the colon. In other words, after the first entry in the document which establishes the default value, the text before the colon (irrespective of the default) is considered the variable. Example:

Thank you for your order of 50 [product:#widgets]. You order entitles you to a coupon for 1 gallon of [flavor:#chocolate mint] ice cream.

You should expect delivery of your order of [product], and your coupon for the [flavor] ice cream, on your doorstep within 3 business days.

Again, thank you for your business. We hope you enjoy both the [product] and the [flavor] ice cream.

Of course, the 'default value' is just a value pre-typed for you into the Instant Database screen as a convenience. You can change the default to any other value you desire.

The Pathagoras System

Instant Database Screen

Part



IV

4 Instant Database Screen

No document assembly system is complete without an efficient means by which it can identify the variables in the document and replace those variables with personal data.

Definition: 'Variables' are place holders strategically placed throughout your document. After a document is prepared for a specific client or customer, the variables will be replaced with personal data. See [Variables](#) ³⁶.

Discussion:

After you have 'assembled' a document (i.e., called in the desired blocks of text, made your multiple choice decisions and deleting those text sections that are not relevant to your client's need), you should have a perfect 'rough draft' of the final document. What remains, however, are the bracketed [variables] you (or another) inserted into the remaining clauses. They need to be replaced with *personal values* (names, addresses, quantities, colors, etc.) to reflect the objectives of the writer and the needs of the client/customer.

The primary 'variable replacement' tool in Pathagoras is the 'InstantDatabase' ('IDB') module. The IDB screen is displayed by pressing the keyboard hot-key combination <Alt-D>.

4.1 The basics

Instant Database is activated from the keyboard by pressing <Alt-D>. Here is a screen shot of what you will see.

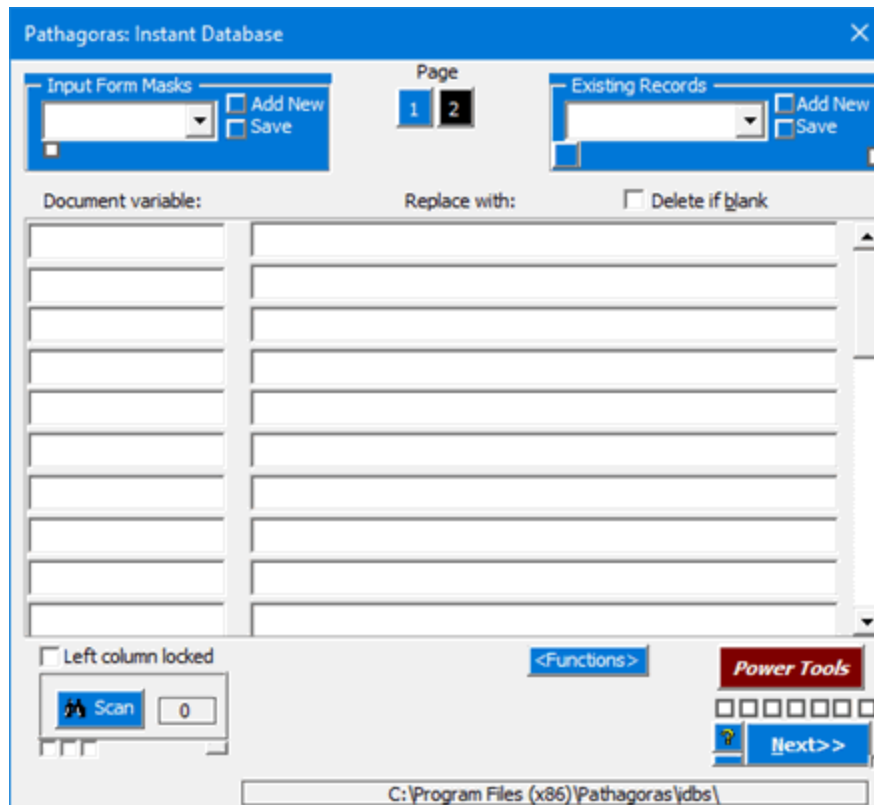


Figure 1. Instant Database Mask
(initial view)

- In the upper left and right corners are dropdown lists:
 - The Input Form Masks dropdown contains all available input templates. These are forms that you create and save. They contain variables (typically related to a specific topic) and completion tips (options). With masks, you can control the order in which the variables appear in the window.
 - The Existing Records dropdown contains all of your saved data records, in alphabetical order. (The top of the list also displays the 9 most recently used records for easy selection.)
- Between the drop downs are the Page selector buttons. By default, two buttons are shown, #1 for the first 30 variables and #2 for 31 thru 60. Initially 60 variables are in play. (30 variables per page x 2 pages).
- The middle section of the screen reflects the document variables. When the document is scanned, or a mask or record is recalled, the left column will contain the bracketed "variables". The right column will display the 'replacement' values (personal data) for each variable.
- To fill the screen with variables, select an item from Masks, from Existing Records or scan the document (button at lower left).
- When you are ready to signal Pathagoras to replace the variables at the left with the values typed at the right, click the **Next>>** button in the lower right corner of the screen.

- To access a series of helpful tools and utilities, click the red Power Tools button. [Power Tools](#)⁵⁵ are discussed in a separate section of the Manual.



You can re-size the Instant Database screen to your liking.

- If you just want to temporarily expand the screen to see a variable or value that is cut off by the current view, drag the re-sizer handle in the lower right corner of the screen. (This tool re-sizes only the screen width.)
- To more permanently re-size the screen, click the red <Power Tools> button and look for the Resizing cluster. Change the width or height of the screen as desired. Be sure to click the button in the center of the cluster to lock in the new size.

See also:

[Variables](#)³⁶

[Multiple Choice Variables](#)³⁷

[Groupings](#)¹³⁶

[File Location](#)¹⁰⁶

Power Tools

[More Tools](#)⁵⁷

4.2 Scan

If this is the first document you are completing for a client or customer, you will want to 'Scan' the document for bracketed variables:

1. Press the <Alt-D> key combination to display the Instant Database screen. The screen will be blank. See [Figure 1](#)⁴⁹, below.
2. Click the blue **Scan** button in the lower left portion of the IDB screen.
3. Pathagoras will display the results of the scan into the left column of the IDB screen. See [Figure 2](#)⁵⁰. (If multiple choice variables were discovered during the scan, the choices are show in the right column, but otherwise the right column will be blank, awaiting replacement values from the user.)

→ Notes:

- All bracketed words are presumed to be the document's variables. Occasionally, Pathagoras will identify a bracketed term that you did not intend to be a variable.
Example: Author and secretary initials are frequently presented at the foot of a letter in brackets. they will be picked up on the scan. When that happens, you can simply ignore it. It will remain in your document perfectly intact.
- Pathagoras' default 'enclosing' characters for Scanning are '[' and ']'. However, you can change the default characters to any character or character sets of your

liking. They can be a single character (more traditional) or up to 3 characters (e.g., “[*&” and “&*]”)

Exception 1 to the above 'anything goes' rule: The enclosing characters “<<” and “>>” are reserved by Pathagoras for other purposes. They cannot be successfully used as enclosing characters for the Instant Database system.

Exception 2 to the above 'anything goes' rule: The enclosing characters for variables must be different from those chosen for [Simple Options](#)²⁰²¹. While it may seem obvious, it is still a 'rule'.

4. 'Scope' of the Scan. By default, Pathagoras will scan for variables in just the 'main' document (and not within headers, footers, text-boxes and other non-main document locations). However, if your documents contain (or may contain) variables in headers, footers, etc., just check the box just above the Scan button that reads "Scan in Headers, Footers, etc." The setting is 'sticky' and will remain set for all future sessions (unless, of course, you uncheck it).
5. Once the scan is done, your job is to provide replacement data for each variable. Hand-type the personal data to replace the variables. See **Figure 3**. It is perfectly fine to leave the value for a variable 'blank.' You can complete it later.
 - If a Data Record was previously saved for a specific client or customer, you can simply recall the Record. It will appear under the Existing Records list in the upper right section of the screen. No retyping required (unless of course, you need to change a specific value from the saved record).

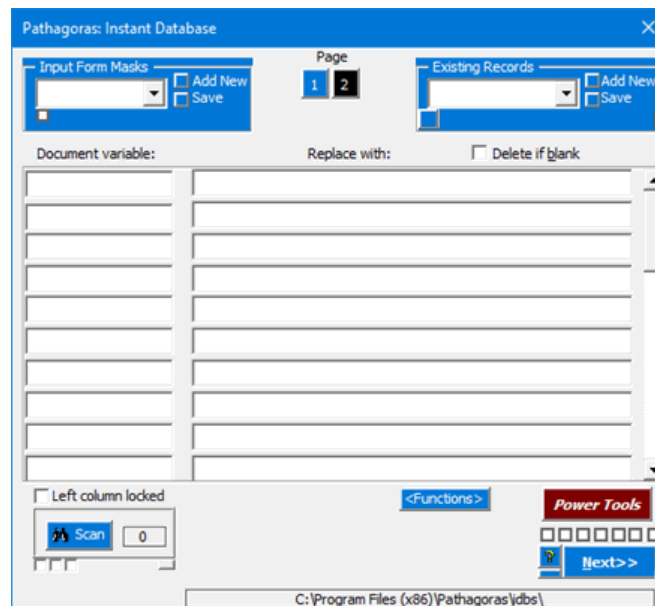


Figure 1. A 'blank' Instant Database screen. The 'search for' text (typically a Document Variable) appears in the left column. You would provide the 'Replace With' text in the right column.

The screenshot shows the 'Pathagoras: Instant Database' window. The 'Input Form Masks' section on the left contains a list of variables: [Testator Name], [Testator Address1], [Testator Address2], [Testator City], [Testator State], [I married/am/am not], [Spouse Name], [I spouse/husband/wife], [Child@1 Name], and [Child@1 DOB]. The 'Existing Records' section on the right has buttons for 'Add New', 'Save', and 'Delete'. The 'Page' indicator shows '1' of '2'. The 'Document variable:' section has a 'Replace with:' field and a 'Delete if blank' checkbox. The 'Left column locked' checkbox is checked. The 'Scan' button is visible. The 'Vars in Document' section has checkboxes for 'Hide variables not present in this document' and '& hide completed variables'. The 'Power Tools' section has a 'Next>>' button. The status bar shows the file path: C:\Program Files (x86)\Pathagoras\dbs\.

Figure 2 Document scanned. Bracketed variables placed in left column. Multiple choice variables are shown in dropdown lists.

The screenshot shows the 'Pathagoras: Instant Database' window with the same form as Figure 2, but with data entered in the right column. The data is as follows: [Testator Name] is John Pathagoras, [Testator Address1] is 1223 Apollo Boulevard, [Testator Address2] is Suite 9-C, [Testator City] is Athens, [Testator State] is Georgia, [I married/am/am not] is am, [Spouse Name] is Mary Q. Aphrodite, [I spouse/husband/wife] is husband, [Child@1 Name] is Artemis, and [Child@1 DOB] is 6/14/2002. The 'Left column locked' checkbox is checked. The 'Scan' button is visible. The 'Vars in Document' section has checkboxes for 'Hide variables not present in this document' and '& hide completed variables'. The 'Power Tools' section has a 'Next>>' button. The status bar shows the file path: C:\Program Files (x86)\Pathagoras\dbs\.

Figure 3. A 'completed' Instant Database form.

See Also:

[Data Records](#) ⁵²

Masks

4.3 Completing a Variable

To complete a variable, type the value with which you want the variable to be replaced in the right column. If the variable is multiple choice, select the entry from the drop down list presented.

When you finish completing all your variables, you will should click the Next>> button. When pressed, Pathgaoras will ask if you want to create a new record or, if appropriate, update the existing record if changes were made. It will then immediately proceed to replacing the variables with the assigned values throughout the document. We recommend that you always click Next unless you do not want the variables replaced.

Two other buttons are 'possibilities', but Next>> is almost always your best choice as it automatically triggers these two anyway:

- The "Add New" button will create a new record based on the current on-screen data.
- The "Save" button will update the record showing in the Existing Records text box (assuming that you called up an existing, or an earlier added) record. Pressing "Save" when there is no record showing in the Existing Records text box is the equivalent of "Add New."

When you have finished providing personal data in the right column of the Instant Database screen, press the **Next>>** button. This is where the "database" part of "Instant Database" comes in:

- Pathagoras will ask if you want to save the personal data as a new data record.



- If you plan to compose more documents for this customer or client (e.g., future letters, forms, contracts, pleadings, addenda etc.), you should say 'Yes.' When prompted, provide a meaningful and unique name for the record. (Typically, the client or customer's name, e.g., "Lastname, Firstname" meets this criteria.)

If you do click 'Yes,' then when you create another document for the customer or client containing some or all of the same variables, you need only drop down the Existing Data list (upper right side of screen) and locate the record. That is why we call it 'Instant Database.'

- Of course, you can also say 'No.' If you believe that the data is not necessary to keep, that should be your answer.

After your 'Yes' or 'No' response, Pathagoras will make the replacements. It should take only a couple of seconds.

4.4 Data Records

A 'Data Record' is a single or unique file for a specific client, patient or customer. From a practical standpoint, in most cases it *is* a single client, patient or customer, but this should be further clarified.

A 'Data Record' is simply the pairings of variables and the values recorded onto the Instant Database screen and which is saved to a file on your computer. This data record can be reused with other documents containing the same variables, avoiding reentry of the same data.

Creating a new Data Record:

After document assembly ("*ad hoc*"):

Fully discussed under the [Scan](#)⁴⁸ sub-heading. Creating the Data Record after the assembly of the document is the 'classic,' but by no means the required, technique. You can create a Data Record at any time.

Prior to document assembly:

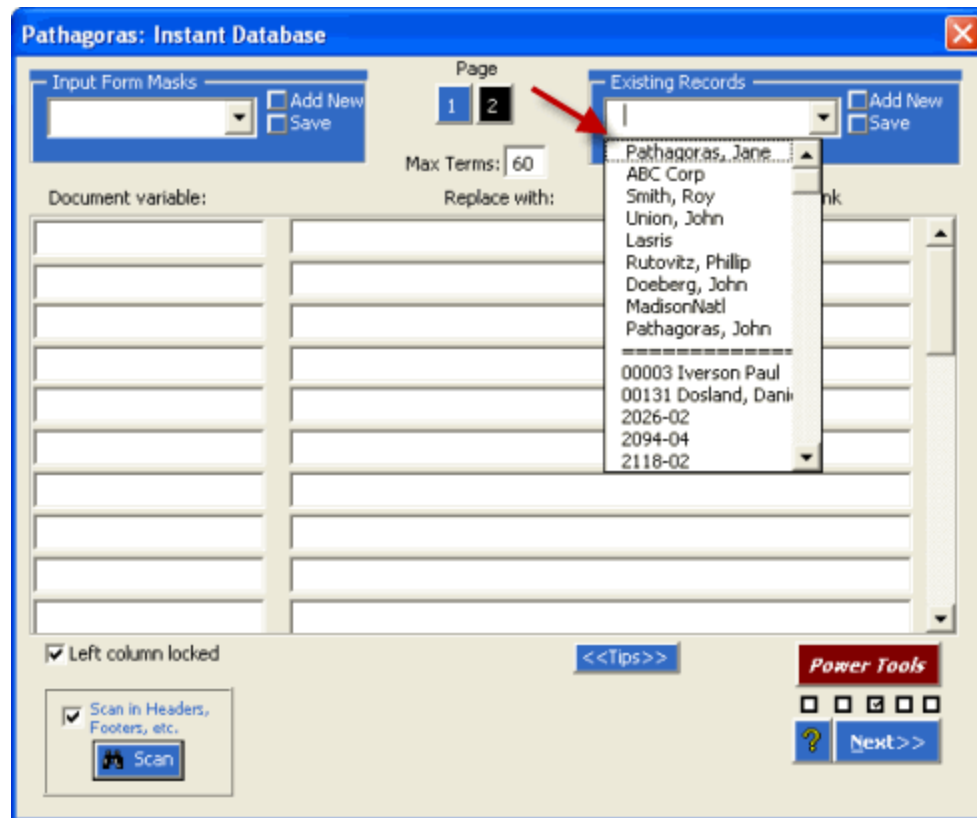
Pathagoras understands that many offices may wish to input a client or customer's data independently of a document assembly session. So, a data input clerk who has no document assembly responsibilities can still create a Data Record for use by other staff members for a later document assembly session.

In order for this feature to work effectively, you must have previously created an 'Input Form Mask' into which the personal data can be entered. An Input Form Mask is simply an Instant Database screen which contains a complete list of the variables for which you wish to capture personal data. The mask becomes the vehicle by which personal data can be requested and then stored.


See Masks for the steps needed to create an Input Form Mask.

Using Data Records -- the 'database' part of "Instant Database":

Assemble a *second* document for a client or customer. Press <Alt-D> to display the Instant Database screen. Instead of scanning for [bracketed] variables or recalling a mask, click the Existing Records drop down list in the upper right corner of the Instant Database screen. Find and click the client or customer name you saved earlier. Make any changes to the data that may be appropriate. Click **Next>>**. That is all!



To call in an existing record, click on the Existing Records drop down list.

 The original default location where Pathagoras stores Input Masks and Data Records is:
"C:\Program Files\Pathagoras\IDBS".

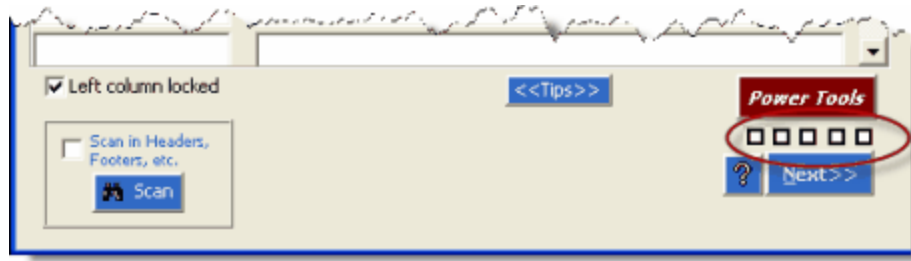
- You can easily navigate to this location using ordinary Word and Windows navigation techniques.
- You can easily change this location. Those who have purchased a network package may wish to point the default location to one on the server. See [IDB File Locations](#) ¹⁰⁶

There is no limit to the number of times that you can reuse a Data Record. There is no limit to the number of data records you can create.

The last used records (up to 9) appear in the top portion of the 'Existing Records' drop down list.

4.5 Advanced Array

Immediately above the Next button are 7 squares that (intentionally) appear to be ornamental. However, each of these square serves a purpose. We call this section of the screen the 'Advanced Array' because the boxes of the array are likely to be used only by advanced users -- those who are already familiar with basic program operations.



Advanced Array

From left to right, here is what each box does when checked:

- ☑ **Apply the Instant Database against all open documents.** This is a wonderful tool if you typically compose several documents simultaneously want to complete them all at the same time.



This feature can also be used to Scan all open documents for variables. Just check it before you click the Scan button.



This feature can also be used as multi-document '*Bulk Edit*' tool. By 'Bulk Edit' we mean searching and replacing multiple terms in multiple documents.

Keep in mind that IDB is essentially little more than a fancy "search and replace" utility. **It can be used to modify any of document.** So, from a blank IDB screen, type the 'old text' in the left column and type the new (replacement) text in the right column. Click the box and then press Next.



The resulting action is a standard 'search & replace', but it will be against *all open documents*.



If your goal is to replace in all documents in a folder (as opposed to all open documents), [click here](#)⁵⁵ for those steps.

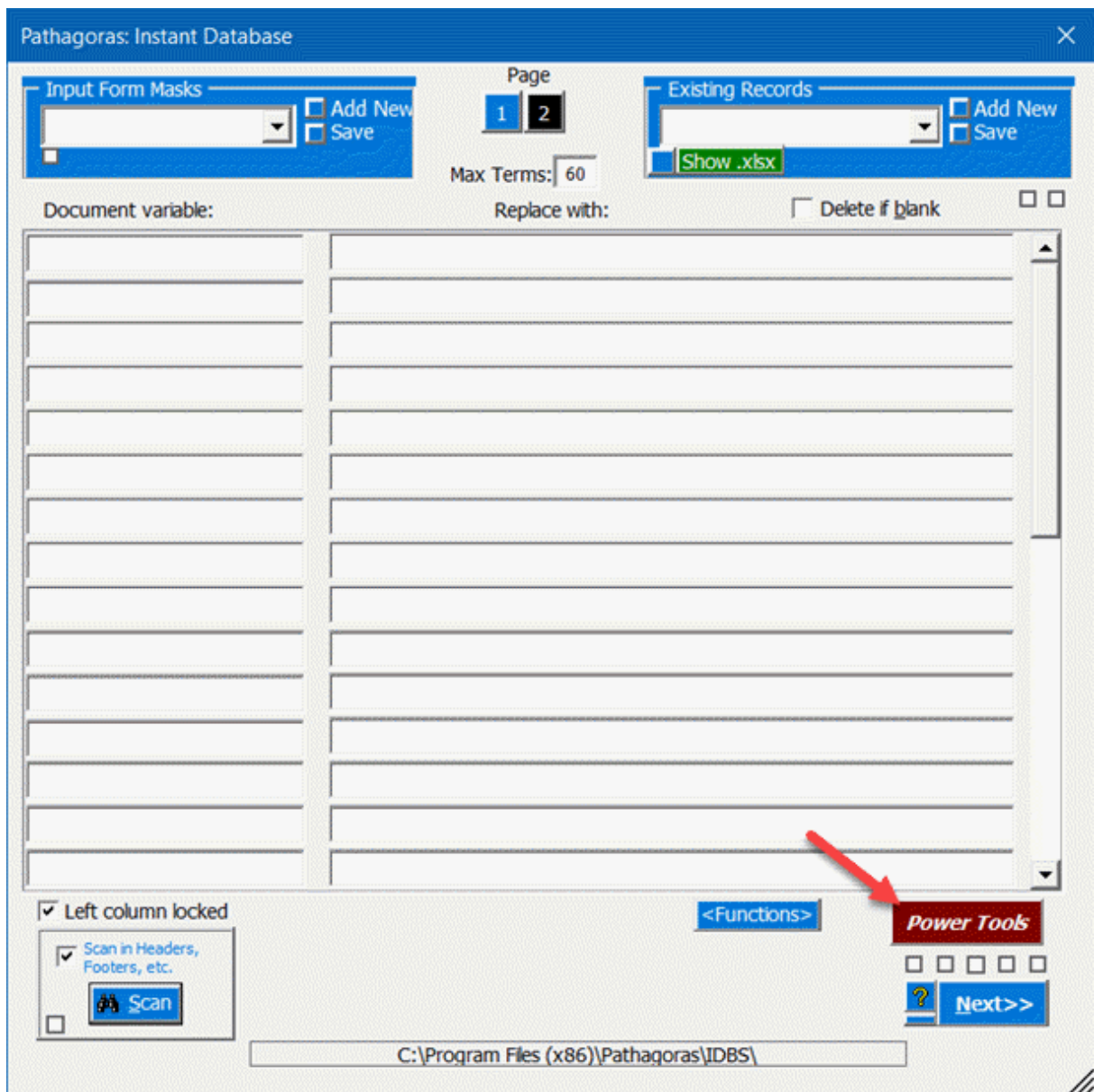
- ☑ **Do not process <<*Options/Optional*>> text blocks** after clicking Next. (By default, after replacing variables, Pathagoras will process terms within [double angle brackets](#)³⁰⁷).
- ☑ **Do not attempt to preserve emphasis** (bold, underlines, CAPS, etc.) when processing replacements. By default, the styles in the document control how the replacement. When you want Pathagoras to NOT change the text to reflect the style presented in the document.
- ☑ **Pure Search and Replace:** The value in the right column will precisely overwrite the document text shown in the left column. Can be use when the variable or replacement has Pathagoras coding that might otherwise be converted to an unwanted or restricted value.
- ☑ **Suppress "Do you want to save/update the matter record?" pop-up.** The question will not be asked, and changes to existing Matter record will not be preserved. (You can permanently suppress the pop-up via the Instant Database setting screen. The switch is under the 'Miscellaneous Settings' tab.)
- ☑ **Turn off screen display.** (Speeds up processing of very large documents.)

- ☒ **Save record in Unicode** For use by certain users in non-English language locations that require extended character sets.

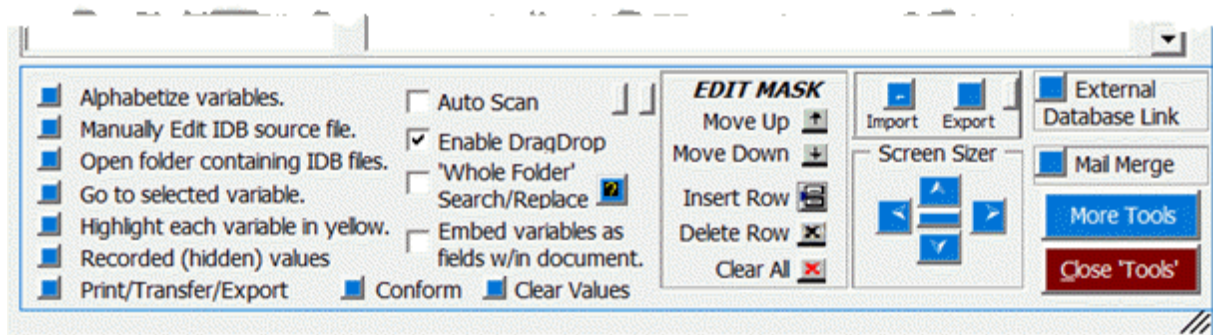
i Don't worry if you cannot remember the Advanced Array functions. Simply hover the mouse over any box in the advanced array. A prompt will appear describing the function of the box.

4.6 Power Tools


A wide range of tools and additional features reside behind the <Power Tools> button on the IDB screen.



Pressing it reveals a collection of editing tools and other resources at the bottom of the screen.



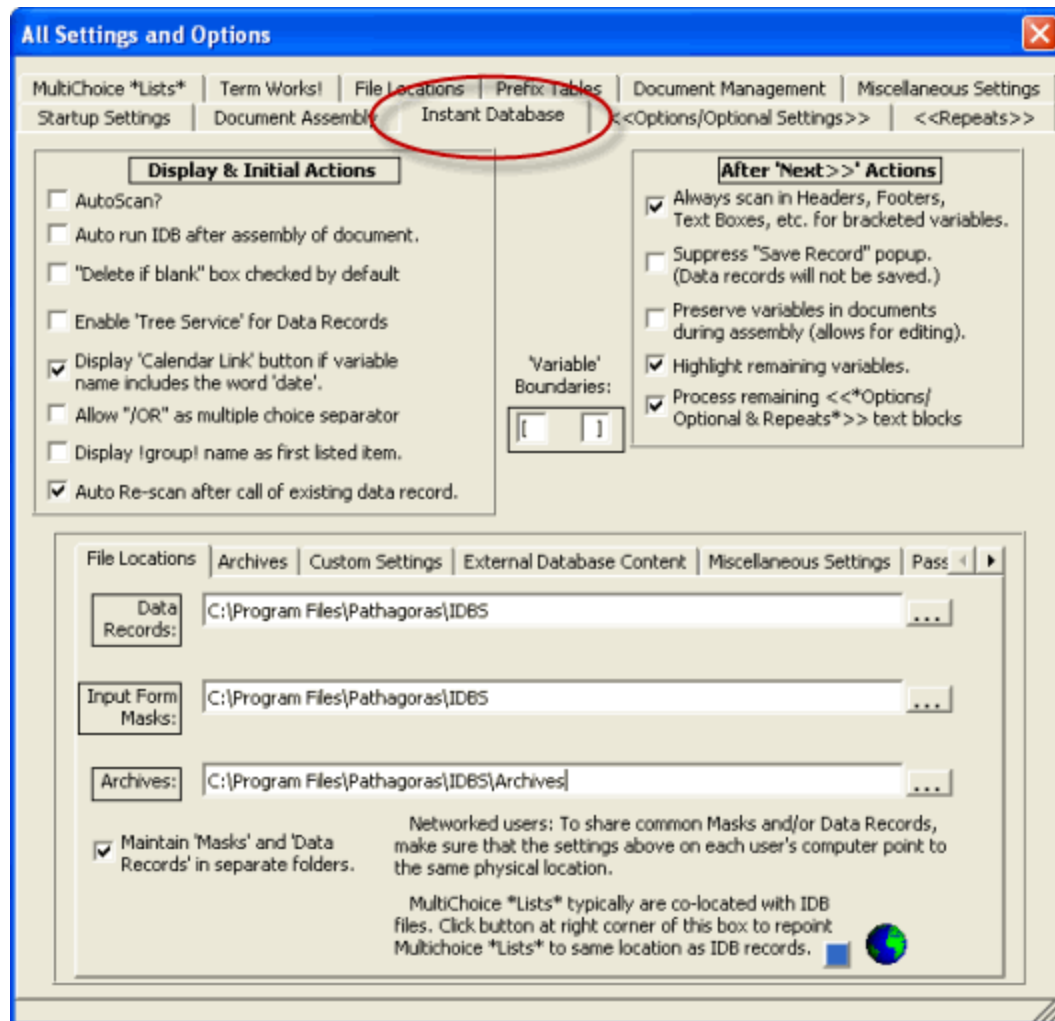
Here is a brief explanation of what each button controls:

- **Alphabetize** entries: Some users who wish to create an Input Form Mask want the variables displayed in alphabetical order. Use this button to achieve that result.
 -  The feature can be used to quickly detect spelling errors in your source documents involving similarly named variables. Let's say you have typed what you intended to be the *same* variable, but used a slightly different 'spelling' of the variable at various locations in the document. (E.g., [ClientName], [Client Name], and [Client_Name].) You can quickly determine whether all variables are properly 'spelled' by scanning the document and 'Alphabetizing' the results. Using the 'Go to selected variable' button, you can quickly locate the 'offending' variable(s) and make the appropriate corrections.
- **Edit Source File:** Should you ever need to access the actual source record (not recommended), click the '**Manually Edit IDB source file.**'
- **Display Source Folder:** Likewise, should you ever want to look at (for editing, or just edification) all of the Instant Database records, click '**Open Folder containing IDB files.**'
- '**Go to selected variable**' is self-explanatory. Just make sure that the cursor is on the proper field. If there is more than one, it will jump to the first instance only.
- **Highlight Variables:** If you want the variables to 'jump out' on the screen, click Highlight each a. The highlighting is 'on-screen' only. It does not print.
- **Display hidden variables:** When Pathagoras is processing documents, it oftentimes must 'record' the values of certain selections so that they are available in later processes. This is especially true re: !groups!. Click "**Recorded (hidden) values**" to see and print these values.
- Clicking '**Print/Transfer/Export**' brings up a menu that will allow you to:
 - **Create DropDown List.** This will place all variables in the left side of the Instant Database screen into a drop down list in the toolbar area of the Word screen. With this list, you can point-and-click those variables into a document under construction. This speeds up the creation of source documents, and avoids spelling errors during the placement of variables. You can refresh this list as you add new variables to your document. See IDB Drop Down List for more on this topic.

- **Insert List** of all variables that are displayed in the current screen. You are given three options. You can insert the list into (1) the current document or (2) into a new document. Either of these documents can easily be made into a client or customer intake sheet. Thirdly, you can insert the variables into your ['Variables' DropDown List](#)¹⁵⁷.
- You can **Export** the current display of variables (and values, if provided) to an Excel spreadsheet. You can edit the data there. The result will be immediately usable by Pathagoras for Word.
- Click the **'AutoScan'** check-box to tell Pathagoras to automatically scan the document for [bracketed variables] whenever the IDB screen is called up. As explained elsewhere, we recommend that users who routinely save and recall IDB records NOT check this box.
- **Enable Drag and Drop:** Easily move a variables into your document . . . drag and drop easy. No more spelling errors. (Just one of several tools to insure consistency of variables.)
- **Replace variables across an entire folder.** Check the box that reads 'Whole Folder Search and Replace'. When you click the Next button on the main screen, Pathagoras will ask you to navigate to the folder which contains the documents in which you want to make the replacements.
- **Edit Current Display:** The stand alone boxes toward the center of the Power Tools strip allow you to manually edit the Instant Database display. You would use this feature if you were creating (or editing) an Input Form Mask. You can move rows, insert blank rows and delete existing ones.
- You can **Import** or **Export** IDB records to share with external users or external programs.
- **Resize Screen:** If the Instant Database screen is not wide enough to show the full name of the variables, or you wish to show more variables than the default '10' you can widen the screen or you can make it taller by clicking the appropriate Screen Sizer buttons. To **lock-in** the resized screen, click the small bar which the Sizer buttons surround. (You can also change the width of the two dropdown boxes, and the default number of records each can show, by making changes in the Miscellaneous Settings. This is found under the "More Tools and Settings.")
- **Change brackets:** If you prefer to use characters other than the default '[' and ']' (square brackets), change them here. You can use any combination of one or two characters except '<<' and '>>'.
- **External Database Link:** Pathagoras can draw data from other data sources. See External Database for more information and setup instructions.
- **Mail Merge:** Pathagoras can create several copies of the same form letter that you want to send to multiple addressees. See Mail Merge for more information.

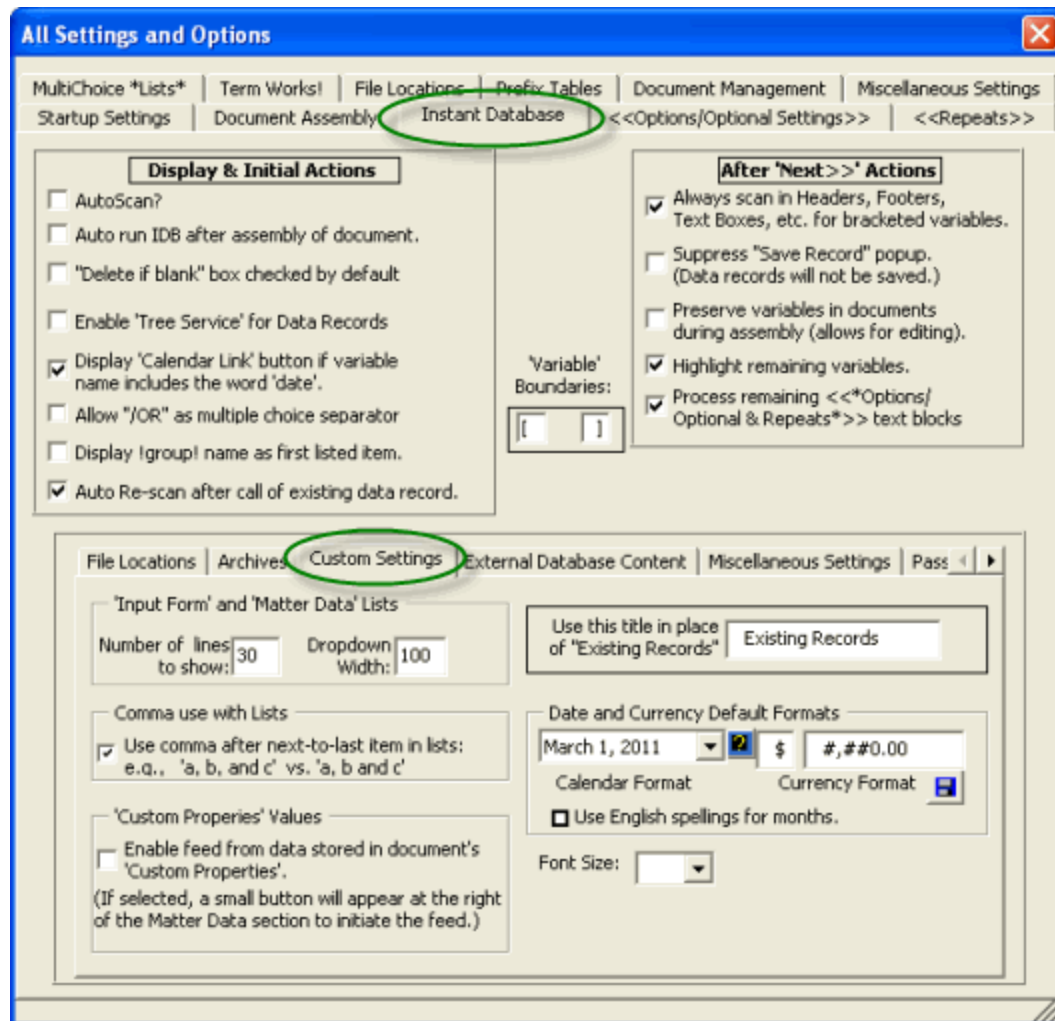
4.7 More Tools and Settings

There are a myriad of settings possible for your IDB module. They are available via the *Instant Database /Instant Database Settings and Options* selection found in the Pathagoras drop down features list. (This same screen can be accessed by clicking 'Power Tools' and then 'More Tools' from the Instant Database screen.)



Instant Database Settings and Tools.

Note the various categories of the display. Your Instant Database is practically infinitely customizable. The primary "Display & Initial Actions" settings are set out at the left of the screen. The "After 'Next>>' Actions" are provided at the right and a wide variety of other settings to customize Pathagoras for your particular office needs are made available in the bottom section of the screen.



Instant Database Tools (Custom Settings)

The various options and switches found throughout this screen allow you to highly personalize the IDB system to your needs. Here are several IDB settings which may be of particular interest:

In the 'Display and Initial Actions' section:

- **AutoScan:** Pathagoras will scan the underlying document for variables without your having to press the Scan button. (If you frequently call in records from you list of existing records, you will probably *not* want this option checked. It is a bit of a time waster for Pathagoras to scan the document only to have you then select a record from the list, which essentially overwrites the scanned for variables. But see the 'ReScan' feature below.)
- **AutoRun IDB:** After the assembly and processing of the document, Pathagoras is automatically activate the Instant Database screen and scan the resulting document for variables without any action on your part.
- **Tree Service:** Instead of storing all data records in a single folder, you can create sub-folders of your records, each sub-folder perhaps representing a major client or customer, or perhaps a major practice area. Individual data records can be stored into (and recalled from) those sub-folders.

- **AutoReScan:** After you call a record from the Existing Records list (upper right corner of IDB screen), Pathagoras will rescan the underlying document for variables that are not in the record. If any are found, it will add them to the bottom. This allows you to continuously update a client/customer's record, adding new variables as they are 'discovered' in new document.



Note: An AutoReScan on long documents with many variables, can take several seconds. If you know that the ReScan will not discover more variables, you can temporarily disable the AutoReScan setting when recalling a record from the list of records in the Instant Database screen. Press Shift-Click on the record name (as opposed to a simple click).

In the 'After Next Actions' section:

- **Highlight Remaining Variables:** If, after running Instant Database against the document, you did not complete all variables, and you chose not to delete uncompleted variables, you can ask Pathagoras to highlight those that remain. This can serve as a visible reminder that there is still work to be done.
- **Process Remaining <<Options/Optional/Repeat>> blocks.** Typically, these 'process' type sections are handled before you would run the Instant Database. However, if one of your variables was a 'document call', you can tell Pathagoras to check for the presence of them after running the Instant Database.

Other Tools and Tabs:

- **Connectors (the Oxford comma):** You can change the presentation of a string of values from "apple, banana and strawberry" to "apple, banana, and strawberry." (Note the 'Oxford' comma after the penultimate element.) This is a personal style choice.
- **'Legal' variable length:** Pathagoras 'gives up' on identifying text as a variable when it exceeds a certain length. 40 is the default 'max'. If you typically to provide 'explanatory text' within your documents using square brackets you use for your variables, you can set Pathagoras 'give up the search' length to exclude that explanatory text. (However, the better practice is to change the boundary markers of either variables or of the explanatory text.)
- **Variable boundaries:** You can change the characters used to denote variables. Square brackets are the default, but you can use anything so long as it does not conflict with the 'simplified options/optional text characters, which are show as well on this screen.

Click the various tabs along the top of the screen and other important information and data regarding the following topics will be displayed.

- **File Locations:** See where your IDB records are being stored. Or easily change the location where they are being stored. Some offices wish to share a common folder for all IDB records. In such case, point all users to the same location. Other offices wish to maintain separate files. In such case, make sure that all users point to separate location.
- **Archives:** If your IDB Data Records are becoming too numerous, save them out to an archive folder. Quickly restore one or more record if needed.

- **Custom Settings:** Under this tab, you can change the width and depth (number of records displayed at one time) in the two drop down lists on the page. You can direct Pathagoras to immediately print your document to the default or to a selected printer. (This latter feature can be viewed both as a convenience feature and as a security feature. By requiring that the assembled document be printed after assembly, there is some assurance that it has not been modified by the end-user or others.)

See Also: [Multi-Choice Lists](#)¹¹⁴

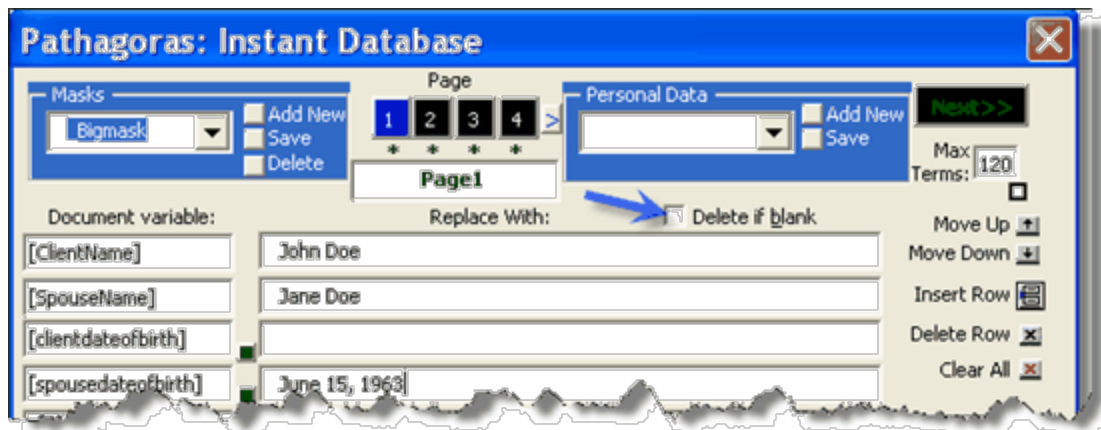
Linking to External Databases

Archiving IDB Records

4.8 Other IDB Functions, Features and Tools

There are a few more features of the IDB system of which you should be aware that have not been discussed elsewhere:

- **Delete if Blank:** By default, any variable for which you do not provide a value (i.e., is 'blank') remains intact in the document when you press Next>>. The presumption is that you will wish to complete blank variables orking with the replacement text you current have and complete the remaining variables at a later time. But if you wish to delete blank variables its value is 'blank' (a 'classic' one is [Client Address 2], check the 'Delete if blank' box. ('Delete if Blank' will also remove any trailing commas and spaces and blank lines so no empty space remains.)



Uncheck 'Delete if blank' box to retain any variables for which data is not currently available.

You can make this check-box 'sticky' (i.e., always checked) with a 'shift-click'. Undo the 'always checked' condition with another 'shift-click.'

- **Copy variable, brackets and all:** Let's say that you want to preserve a specific variable, but want to delete all other 'blank' variables (the bullet immediately above). To accomplish this seemingly contradictory feat, just copy the bracketed variable onto itself. The value is not 'blank' so it won't be deleted, but the variable also remains in the document for later completion. 'Shift-Double click' on the variable (left column) accomplishes this.
- **Don't re-scan:** By default, Pathagoras re-scans the underlying document for additional variables after you have recalled an Existing Record. If the underlying document contains many variables, and these variables are already contained in the Existing Record, this re-scan is

consuming several seconds of your time for no good purpose. While the rescan is typically just a few second, not long, it is still 'time spent' if no new variables exist. So if you have a 'mature' record (one that you know contains all possible variables, you can skip this re-scan by clicking the box in the lower right corner of the 'Existing Records' section.

4.9 Double-Click Tools

Double-clicking in certain fields in the Instant Database screen will trigger Pathagoras to perform designated functions. The state or a value within the field determines the double-click action.

Right-side double-click possibilities:

(a) For 2, 3 and 4 below, a key word must exist in any form with the variable name so Pathagoras knows what kind of data you are working with. So if the keyword is SSN, it 'exists' only if SSN is anywhere in the variable: [SSN] or [Client SSN] or [SSN of Spouse], etc.

(b) Again, for 2, 3 and 4 below, the right (value) side must contain at least an 'acceptable value'. (For example, if you want to convert a telephone number that contains letter, or an insufficient number of digits, Pathagoras won't know what to do with it.)

1. Cycle the case of the text.

This is the 'default' action, triggered if the keywords described below don't exist.

The double-click 'cycle' is from 'ALLCAPS' to 'First Letter Caps' to 'lower case' back to 'ALLCAPS'

2. Cycle Date formats if 'date' keyword exists:

(a.) A 'date' keyword (in the left column) is one that contains 'Date', 'DOM', 'DOS', 'DOB', 'Birthdate', and 'Today'.

(b.) An acceptable beginning value is anything that appears as a date.

Return: Pathagoras will cycle through a selection of 6 of date options. Keep double clicking to see them all.

Note: If you want to preserve the date as typed in the value box, but need different formatting for that same date in other parts of the document, use the date formatting arguments as discussed [in this section](#)⁸⁴ of the Manual.

3. Convert Social Security Numbers to proper format if 'SSN' keyword exists:

(a) The key word is 'SSN' or 'Social Security' in any fashion within the variable name.

(b.) An acceptable value at the right must be a string of digits, 9 digits long.

Return: The 'return' will be in the format XXX-XX-XXXX. So, 222334444 becomes 222-33-4444

4. Convert Telephone Numbers to proper format if 'phone' keyword exists:

(a) The key word is "phone" or "fax" in any fashion within the variable name (left column).

(b) An acceptable value is any 10 numbers, with or without parentheses and hyphen. (If you want to convert a telephone number that contains a letter, or an insufficient number of digits at the right, Pathagoras won't know what to do with it.)

Return: The 'return' will be the default telephone format you selected in the Instant Database Settings and Options screenage. So, 5555551212 becomes (555) 555-1212

Other *right-side* double-click possibilities:

Equivalencies:

If the value at the right is an equivalency function, (e.g., "[Client Name]"), and if the variable doesn't yet exist in the left column, Pathagoras can create variable for you. Just double-click in the text box at the right showing the equivalency. The variable will appear immediately below the equivalency. (If the variable exists, Pathagoras will tell you so, and also tell you the page if the IDB screen on which it resides.) [See this page in the Manual.](#) ⁸⁵

Nested variables:

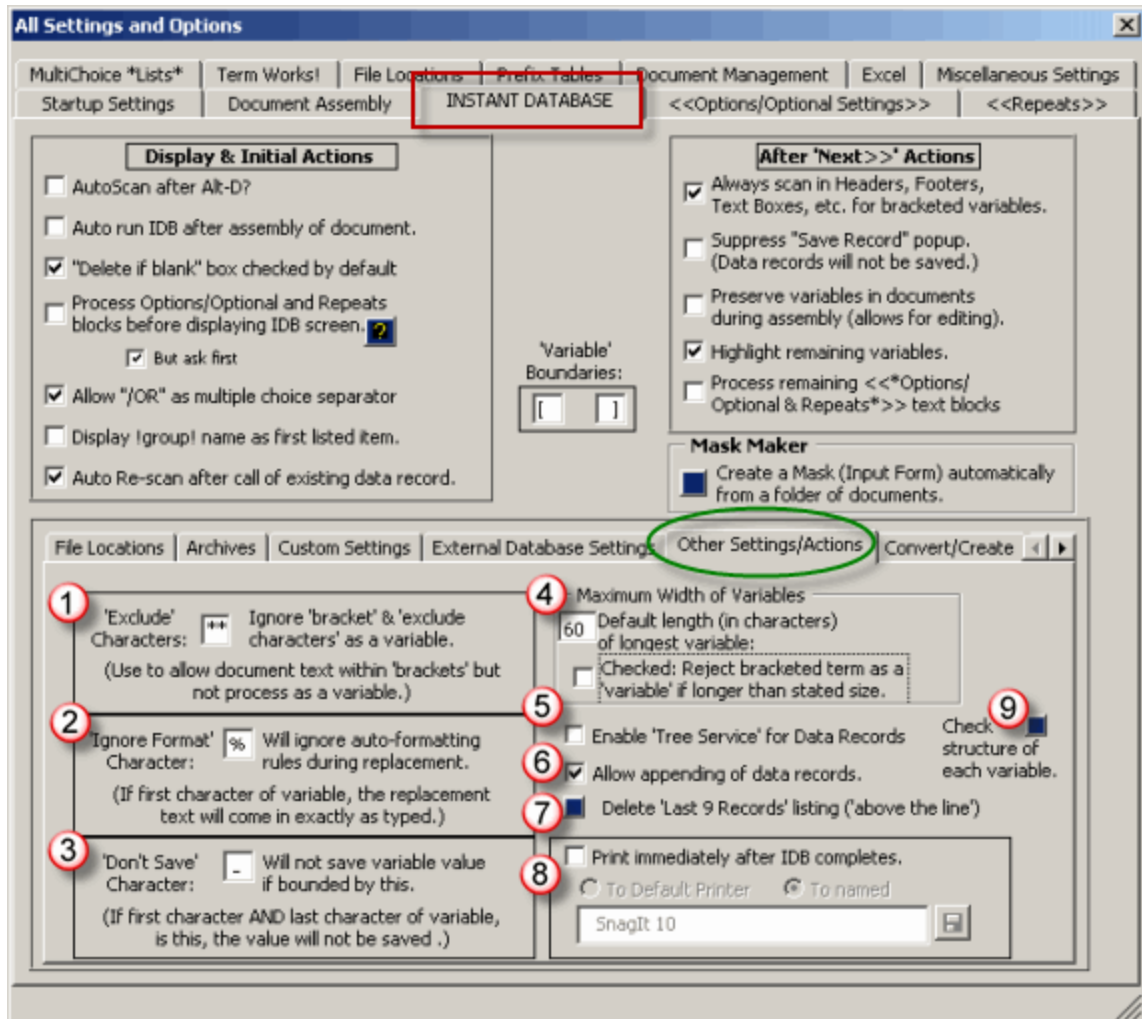
If your selected multiple choice option itself contains one or more variables nested within it, AND the nested variable(s) do not yet exist in the left column, Pathagoras can create the variable(s) for you. Just double-click in the text box at the right showing the nested variable. The variable(s) will appear immediately below the original variable. (If a variable exists, Pathagoras will tell you so, and also tell you the page if the IDB screen on which it resides.) See this page in the Manual.

Left-side (variable side) double clicks:

When you double-click on a variable, the name of the variable transfers to the right side with the brackets stripped away. This is quick way of transferring over a default value (if the default value happens to be the name of the variable). A shift-double click will retain the brackets. (This latter tool might be used to preserve otherwise blank values if you have 'Delete Blank Variables' selected. A variable that replaces itself is not 'blank'.)

4.10 Still More IDB Settings

As stated in the previous screen, the settings for Instant Database processing is practically infinite. Here are even more that are available behind the "Other Settings/Actions" screen:



1. **'Exclude' characters:** You can tell Pathagoras to ignore as a variable any bracketed text that begins with a designated character set.
2. **'Ignore Format' character:** You can tell Pathagoras to replace a variable exactly as typed. (Use when you don't want Pathagoras to convert a variable to ALL CAPS or all lower case when the variable within the document text is written as such.)
3. **'Don't Save' character:** A variable written [-Date-] or [-Name of Document-] will be replaced with the designated replacement, but the replacement value will not be saved with the other terms in the record. Look at this as a 'temporary' variable.
4. **Maximum width of variables:** Designate the default maximum width which Pathagoras will 'automatically see' a bracketed set of word as a variable. If longer, Pathagoras will Ask if you want to process it as a variable. Check the box and any bracketed text longer than 60 (or whatever) will be ignored.
5. **'Tree Service':** Check this box to direct Pathagoras to store your data records in sub-folders. Designed for use in offices where there are 'primary' clients (e.g., a creditor), and many files associated with that client (customers against whom collection efforts are being pursued.).

6. **'Allow appending of Data Records':** You can select one record, append a set of related data to that record and end up with an augmented 'third' data set. An advanced function, but it allows you to have a 'header' record of standard client data, and then any number of secondary records for specific matters. When used, the augmented record can be saved, but it is probably best if it is not. (No need to re-save -- and possibly have to edit -- duplicated information.)

7. **Delete 'Last 9 records; List:** Delete the 'above the line' listing of records, which typically show the last 9 recalled data records. (Only the listing is deleted, not the records.)

8. **'Print':** If you want to immediately proceed to printing after the Instant Database record is processed, check this box. Not a recommended setting for most offices.

4.11 Multiple Choice Selector

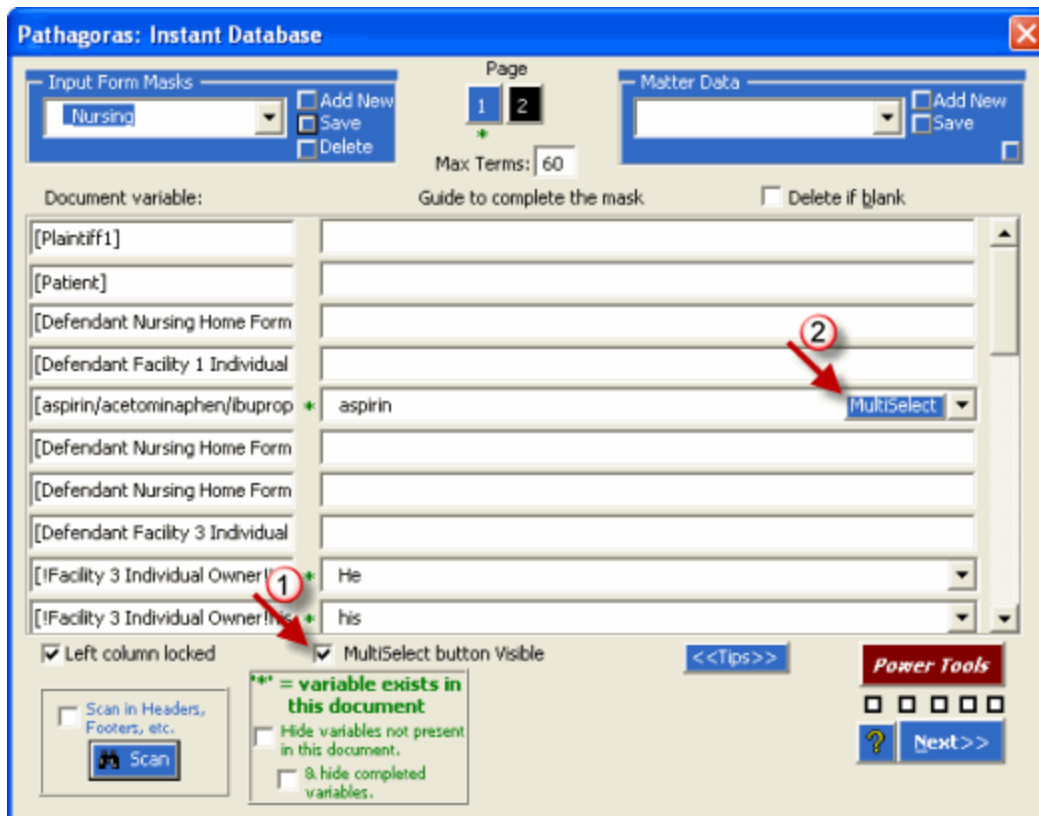
"[Chocolate/Vanilla/Strawberry]" is a multiple choice item. So is [*States*]. When a multiple choice list is encountered by Pathagoras, it parses the various options and presents those choices in a dropdown list on the face of the Instant Database screen. Typically, you would drop down the list, select a single value from the list, and move on. But what if the answer requires multiple selections from the list. Can that be done? Of course!

When you click inside of a drop down list containing multiple choices, a MultiSelect button will appear inside the row at the right. (NOTE: The first time you try this, you may have to tell Pathagoras to display the MultiSelect button. Look for the check-box that appears just beneath the last 'replacement entry' location titled "MultiSelect Button Visible". (It will appear only when you are within a multiple choice dropdown.) Click it.

The screenshot displays the 'Pathagoras: Instant Database' application window. At the top, there are sections for 'Input Form Masks' (with a dropdown set to 'Nursing' and buttons for 'Add New', 'Save', and 'Delete') and 'Matter Data' (with a dropdown and 'Add New', 'Save' buttons). Below these is a 'Page' indicator showing '1' of '2' and a 'Max Terms: 60' setting. The main area is titled 'Document variable:' and contains a list of variables on the left and their corresponding values on the right. The variables listed are: [Plaintiff1], [Patient], [Defendant Nursing Home Form], [Defendant Facility 1 Individual], [aspirin/acetaminaphen/ibuprofen] (with a green asterisk), [Defendant Nursing Home Form], [Defendant Nursing Home Form], [Defendant Facility 3 Individual], [!Facility 3 Individual Owner!H] (with a green asterisk), and [!Facility 3 Individual Owner!H] (with a green asterisk). The values shown are: [aspirin/acetaminaphen/ibuprofen] is 'aspirin' (indicated by a red circle 1), [!Facility 3 Individual Owner!H] is 'He' (indicated by a red circle 2), and the next [!Facility 3 Individual Owner!H] is 'his'. At the bottom left, there is a 'Left column locked' checkbox and a 'Scan' button. At the bottom center, there is a 'MultiSelect button Visible' checkbox and a text box explaining that '** = variable exists in this document' and that it can hide variables not present or hide completed variables. At the bottom right, there is a '<<Tips>>' button and a 'Power Tools' section with a 'Next>>' button.

If you desire to choose more than one item from a list, enter the text box containing the list and, if it is not already checked, check the 'MultiSelect button Visible' box.

A new button will appear within the boundary of the current list (and any other list you may enter) called MultiSelect. Click it.



When MultiSelect box is checked, a MultiSelect button appears.

A new section of the screen opens up containing a listing of all of the choices. Make appropriate choices from the list. (Regular Windows list-choosing techniques -- shift-click and ctrl-click -- work here.) Next, if desired, choose a method by which the terms will be 'separated' and 'connected' (by commas, followed by 'and' or 'or', etc. or 'No connectors.') Press the Transfer button and the properly formatted text will be transferred into the Replacement text area.

Select choices (Ctrl- or Shift- click techniques work).

Click a separator, and then click Transfer. That's it!

In the example above, when <Transfer> is clicked, the replacement text will become "aspirin, ibuprophen and naproxin".

You can change the 'default' separator via the 'All Settings' | 'Miscellaneous Settings' screen. See Separators and Connectors for more information.

4.12 Variables, Length of

A variable can be any length.

That said, the best variables are short. Short variables are easier for the user to see and process when encountered in a document.

When a variable exceeds a certain length, it becomes unwieldy. There is no definitive size when this occurs, but Pathagoras suggests 40 characters as the 'max.' ('60' is set as the default maximum variable length when you install Pathagoras.)

Can a variable be 'too short'? Yes. When the replacement value is not clearly suggested by the variable, it is too short. The variable '[NOC]' (standing for, perhaps, Name of Client) is probably too short. Why? Few people (especially new staff just learning your systems) would know what that abbreviation means. ('[DOB]' may work because it somewhat has a universal acceptance as the abbreviation for Date of Birth.)

Similarly, "[Name]" is probably too short for most situations. Who's name is being called for? On the other hand, "[Name of Client]" is perfect. So would be "[Name of Our Client]", "[Client Name]", "[ClientName]", and a myriad of other choice. "[Name of our client as indicated on the first line of

the Client Intake Sheet]" is 'legal', but too long for 'best practice.' (What about [Client's Name]? We also suggest no quotation marks, apostrophes, dashes and other punctuation.

4.13 Other Uses for IDB

Simple Find and Replace using IDB

Display the Instant Database screen (<Alt-D>).

Type into the left-hand column the text that you want to change, and in the right-hand column the replacement text. Press 'Next'. That's it.



The Instant Database module is really nothing more than an elaborate “Find & Replace” tool. However, unlike Find & Replace where replacements can only be made one at a time, Pathagoras can make replacements 10, 20, 100, etc., at a time.

- In the right column, type the word or phrase that will replace what you have typed at the left.
- When all “find” and “replace with” terms are listed, press the Next>> button.
- The questions “Do you want to save the data as a personal record?” will appear. If you plan to reuse this 'find and replace' grouping again for other documents, say "Yes." Otherwise, answer “No”.
- Very quickly, Pathagoras will locate each instance of the “find” text and replace it with the corresponding “replace with” text.

Make a ‘Sex Change’ Mask

This is not what you think. This is simply a different application of Pathagoras' Instant Database by which you can reverse ‘he’ and ‘she’ and ‘him’ and ‘her’ throughout an entire document. The following ‘pattern’ of left and right column entries will take care of the issue.

This example also illustrates how the IDB mask can be programmed to handle simple housekeeping chores when you do not want to create an elaborate mask. (Notes: there must be spaces on both ends of each word you wish to change. Doing so prevents the possibility of the change from 'he' to 'she' from affecting words that contain ‘he’. So ‘the’ (which contains ‘he’) does not become ‘tshe’. The ‘xxx’ suffix serves to make sure that the order of replacements are maintained.)

Variable	Replacement Text
she	shexxx
her	[himxxx/hisxxx]
hers	hisxxx
female	malexxx
woman	manxxx

(add any other female sex based term)

he	she
him	her
his	[her/hers]
male	female
man	woman
hexxx	he
hisxxx	his
malexxx	male
manxxx	man

➔ This is but one of any number of ‘transformational’ masks that you could create. (Something like the above is not necessary with a typical ‘Pathagorized’ document. [Groupings](#)²¹⁴, discussed on an earlier page can easily take care of gender based pronouns. But the above, and others like it, are ‘cool tricks’ when it comes to modifying documents that have not yet been Pathagorized.)

4.14 Screen Shots Depicting Screen Features

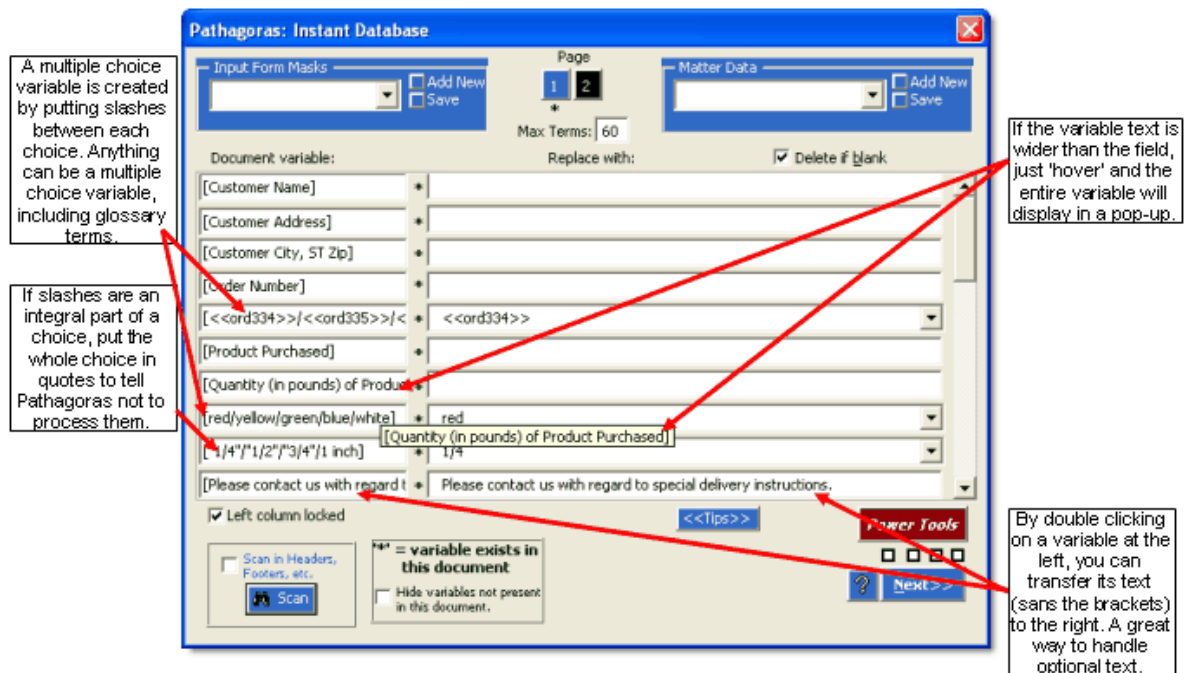


Figure 1. Results from a <Scan>.

All variable displayed in the left column are plain text, [bracketed variables] in the underlying document.

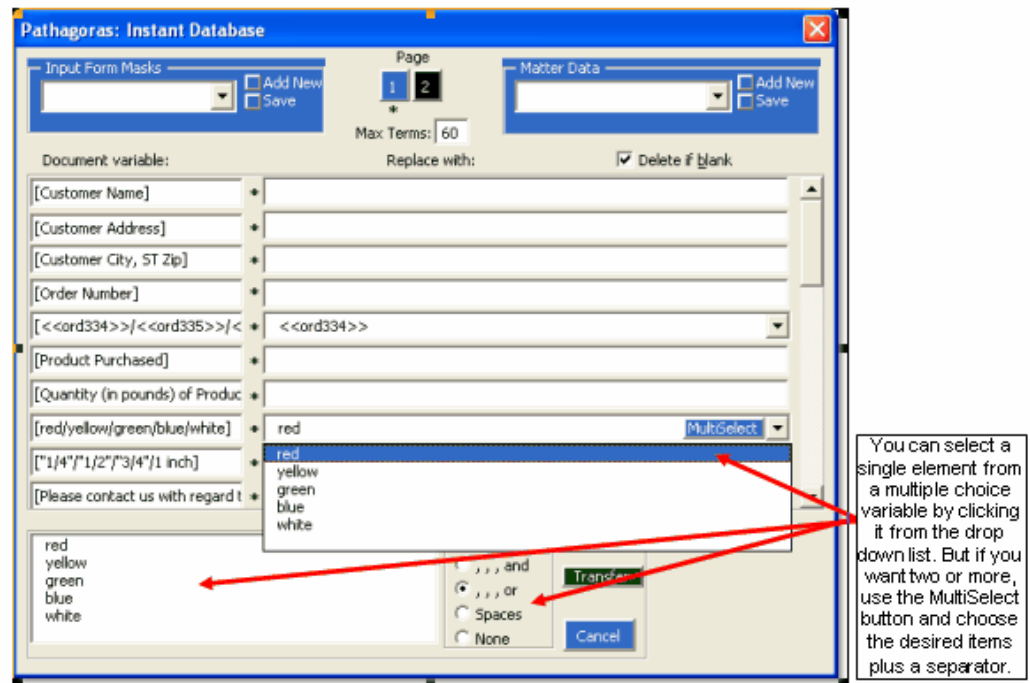
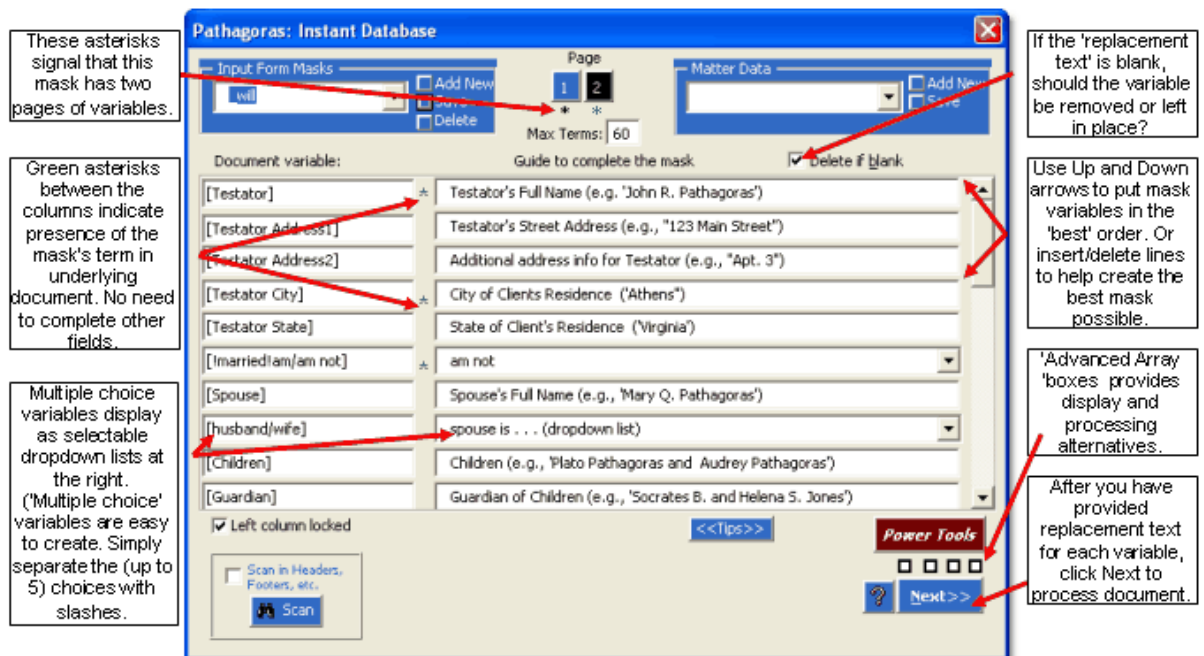
Figure 2. IDB: [Multiple Choice Selector](#) ⁶⁵

Figure 3. Instant Database Mask w/ Completion Tips (ready to be saved as a mask).

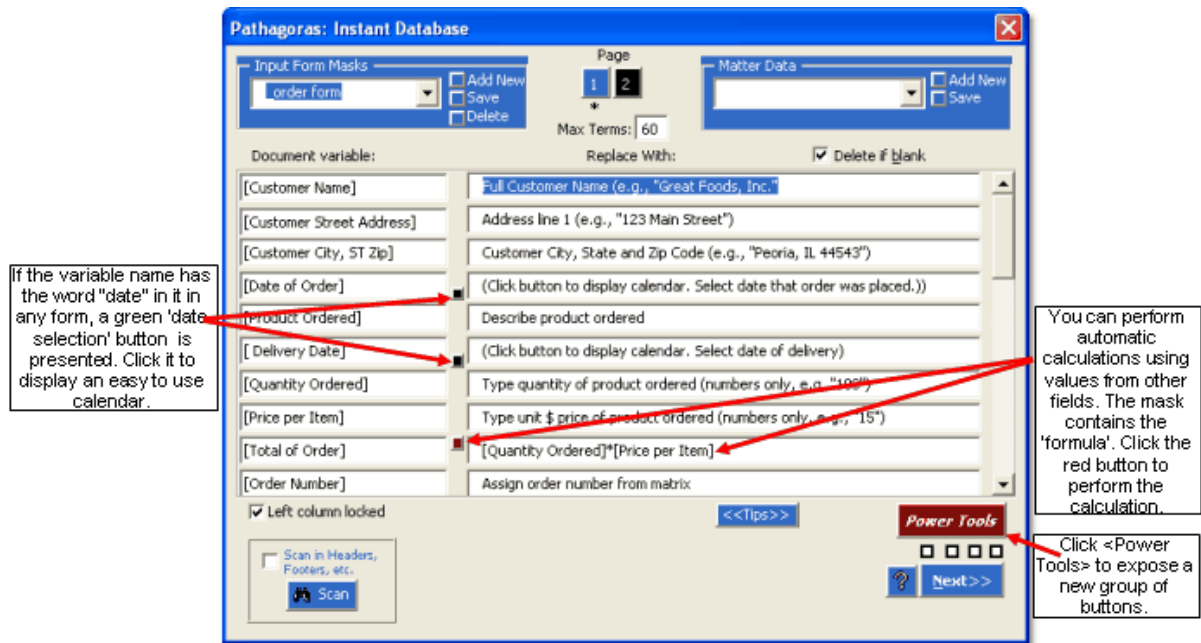


Figure 4. More IDB features, including [Date Picker](#)^[74] and [Calculator](#)^[76] buttons on display.

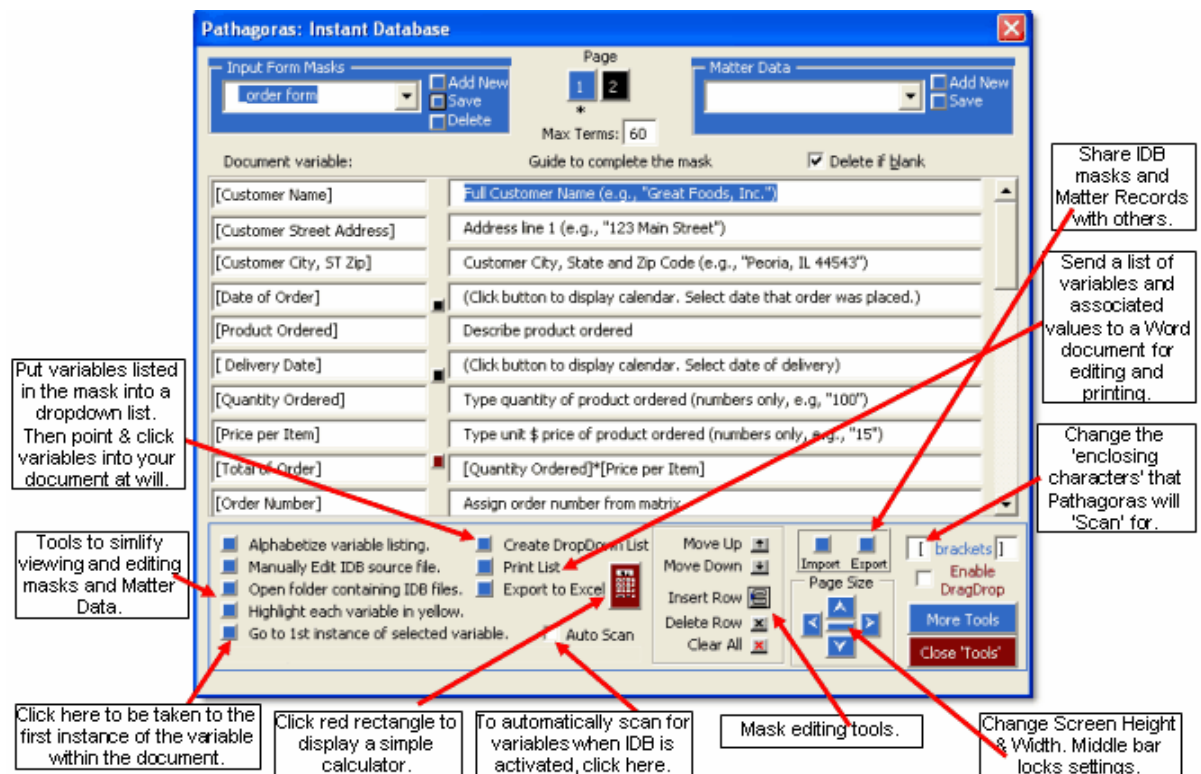


Figure 5. The additional IDB features show under ['Power Tools'](#)^[59].

The Pathagoras System

Instant Database Functions

Part



V

5 Instant Database Functions

Pathagoras offers a wide variety of tools, formatting options and shortcuts which allow you to easily build, augment and conform your documents to designated office standards. They include:

- [date formatting](#)⁸⁴,
- [date math](#)⁷⁴,
- navigation tools which allow you to set a 'variable' to a folder. Use, for example, when you want to insert an entire document (such as a property description) in place of the variable.
- an [equivalency function](#)⁸⁵ that allows the pairing of a single 'actor' with multiple roles.
- [spell out functions](#)⁹⁶: dollar and other number values; percents; fractions.
- [concatenizing](#)⁹⁴ (building new variables from existing ones).
- exporting your Instant Database data to Excel.

These, and other, functions are discussed in the following sections of this Manual.

5.1 Calendar and Date Math Features

Instant Database: Calendar and Date Math features

If Pathagoras detects a variable which includes the word “date” in any form, Pathagoras will display a small green button that will appear between the two columns on the same row as the variable being processed.

When clicked (optional), Pathagoras will present a calendar from which you can

- (1) pick a date and
- (2) perform date math (add a certain number of days, weeks, months or years to a starting date).

Examples of "date" words that will trigger the display of the calendar button: 'Date', 'Birthdate', 'Date of Sale', 'Intimidate' (sorry).

Screen Shots:

Pathagoras: Instant Database

Input Form Masks: Real Estate (dropdown) Add New Save Delete

Page: 1 2 Max Terms: 60

Matter Data (dropdown) Add New Save

Document variable:

[Landlord]	Name of our client
[Tenant]	Name of tenant
[Address of Property]	full street address
[Inception Date]	beginning date of lease
[Termination Date]	end date of lease
[Contract Date]	date lease signed
[monthly rent]	monthly rental amount
[deposit]	deposit (typically)

Left column locked

Scan in Headers, Footers, etc. (checked) Scan

Hide variables not present in this document. (unchecked)

Show Tips

Power Tools

? Next>>

[Inception Date]

Today September 2008

Sun	Mon	Tue	We	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

September 28, 2008

OK

Whenever a variable contains the word 'date' (in any form), a small green button between the columns will display. Click it to reveal the calendar.

Select a date. (Insert the selected date into the Instant Database screen by 'double clicking' or pressing the okay button.)

To perform date math, click the Calculator to the right of the calendar overlay.

The screenshot shows the 'Pathagoras: Instant Database' application window. A modal dialog titled '[Inception Date]' is open, displaying a calendar for September 2008. The date 'September 28, 2008' is selected. To the right of the calendar is a 'Date Math' section with a text input showing 'September 28, 2008', a numeric input set to '1', and radio buttons for 'Days', 'Months', 'Weeks', and 'Years'. Below these is a 'Calculate' button and an 'Expiry Date' checkbox. The result of the calculation, 'October 27, 2008', is shown in a text box, along with a 'Transfer to IDB' button. The background application window shows a form with fields for document variables: [Landlord], [Tenant], [Address of Property], [Inception Date], [Termination Date], [Contract Date], [monthly rent], and [deposit]. It also includes a 'Page' indicator (1/2), 'Input Form Masks' (Real Estate), and 'Matter Data' sections.

Perform date math by adding or subtracting days, weeks, months or years.

The option 'Expiry date' is one period minus one day to reflect (as do most leases) the last day of the term.

The check-box 'End date must be workday' insures that discovery and other important dates don't fall on weekend.


Notes:

- If you have created a mask, you can perform simple date-addition directly on face of the Instant Database screen. See the [next section](#)⁷⁶ for the steps to set up the formula.
- The format for the return value of the date (e.g., "01/01/27", "January 1, 2027" etc.) can be set via the Instant Database Settings Screen. [See this link](#)⁵⁷.
- You can insert dates in multiple formats from a single entry. See [this page](#)⁸⁴ for more info.

5.2 Math and Date Math

Simple math can be performed using any existing Instant Database variable which has been assigned a numeric value. Create a formulas and assign its result to another variables.

Pathagoras can perform addition, subtraction, multiplication or division on up to four operands.

 This feature presupposes that you will be preparing an Instant Database mask to house the formulas. If you want to place the formula in-line with the text of the source document, [see this page](#).⁸¹

Example: Let's say that you have a document containing, among others, four variables that you want to use to calculate a total price. Perhaps they are called [Unit Price], [Quantity], [Discount] and [Total Price]. These variables are contained in a mask designed for this and similar 'price quoting' documents.

The formula in the mask adjacent to [Total Price] might look like one of these:

[Unit Price] * [Quantity] - [Discount] (*if the variable discount is a flat amount*) OR

[Unit Price] * [Quantity] * [Discount] (*if the variable 'discount' is a percentage*) OR

[Unit Price] * [150] * .25 (*you can use 'real' numbers if you wish, with or without brackets*).

[Unit Price] * 150 * .25

In actual operation, you would assemble the document for the customer and display the appropriate IDB mask. You would complete the 'personal variables' (name, address, etc.) and insert the appropriate values for each of the first three variables (unit price, quantity and discount) which feed the formula.

When Pathagoras detects a formula (the presence of math symbols is the give away) in the personal data side of the IDB screen, it will display a small red button between the two columns. (Similar in size to the 'date picker' button.) Click the red button and the formula is processed.

If any figure that is part of the formula changes, the [Total Price] can be recalculated by two successive clicks of the calculator button. The first click restores the formula. The second click recalculates using the new values.

You can have a total of 4 "operands" (and therefore up to 3 operators).

Here is another example:

A mask called Order Form was created for quoting prices. It contains a simple formula to calculate the value for the variable called [Total of Order].

When the mask is completed, and after values have been assigned to the base variables upon which the formula depends, click the red Calculate button. If a base figure changes, click the button twice to recalculate the value.

The allowable 'math operators' are:	
Addition: "+"	Subtraction: "-"
Multiplication: "*" or "x"	Division: "/"

In the examples on this page, you will see an introductory '=' sign in some and not in others. When all variables appear above the formula, the '=' is not required. The better practice would be to include the '=' in all cases.

DATE MATH:

Date math (addition) can be performed from the Instant Database screen following the same setup styles as used with the above 'plain' math examples. The formula is written like this:

$$=[\text{Base Date}]+[4 \text{ months}]$$

where [Base Date] is any date variable in the mask (the variable must contain the word 'date', but it can be in any form). The time interval addend can be any number, followed by a space, and concluded with either 'days', 'weeks', 'months' or 'years' (or the singular).

Example:

The screenshot shows the 'Pathagoras: Instant Database' window. It has a 'Formula' tab selected. On the left, under 'Input Form Masks', there's a dropdown menu showing 'Order Form' and buttons for 'Add New', 'Save', and 'Delete'. In the center, there's a 'Page' indicator showing '1' and '2', and a 'Max Terms: 60' label. On the right, under 'Matter Data', there's a dropdown menu and buttons for 'Add New' and 'Save'. Below these, there's a 'Document variable:' label and a 'Replace With:' label with a checked 'Delete if blank' option. The main area is a table with two columns: 'Document variable:' and 'Replace With:'. The table contains the following data:

Document variable:	Replace With:
[Customer Name]	Great Foods, Inc.
[Customer Street Address]	123 Main Street
[Customer Address 2]	Suite 45
[Customer City, State Zip]	Jacksonville, FL
[Product Ordered]	Widgets
[Quantity Ordered]	25
[Price per Item]	47
[Total of Order]	1,175
[Date of Order]	March 22, 2010
[Delivery Date]	[Date of Order] + [3 weeks]

Red arrows and numbers indicate the steps: (1) points to the green 'Calendar' button next to the 'Date of Order' row, and (2) points to the red 'Calculate' button next to the 'Delivery Date' row.

- (1). Set the 'base' date by clicking the green 'Calendar' button. (You can also manually type in the date).
- (2). Click the red 'Calculate' button on the target line. You should see the result as shown below:

The screenshot shows the 'Pathagoras: Instant Database' window. It has a blue title bar and a menu bar with 'Input Form Masks' and 'Matter Data'. Below the menu bar, there are buttons for 'Add New', 'Save', and 'Delete'. The 'Input Form Masks' section shows a dropdown menu with 'Order Form' selected. The 'Matter Data' section shows a dropdown menu with a blank space. Below these, there are 'Page' buttons (1 and 2) and a 'Max Terms' field set to 60. The main area is a table with two columns: 'Document variable:' and 'Replace With:'. The table contains the following data:

Document variable:	Replace With:
[Customer Name]	Great Foods, Inc.
[Customer Street Address]	123 Main Street
[Customer Address 2]	Suite 45
[Customer City, State Zip]	Jacksonville, FL
[Product Ordered]	Widgets
[Quantity Ordered]	25
[Price per Item]	47
[Total of Order]	1,175
[Date of Order]	March 22, 2010
[Delivery Date]	April 12, 2010

The 'Delivery Date' field is circled in red. There is a checkbox labeled 'Delete if blank' which is checked.

If you change the 'base date,' click the calculate button twice.
 The first click will restore the formula (in case you want to modify it).
 The second click will perform the calculation on the new assumptions.

Date "difference" math (how many days, weeks months or years have passed between two known dates) can be performed from the Instant Database screen, again using a mask to save the formulas for repeated use.

The same setup styles as shown above is used. Sample variables, a sample formula and screen shots, are provided below:

[Base date]

[End date]

[Span, in Months] =[Begin Date]-[End date] (the formula beginning with '=' is placed in the right column)

[Age in Years] =[Begin Date]-[End date] (the formula beginning with '=' is placed in the right column)

Other acceptable calls:

[Age in Years and Months] will return e.g., 10 years and 9 months

or

[Age in YM] (same as above)

The screenshot shows the 'Pathagoras: Instant Database' window. The 'Input Form Masks' section has a dropdown set to 'aaaTestDate' with buttons for 'Add New', 'Save', and 'Delete'. The 'Page' indicator shows '1' of '2'. The 'Existing Records' section has a dropdown and buttons for 'Add/Move' and 'Save'. The 'Document variable:' section is empty. The 'Guide to complete the mask' section has a checkbox for 'Delete if blank' which is checked. The main table has four rows with the following values: '[Base date]' is 'January 1, 1950', '[End date]' is 'March 1, 2013', '[Span Months]' is '= [Base Date]-[End date]', and '[Age in Years]' is '= [Base Date]-[End date]'. The bottom of the window has a 'Scan' button, a 'Scan in Headers, Footers, etc.' checkbox, and a 'Power Tools' section with a '?' icon and a 'Next >>' button.

The screenshot shows the same 'Pathagoras: Instant Database' window, but with calculated results. The values in the table are: '[Base date]' is 'January 1, 1950', '[End date]' is 'March 1, 2013', '[Span Months]' is '758', and '[Age in Years]' is '63'. The '758' and '63' are circled in green and blue respectively. The 'Scan' button and 'Power Tools' section are still visible at the bottom.

There are 758 months between the two dates, which converts to 63 (whole) years. The numeric result of number of days, weeks, months or years is controlled by the existence of the appropriate term "day(s)", "week(s)", "month(s)" or "year(s)" contained in the variable at the left.

5.3 In-line Math and Date Math

As the previous sections indicate, Pathagoras has the ability to perform simple mathematical calculations.

Your formulas can be stored in Input Form Masks (as previously shown). However, you can also insert math formulas 'in-line' with the source text. The requisites are as follows:

- The maximum number of operands is 4 (and by extension, the maximum number of actual calculations is 3)
- Only addition, subtraction, multiplication and division calculations are possible.
- If a formula contains variables, the precedent values of those variables must appear 'above' the formula.
- If you are performing mixed date and 'standard' math calculations, the date calculations must come first. (see below for an example of a 'mixed' calculation). Processing proceeds from left to right

A precise structure is required (examples shown below), but it's really not that hard:

The 'formula' is:

1. Square brackets surrounding the entire variable term;
2. The call-to-action "#Math#" (no quotes) placed immediately after the opening bracket;
3. The variable name;
4. An equal sign to indicate the equivalency;
5. The formula, which is simply a reference to the existing variable, and the math you wish to perform. Note that there will always be a nesting of the variables, and the closing of the variable will very frequently be a double ']]'.

Here are some examples. The first elements are simply plain variables. They need to be completed in order for any in-line math to be performed. There is no order requirement EXCEPT that all references must be on the same Instant Database page as the formula.

NOTE: The 'in-line' formulas discussed on this page are what appear 'in-line' in your text. They are just formulas that you would type into a Mask. When you invoke 'Scan', Pathagoras identifies the in-line formula and parses out the 'variable name' from the formula and places the components into the proper column on the Instant Database screen. The resulting entries are the same as if you manually typed a variable and a formula per the guidance provided in the preceding section.

Simple Math ('total price', 'half unit price' and 'half total price' are the target variables):

[Quantity]

[Price]

[#Math#total price=[Quantity]x[Price]]

[#Math#half unit price=[Price]/2]

[#Math#half total price=[Quantity]x[Price]/2]

(Note: multiplication may be indicated by 'x', or '*'. However a capital 'X' will **not** be seen as an operand.)

Date Math:

This contract will begin on [start date].

It will automatically renew on [#Math#termination date=[start date] + [3 months]], and every 3 months thereafter.

If you want to return the 'expiry' date (the day before the 'typical' return date, use this setup:

This contract will begin on [start date].

It will expire at midnight on [#Math#termination date=[start date] + [3 months] - [1 day]].

Here is an example of a 'mixed' calculation, first calculating a 'difference' in days between two dates, and then multiplying that difference by a penalty per day figure.

We requested the supporting documents on [Date Docs Requested] but to date have not received them.

Federal regulations require production of the documents within 30 days, and allow a penalty for each day the documents are not produced after that. Therefore, the penalty for the delay is <\$>[#Math#Penalty=[Today] - [Date Docs Requested] - [30 days] * 110] as of today's date.

A calculated date in the future, expressed in the 'extended' the Xth date of Month, Year' format:

The notice was published on [Date of Notice].

If you do not respond by [#Math#Date Sale(ext)=[Date of Notice]+[21 days]], we shall proceed with the foreclosure sale

Age Math ('Age of H' is the target variable):


Husband was born on [Birthday of H].


He is [#Math#Age of H in years=[Today]-[Birthday of H]] years old.


Susie was born on [Susie Birthday]

Susie is [#Math#Age of Susie in years and months=[Today]-[Susie birthday]] old.

(Note: When the variable [Today] is in the formula, you need not pre-set its value. Pathagoras will assign the current date to the variable.

 One might say that the above formulas are pretty complex to set up. While that may be true, keep this in mind: Math in any program requires setting up a formula. With Pathagoras (as opposed to other programs), the formula is shown 'in-line' and on the face of the document. That makes it more accessible for editing, and more accessible for 'seeing' what is happening.

 Once the variable in the in-line formula has been 'calculated,' the variable can be used in other parts of the document without the 'formula'. So once 'total price' or 'half unit price' or 'termination date', etc. have been used and processed in a document, that variable can be referenced later in the document as a 'normal' variable, e.g., [total price] or [half unit price] or [termination date]. After all, it's just a plain text variable!

 By default, Pathagoras will return the calculated value in number of days. To return weeks, months or years, be sure to include the term 'weeks' 'months' or 'years' somewhere in the formula. E.g.,

John Smith worked for Spacely Sprockets for [#Math#duration in weeks=[Client 1 Empl Start Date] - [Client 1 Empl End Date]] weeks.

5.4 Date Format Arguments

Pathagoras let's you present the same date in both the 'regular' (e.g., "August 23, 2019"), 'extended' (e.g., "23rd day of August, 2019") and shortened (e.g., '8/23/19') formats in different locations in the document, *based on a single presentation of that date*. It is done simply by appending certain 'arguments' to the base date variable.

- So, in one document location, create the 'base' date variable. E.g., **[Date of Contract]**
- In a second location, create the identical variable, but add the text "(ext)" (or other appropriate argument) immediately after the end of the variable name and before the closing bracket. E.g., **[Date of Contract(ext)]**
- When you scan the document with Instant Database, only the base variable will appear on the IDB screen. Provide a date in the adjacent text block on the IDB screen.
- When you press the Next button, the variables **[Date of Contract]** and **[Date of Contract(ext)]** will be replaced with the appropriately formatted dates.

Base date, completed with the date August 23, 2020:

[Date of Contract] = August 23, 2020)

Acceptable arguments (shown in bold, but emphasis not required):

- o [Date of Contract(**ext**)] = twenty-third day of August, 2020
- o [Date of Contract(**num**)] = 8/23/20
- o [Date of Contract(**num4**)] = 8/23/2020
- o [Date of Contract(**day**)] = 23
- o [Date of Contract(**month**)] = August
- o [Date of Contract(**mon**)] = Aug
- o [Date of Contract(**year**)] = 2020
- o [Date of Contract(**yr**)] = 20
- o [Date of Contract(**long**)] = August 23, 2020 (use when variable entered as '8/23/20')
- o [Date of Contract(**daylong**)] = Sunday, August 23, 2020
- o [Date of Contract(**euro**)] = 23 August, 2020 (written for users with 'USA' default date style who occasionally need European date presentation)

Notes:

- There should be no space after the last word of the 'real' variable and the 'argument.'
- It does not matter the formatting of the 'real' variable. Pathagoras will make all conversions. (So, the value of [Date of Contract] in the above example could have been '8/23/20'.

5.5 Auto-Create Incrementing Variables

Most variables are manually created by inserting them at strategic points in your document as '[' + variable name + ']'. You can also globally create variables semi-automatically using the Create Variables (Alt-V) tool. See this link.

But you can also create a series of variables from another variable that itself suggests the 'need' for variables. Pathagoras can automatically create the variables associated with a *'number of'* call. E.g., the variable '[Number of Children]' leads to [Child@1], [Child@2], etc.; [Number of Shareholders] can automatically result in [Sharholder@1], [Shareholder@2], etc.

Here's how:

1. Create a variable in your document that contains the word 'Number' (in any position).
2. After scanning the document, or recalling a intake form (mask), double click on that variable.
3. Pathagoras will ask 'how many (children, shareholders, etc) are there?,' (taking its cue from the variable name itself). Provide a numeric answer.
4. Pathagoras will then ask you for a 'base name' (and suggest one for you, again based on the 'number of' variable).
5. Pathagoras will create and increment the number of variables that match the 'how many' answer.

Exercise:

Type [Number of Children] in a document. Press Alt-D to bring up the IDB screen and Scan for the variables.

Double click on the variable. Answer the prompts. Watch the action.

Note: This tool would be needed only in the situations where the variables are completed independently of document creation, e.g., using a mask or intake form. If the document is created first and the <<*Repeat* . . .>> command is used to duplicate and increment the base variable, and the document is then scanned using the Instant Database, there would be no need to auto-create (pre-create) the variables. The IDB scan of the document would pick up the variables that the <<*Repeat* . . .>> command created.

Note: Yet another automation tool exists to augment your variables list. If you create a multiple-choice list of variables (perhaps to select a variable representing an Actor, if you select a variable from the list, and if that variable doesn't already exist at the left side of the IDB screen Pathagoras will create that variable for you so you can provide a final value for that variable. See [this page in the Manual](#)⁶² for further explanations and illustrated examples.

5.6 '=' Equivalency Function (single element)

A user wrote:

"We use the variable '[OurClient]' to capture the name of, and produce documents and letters to or concerning, our client. But for documents

prepared in the course of litigation, 'Our Client' is sometimes the '[Plaintiff]' and sometimes the '[Defendant]'. But our client is always '[OurClient]'. It sure would be nice to have a way to quickly equate the data recorded next to [OurClient] to either the [Plaintiff] or the [Defendant] value field in the IDB mask without risking a typo."

You can easily set the value of one variable to automatically be equal to the value of another variable. Simply put in the following equivalency function: "[=variablename]" (no quotes) in the right column of the IDB screen next to any other variable. See the below screen shot where the variable [Plaintiff] is equal to the value of [ClientName].

The screenshot shows the 'Pathagoras: Instant Database' window. It features a table with two columns: 'Document variable:' and 'Replace With:'. The table contains several rows of variables and their corresponding values. A red arrow points to the '[ClientName]' variable in the first column, and another red arrow points to the '[Plaintiff]' variable in the second column, which is set to '[ClientName]'. A red button is visible between the two columns for the '[Plaintiff]' row. The interface also includes a 'Page' selector (1, 2), a 'Max Terms' field (60), and a 'Delete if blank' checkbox. At the bottom, there are checkboxes for 'Left column locked' and 'MultiSelect button Visible', a 'Scan' button, and a 'Power Tools' section with a 'Next' button.

Document variable:	Replace With:
[ClientName]	John Q. Doe
[VENUE LOCALITY]	Newport News
[PLAINTIFF]	= [ClientName]
[DEFENDANT]	Rachel M. Badactor
[Defendant Address]	1234 Main Road
[Defendant City, State ZIP]	Williamsburg, VA 23454

The variable [ClientName] found in position #1 of the variable list was inserted in the third line, with an '=' sign added to create the equivalency.

When Pathagoras detects the '=' sign adjacent to a bracketed variable at the 'right', it displays a red button between the two columns.

When you click on the red button, "[Client Name]" will become "John Q. Doe".

If you click the red button again, the 'equation' is restored so you can edit the formula or (if a multiple choice selection is supplied) choose another item from the list.

You can concatenate two, three or four individual variables into a new variable. E.g., [Full Name] = [First Name] & [Middle Name] & [Last Name]. Click this link for [more information](#)⁹⁴.



Creating the equivalency can be done with a Drag and Drop action if the two variables are on the same IDB screen page. First, make sure the 'Enable Drag Drop' box is checked under Power Tools. Next, drag the name of the first variable into the 'value' spot for the second variable. Pathagoras will even add the '=' sign for you automatically.

See [next page](#)⁸⁷ for a discussion on how to assign a role to one of several possible actors.

5.7 '=' Equivalency Function (*Actors* and their Roles)

On the [previous page](#)⁸⁵, we assigned the value of a variable in a mask to that of a known precedent variable. In that example, we assigned [Plaintiff] in the document to be equal to [Client Name] in the database. We call this an A = B equivalency.

But oftentimes the direct A = B assignment is not practical. When you are drafting a Will, for example, the Executor can be one of several people. It most often will be the spouse, but it can just as easily be a child or another relative or even a non-relative. And the possible actor filling the role of the Alternative PR can be quite large.

Using *Aliases*, Pathagoras allows you to create equivalency lists that will allow you to select on-the-fly a desired 'actor' for each 'role' from a multiple choice selection. The alias contains a listing of the various roles a specific actor can assume.

Here is a possible setup for a Will using a nested multiple choice variable to assign the actors to

LAST WILL AND TESTAMENT

OF

[TESTATOR NAME]

I, [TESTATOR NAME] declare this writing to be my Last Will and Testament.

Family

I am married and my spouse's name is [Spouse Name].

We have two children together, namely, [Joint Child@ 1 Name] and [Joint Child@ 2 Name]. I have no other children from prior relationships

Personal Representative

I hereby name as my Personal Representative [PR:=[Spouse Name]/=[Joint Child@ 1 Name]/=[Joint Child@ 2 Name]] and if my said Personal Representative cannot server, I appoint [AltPR:=[Spouse Name]/=[Joint Child@ 1 Name]/=[Joint Child@ 2 Name]] in his or her place.

The above theoretically is an acceptable setup. When you press Alt-D, it asks for values for the various actors and allows you to assign one of the actors (in the above example, a spouse or a child) to the PR or AltPR role.

The problem is that as soon as [Spouse Name] - the first variable on the page -- is replaced, perhaps with Jane Doe, all subsequent appearances of [Spouse Name] are also replaced. The multiple choice variables [PR:=[Spouse Name]/=[Joint Child@ 1 Name]/=[Joint Child@ 2 Name]] will have change to [PR:=Jane Doe/=[Joint Child@ 1 Name]/=[Joint Child@ 2 Name]] and Pathagoras won't be able to find it. (That's just how plain text replacements work.)

What is needed is a way to set up the variable so that it remains intact in the document. And that exactly what **Aliases** will do. While the value of the **Alias** at the right might change, the variable itself does not.

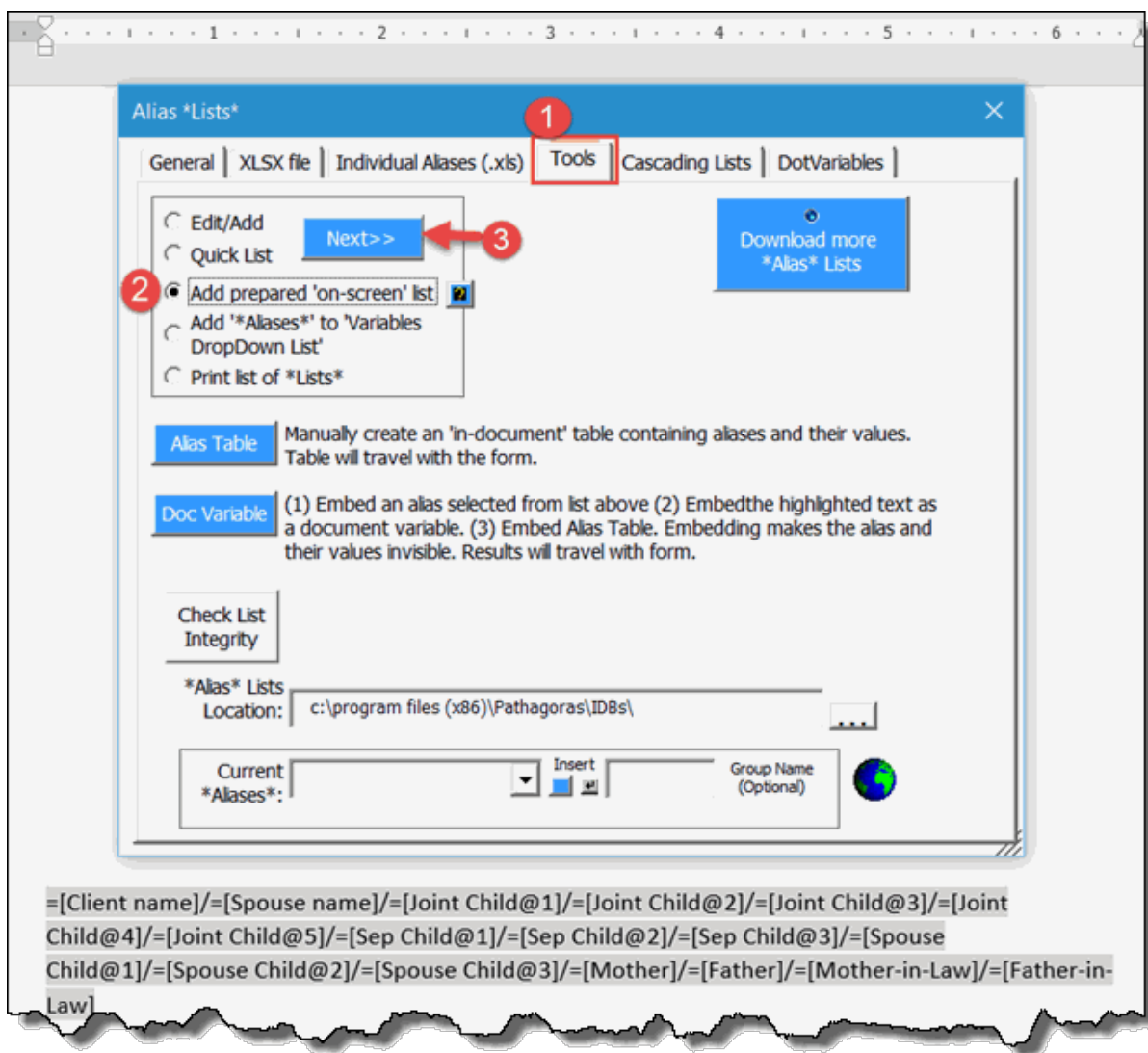
Assigning Equivalencies to an **AliasList** (Click here to read more about [*Alias*Lists](#)¹¹⁴):

You may recall that you can assign a word or term (an 'alias') to represent all 50 United States, or the 212 countries of the world, or the 278 Ben & Jerry's Ice Cream flavors. So too can you represent the large variety of actors you wish to make available for various roles they might play.

The below is a typical list of actors in a Will. (Your list could be different, and much longer. This is for example purposes only.) Copy this list of potential actors to your editing screen:

= [Client name] /= [Spouse Name] /= [Joint Child@ 1 Name] /= [Joint Child@ 2 Name] /= [Joint Child@ 3 Name] /= [Joint Child@ 4 Name] /= [Joint Child@ 5 Name] /= [Sep Child@ 1 Name] /= [Sep Child@ 2 Name] /= [Sep Child@ 3 Name] /= [Spouse Child@ 1 Name] /= [Spouse Child@ 2 Name] /= [Spouse Child@ 3 Name] /= [Mother Name] /= [Father Name] /= [Mother-in-Law Name] /= [Father-in-Law Name]

- Highlight the text.
- Call up the **Alias** Lists editor. "Pathagoras Features | Authoring/Editing **Alias** Lists | Tools". (1)
- Check the 'Add prepared 'on-screen list' radio button (2). Your screen should look something like this:



- Click the blue "Next>>" button (3). When you are asked to name the 'alias,' you can use any term you want, but we suggest something like 'Actors'.

The 'Equivalency Function in action.

Now that you have an 'Actors' alias, you can begin assigning actors to their various roles.

The below represents text that illustrates the need for this equivalency feature. The Personal Representative section, contains a simple multiple choice variable from which a specific relationship is identified, but the actual name of that relative is called for in the PR and AltPR *Actors* alias. Copy the above text into a new document. (We assume that you have added the *Actors* list [above](#) to your Alias list.) Follow the identical steps as in the example at the top of the page.)

LAST WILL AND TESTAMENT OF

[TESTATOR NAME]

I, [TESTATOR NAME] declare this writing to be my Last Will and Testament.

Family

I am married and my spouse's name is [Spouse Name].

We have two children together, namely, [Joint Child@1 Name] and [Joint Child@2 Name]. I have no children from prior relationships.

Personal Representative

I hereby name as my Personal Representative my
[PR:husband/wife/spouse/son/daughter/brother/sister] [PR:*Actors*]. If my Personal
Representative named in the preceding sentence cannot serve, I appoint my
[AltPR:spouse/son/daughter/brother/sister] [AltPR:*Actors*] in [PR:his/her] place.

Because the equivalency alias *Actors* is used twice in the document, we assigned titles 'PR:' and
'AltPR:'
to the respective appearances of the formula. That way, Pathagoras can distinguish one use from the
other.
(We could also use a !groupname! instead of a title, but the position of the 'relation' list do not pair up with
the actors alias. Do so when several related *aliases* --
ones that depend upon the values of the other -- are being used.)

Press Alt-D to bring up the Instant Database screen. Press Scan. You should see the below:

Pathagoras: Instant Database

Input Form Masks: ☐ Add New ☐ Save

Existing Records: ☐ Add New ☐ Save

Page: 1 2

Max Terms: 60

Show .xlsx

Document variable: Replace with: ☐ Delete if blank ☐

[TESTATOR NAME]	*	
[Spouse Name]	*	
[Joint Child@1 Name]	*	
[Joint Child@2 Name]	*	
[PR:spouse/son/daugh	*	
[PR:*Actors*]	*	PR:
[AltPR:spouse/son/dau	*	
[AltPR:*Actors*]	*	AltPR:
[PR:his/her]	*	

☒ Left column locked

☒ Scan in Headers, Footers, etc.

Scan 10

<Functions>

Power Tools

? Next>>

C:\Program Files (x86)\Pathagoras\IDBS

You can select a relationship ('husband', 'wife', 'son', 'daughter' from the simple multiple choice dropdown. (Note: this can also be converted to an 'alias' list.)

Now its time to assign the previously named actors to the roles of PR and AltPR. To do so, just click the dropdown list to the right of the appropriate row and select a variable (in this case, the spouse.)

Pathagoras: Instant Database

Input Form Masks: ☐ Add New ☐ Save

Existing Records: ☐ Add New ☐ Save

Page: 1 2

Max Terms: 60

Document variable: Replace with: ☐ Delete if blank ☐

[TESTATOR NAME]	*	
[Spouse Name]	*	
[Joint Child@1 Name]	*	
[Joint Child@2 Name]	*	
[PR:spouse/son/daugh	*	spouse
[PR:*Actors*]	*	PR:
[AltPR:spouse/son/dau	*	PR:
[AltPR:*Actors*]	*	[Client name]
[PR:his/her]	*	[Spouse name]
	*	[Joint Child@1]
	*	[Joint Child@2]
	*	[Joint Child@3]
	*	[Joint Child@4]
	*	[Joint Child@5]

☒ Left column locked

☒ Scan in Headers, Footers, etc.

10

<Functions>

Power Tools

C:\Program Files (x86)\Pathagoras\IDBS

Pathagoras: Instant Database

Input Form Masks: ☐ Add New ☐ Save

Existing Records: ☐ Add New ☐ Save

Page: 1 2

Max Terms: 60

Replace with: ☐ Delete if blank

Document variable: ☐

[TESTATOR NAME] *

[Spouse Name] *

[Joint Child@1 Name] *

[Joint Child@2 Name] *

[PR:spouse/son/daugh] * spouse

[PR:*Actors*] * = [Spouse name]

[AltPR:spouse/son/dau] * brother

[AltPR:*Actors*] * AltPR:

[PR:his/her] *

☒ Left column locked

☒ Scan in Headers, Footers, etc.

Scan 10

<Functions>

Power Tools

Next>>

C:\Program Files (x86)\Pathagoras\IDBS

Note the red button that appears between the two columns when the equivalency function is activated.

To illustrate the next step, we have added values to the first 5 variables. Now, when you click the red button, Pathagoras will return the referenced value to the appropriate line.

Notes:

When you select an item from the equivalency *alias* that already appears above the item you are setting, Pathagoras will insert the "=" in front of your selection to set up the equivalency. ***If the variable you select does not yet exist in the Instant Database record, Pathagoras can create it for you.*** Just double click on the right-side field. (What is happening here is that upon the double click, Pathagoras scans all existing variables (left hand side, but all pages) to determine if the variable exists. If it does, it notifies you of the page it exists on. (This is really cool, so we are emphasizing it here.) Pathagoras will insert the new variable just beneath your selection so you can assign a personal value.

5.8 Concatenation of Variables

Concatenation:

You can combine ('concatenate') two, three or four pre-existing variables to create a new variable. Type (in the left column if the IDB screen) the new variable beneath the existing variables. (It does not have to be immediately beneath an existing variable. It can also be on a different page.) In the

right column, type the two to four existing variables that you want to combine. You must include the '=' sign to indicate a 'calculation event' and the formula, and an ampersand ('&') to indicate the concatenation.

So, if you have the variables [First Name] and [Last Name] in your Instant Database mask, but wish to use the variable [FullName] in the document, you do not need to type the First and Last Name values again. Just provide in the Instant Database mask the target variable '[FullName]' at the left and the equivalency formula at the right.

Variable		Value
[First Name]		John
[Last Name]		Doe
[FullName]	■	=[First Name]& [Last Name]

A red square appears in the 'middle' column when the '=' is present.

Click the red square between the two columns and the equivalency is 'calculated'.

(The space between the & and [Last Name] is needed to assure the proper formation of the new variable.)

Variable		Value
[First Name]		John
[Last Name]		Doe
[FullName]	■	John Doe

Click it again and the formula is restored for editing, if necessary.

In-line concatenation:

The above can be also accomplished 'in-line' (i.e., within the document body. (This avoids the need for a mask.)

Pathagoras will see the in line formula as a complete variable. It will automatically create the equivalency formula at the right with no action on your part. Click the red button that will appear between the two columns to toggle between the formula and the calculated value.

Let's assume that [First Name] and [Last Name] (the 'precedent' variables) reside in the 'upper part' document. You now want to reference the concatenated value of those two variables somewhere below the first appearance of the precedent variables. Here is the formula that would reside in the actual document.

[#Concat#Full Name=[First Name]& [Last Name]]

When the document is scanned, the precedent variables are (presumably) found first and inserted into the IDB screen. When the #Concat# instruction is encountered, entire block is place into the variables list (left side) the variable. The formula is then automatically parsed from the #Concat# block and placed in the 'values' column of the IDB screen

Variable		Value
[First Name]		
[Last Name]		
[#Concat#FullName]	■	=[First Name]& [Last Name]

(Note: just like in the IDB screen, the example doesn't show the entire very long variable in the left column.

If you hover your mouse over the variable, it will display in its entirety.)

A red square appears in the 'middle' column when the '=' is present.

Variable		Value
[First Name]		Eleanor
[Last Name]		Roosevelt
[#Concat#FullName]	■	=[First Name]& [Last Name]

Click the red square between the two columns and the equivalency is 'calculated'.

Variable		Value
[First Name]		Eleanor
[Last Name]		Roosevelt
[#Concat#FullName]	■	Eleanor Roosevelt

Click it again and the formula is restored for editing, if necessary.

5.9 <S>pell out Function

If you wish your replacement text to be, or include, the spelled out equivalent of the Arabic numeral, precede or follow the variable with

- <S> (for just a conversion to the spelled out version of the number) or
- <SS> (for both short and spelled out versions of the number).

The case of the markers (<SS> vs. <Ss> vs. <ss>) controls the case of the spelled out value.

The placement of the markers controls the placement of the 'spelled-out' value. If <SS> is placed first, the spelled out amount is the 'primary' value, and the number appears in parentheses. If <SS> is placed last, the number is the primary value, and the spelled-out value appears in parenthesis.

Example:

Document text:

Thank you for your order of [quantity]<SS> widgets.

If, when you display the IDB screen and type in 65344 beside the variable [quantity], the text will become:

Thank you for your order of 65,344 (SIXTY FIVE THOUSAND, THREE HUNDRED FORTY-FOUR) widgets.

Document text:

Thank you for your order of <ss>[quantity] widgets.

If, when you display the IDB screen and type in 65344 beside the variable [quantity], the text will become:

Thank you for your order of s65,344 (sixty five thousand, three hundred forty-four) widgets.



Optionally, can manually type the <S> or <SS> markers in the 'value' column of the Instant Database screen. The same results will occur because the markers are transferred into the document and Pathagoras processes them as if they were pre-placed.

Examples:

Variable (in document and on left side of IDB Screen)	Replacement Text (right side of IDB Screen)
[Quantity]	65344<SS>
[Number Ordered]	<SS>65344

Results:

Thank you for your order of [quantity] widgets.

will become:

Thank you for your order of 65,344 (sixty five thousand, three hundred forty-four) widgets.

5.10 <\$>Currency Function

If you wish your replacement text to include a currency equivalent of the Arabic numeral, precede or follow the variable with

- <\$> (for just formatting to a dollar value style) or
- <\$ \$> (for both short and spelled out versions of the dollar amount).
- Append 'UC' or 'lc' to direct Pathagoras to return UPPER CASE or lower case 'spelled out' value.
(E.g., [Total Cost]<<\$ \$UC>> or [Total Cost]<<\$ \$lc>>)

The placement of the markers controls the placement of the 'spelled-out' value. If <\$\$> is placed first, the spelled out amount is the 'primary' value, and the number appears in parentheses. If <\$> is placed last, the number is the primary value, and the spelled-out value appears in parenthesis.

Example:

Document text:

The total cost of the project will be [Total Cost]<\$\$>.

If, when you display the IDB screen and type in 65344 beside the variable [quantity], the text will become:

The total cost of the project will be \$65,344.00 (Sixty five thousand, three hundred forty-four and 00/100 Dollars).

/-----/
-----/

Document text:

We are pleased to offer to perform the work for <\$\$>[Price Quoted].

If, when you display the IDB screen and type in 65344 beside the variable [quantity], the text will become:

We are pleased to offer to perform the work for Sixty-five thousand, three hundred forty-four and 00/100 Dollars (\$65,344.00)



Optionally, can manually type the <\$> or <\$\$> markers in the 'value' column of the Instant Database screen. The same results will occur because the markers are transferred into the document and Pathagoras processes them as if they were pre-placed.

Examples:

Variable (in document and on left side of IDB Screen)	Replacement Text (right side of IDB Screen)
[Total Cost]	65344<\$\$>
[Price Quoted]	<\$\$>65344

Results:

The total cost of the project will be [Total Cost].

will become:

The total cost of the project will be \$65,344.00 (Sixty five thousand, three hundred forty-four and 00/100 Dollars).

We are pleased to offer to perform the work for [Price Quoted].
(document text)

will become:

We are pleased to offer to perform the work for Sixty-five thousand, three hundred forty-four and 00/100 Dollars (\$65,344.00)

5.11 <%> Percent Function

If you wish your replacement text to include a 'spelled out' equivalent of the Arabic numeral, precede or follow the number with

- <%%> (for both numeric and spelled out versions of the percentage amount).
- Append 'UC' or 'lc' to direct Pathagoras to return UPPER CASE or lower case 'spelled out' value.
 (E.g., [Contingency Fee]<%%UC> or [Contingency Fee]<%%lc>)

If <%%> is placed to the left of the number, the 'spelled-out' amount is the 'primary' value, and the formatted number appears in parentheses. If <%%> is placed at the end of the number, the spelled out number is the primary value, and the 'spelled-out' amount appears in parenthesis.

Example:

Document text:

We agree to represent you in this matter for a fee of [Contingency]<%%> of the total amount recovered

If, when you display the IDB screen and type in 33.33 beside the variable [Contingency], the text will become:

We agree to represent you in this matter for a fee of 33 and 1/3% (Thirty-three and 1/3%) of the total amount recovered.



Optionally, can manually type the <%> or <%%> markers in the 'value' column of the Instant Database screen. The same results will occur because the markers are transferred into the document and Pathagoras processes them as if they were pre-placed.

Examples:

Variable (in document and on left side of IDB Screen)	Replacement Text (right side of IDB Screen)
[Contingency Fee]	33.33<%%>
[Fee]	<%%>40

Results:

Document text:

We agree to represent you in this matter for a fee of [Contingency Fee] of the total amount recovered

will become:

We agree to represent you in this matter for a fee of 33 and 1/3% (Thirty-three and 1/3%) of the total amount recovered.

Document text:

We are pleased to offer to perform the collections work for [Fee] of any amounts recovered.

will become:

We are pleased to offer to perform the collections work for forty percent (40%) of any amounts recovered.

5.12 <F>ormat Function

If you wish the replacement text of a 'simple' numerical value to be 'formatted' in the final text with appropriate separators, precede or follow the number with '<F>' (the placement of the modifier is not critical for this function).

Example:

Document text:

Thank you for your order of [quantity]<F> widgets.

If, when you display the IDB screen and type in 65344 beside the variable [quantity], the text will become:

Thank you for your order of 65,344 widgets.



Optionally, can manually type the <F> markers in the 'value' column of the Instant Database screen. The same results will occur because the markers are transferred into the document and Pathagoras processes them as if they were pre-placed.

Example:

Variable (in document and on left side of IDB Screen)	Replacement Text (right side of IDB Screen)
-----------------------------------------------------------------	-------------------------------------------------------

[Quantity]	65344<F>
------------	----------

Thank you for your order of [quantity] widgets.

will become:

Thank you for your order of 65,344 widgets.

5.13 <Fr>action Function

If you wish your replacement text to include a 'spelled out' equivalent of a fraction, precede the number with

- <Fr>
- <FR> or
- <fr>

(the case of the text indicates the case of the replacement.

Examples:

Variable (in document and on left side of IDB Screen)	Replacement Text (right side of IDB Screen)
[Contingency Fee]	<FR>1/3
[Distribution]	<fr>1/4

Results:

Document text:

We agree to represent you in this matter for a fee of [Contingency Fee] of the total amount recovered

will become:

We agree to represent you in this matter for a fee of ONE-THIRD (1/3) of the total amount recovered.

Document text:

John Doe shall be entitled to [distribution] of my estate.

will become:

John Doe shall be entitled to one-quarter (1/4) of my estate.



You can '*pre-place*' the <Fr> markers in your source documents. (Indeed, you are more likely to do this than manually type the markers into your Instant Database screen.) Just place the markers to the immediate left of the actual variable in the source text

Example:

Document text:

We agree to represent you in this matter for a fee of <FR>[Contingency] of the total amount recovered

If, when you display the IDB screen and type in 1/3 beside the variable [Contingency], the text will become:

We agree to represent you in this matter for a fee of ONE-THIRD (1/3) of the total amount recovered.

5.14 <P>aragraph Function

If you wish your replacement text to span more than one line (perhaps to put an address block with Name, Address, City, ST ZIP), type '<P>' or '<p>' where you wish a line break to occur.

- Examples:

Variable (in document and
on left side of IDB Screen)

Replacement Text
(right side of IDB Screen)

[Name & Address] John Q. Public<P>1234 Main Street<P>Boston, MA 23454

[Signature Line] Sincerely Yours,<p><p><p>Barack H. Obama<P>President, United States

[Property Desc] <<C:\data\property descriptions\Bush, G\description.doc>>

[Name & Address]

Thank you for your interest in [Property Desc].

It is not currently for sale at any price, but under certain circumstances you may be allowed to live in it for free. Call me for details. Let me know if you have any other questions.

[Signature Line]

will become

**John Q. Public
1234 Main Street
Boston, MA 23454**

Thank you for your interest in 1400 Pennsylvania Avenue, Washington, D.C. It is not

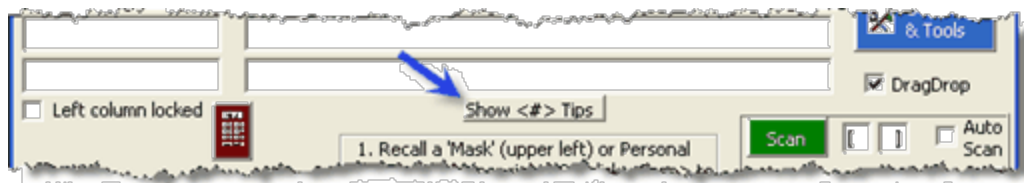
currently for sale at any price, but under certain circumstances you will be allowed to live in it for free. Call me for details. Let me know if you have any other questions.

Sincerely Yours,

Barack H. Obama
President, United States

5.15 <.x.> Function Reminders

You can get a 'reminder' of the <P>, <S> and <\$> functions discussed above directly from the Instant Database screen. Just click the "Show <#> Tips" button just below the last replacement text box. A box will pop up with examples and usage tips.



The Pathagoras System

Instant Database Records

Part



VI

6 Instant Database Records

After you have completed the Instant Database form, you are presented the question "Do you want to save the contents of this Instant Database screen as a new Data record?"

If you answer this question "Yes", you will be asked to provide a name for the new record. Once you provide the name and click 'OK', the data on the screen is saved into a file bearing the name you just provided. Once saved, the data can be recalled and reused any number of times. This is the 'database' aspect of the Instant Database module.

A couple of pointers:

- The name you provide can be anything. Your client's or customer's name; a file number; a combination of the two. (A 'classic' naming style is 'Doe, John' --last name, comma, first name).
- The entries in the Existing Data dropdown list are presented in alphabetical order. Therefore, if you use a client or customer's name as the record identifier, we recommend the "LastName, FirstName" style.
- Many law firms represent major clients with many files associated with a specific client. Those customers may wish to store individual records in sub-folders beneath a parent folder named for the client name. Pathagoras allows this in its 'tree service.' You can create a 'tree' in the same folder in which data records are originally stored.
 - To activate 'tree service' for your records, click Utilities/Settings from the Pathagoras drop down features list. Click All Settings and then select the Instant Database tab. Click the box labeled "Enable Tree Service for Data Records."
 - Once tree service has been enabled, the next time you save a Data Record, you can select (or create) the sub-folder into which you wish to save the record.

6.1 Sharing Data: Instant Database Records

Your Instant Database records are initially stored in the folder 'C:\Program Files (x86)\Pathagoras\IDBs'. You can easily re-point that location to a 'better' one. You would typically do so if you want to share records you create with other users (and conversely to allow you to use records created by others).



If you have a single Pathagoras license, you will not likely be changing the location of your Instant Database records.

But if you have purchased the Network version of the program, or otherwise want to make your records more accessible (perhaps cloud based), you can easily tell Pathagoras to keep your records in a different (common) folder.

To 're-point' the location where your IDB records are stored, follow one of the following three alternatives.

- a) **From the Instant Database screen:** Right-click on the text box that shows the current location of the files

Pathagoras: Instant Database

Input Form Masks: ☐ Add New ☐ Save

Existing Records: ☐ Add New ☐ Save

Page: 1 2

Max Terms: 60

Document variable: Replace with: ☐ Delete if blank

☐ Left column locked

☒ Scan in Headers, Footers, etc.

Right-Click in text showing IDB files path to re-point.
A menu will appear. Select the item that says "Repoint"

OR

b) From the Instant Database screen:

1. Click the red <Power Tools> button.
2. Click the <More Tools> button on the resulting screen.

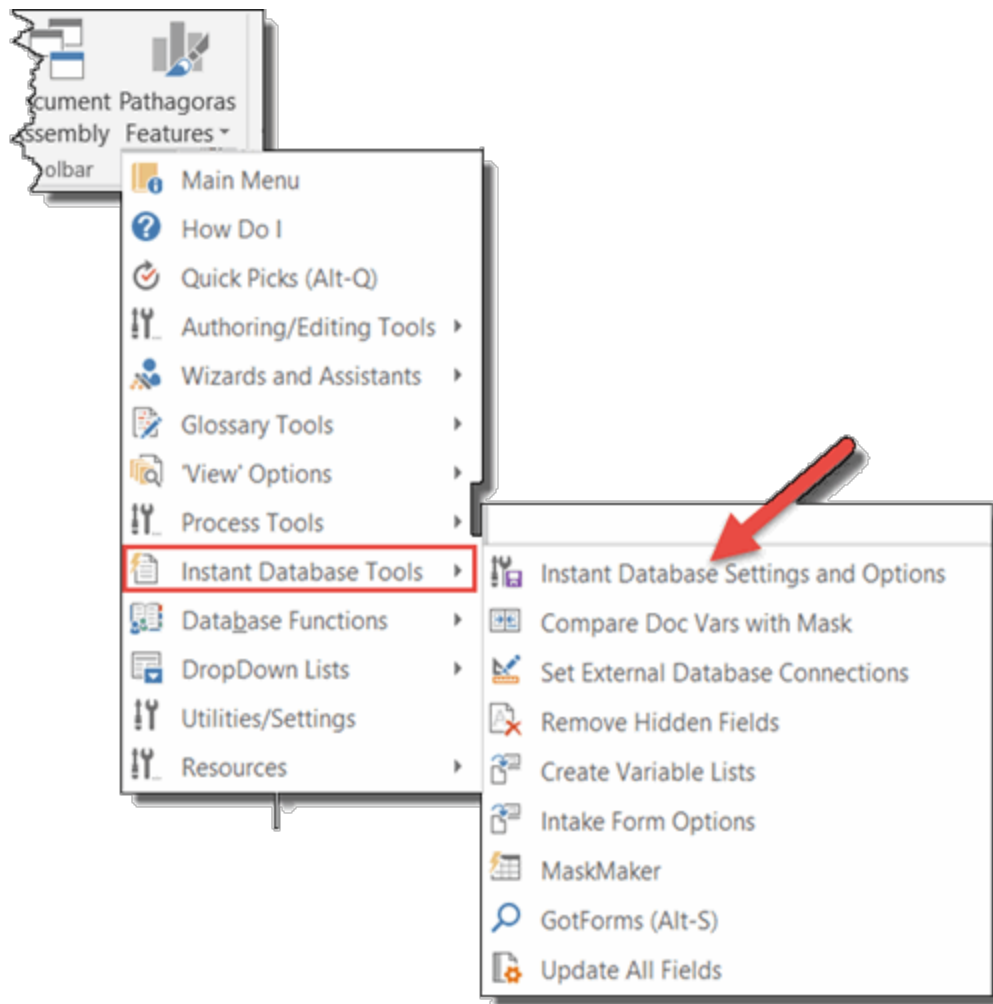


Click Power Tools | More Tools

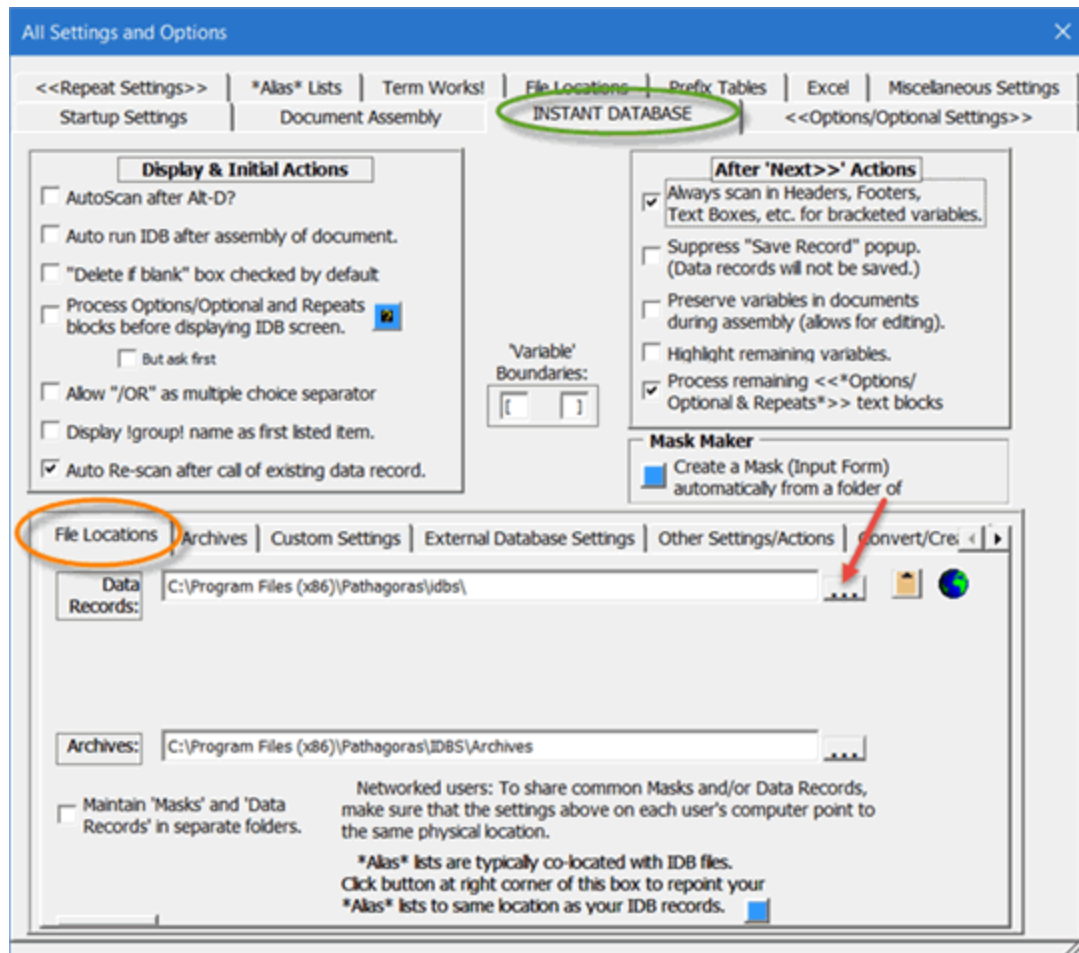
OR

c) Via Pathagoras Features:

Click the *Instant Database Tools /Instant Database Settings and Options* .



'B)' and 'C)' above lead to this screen:



IDB Tools (File Locations tab exposed).
 Click the button with the ellipse (3 dots) and follow the prompts
 to select a new location to store your IDB records.

NOTES: It does not matter where you locate your Data records, or what you call the folder in which they are saved. We do, however, recommend using 'IDBs' as part of the path name. It just adds one more check that another user has selected the right folder.

There is no requirement that all users use the same database. Some offices want to share everything and therefore all users must point to the same location. But some large offices only want records within a particular practice group to be shared. Not a problem. Those in practice group 'A' would point to a common location, other those in practice group 'B' (C, D, etc) would point to a different, distinct locations. Some offices share nothing, and run all instances of Pathagoras as totally independent operations. That's fine as well.

6.2 Changing Variable Names in Existing Records

Possible scenario: When you first started working with Pathagoras, you called your variables "[Name of Client]" and "[Address of Client]". (Note that when these sort alphabetically, they likely will not appear next to each other.)

Now that you have more experience with Pathagoras, and your system has matured, you have decided that you want your variables names such that the information 'groups together' better. You want the variables to be "[Client Name]" and "[Client Address]".

You can make the change in future source documents by using "[Client Name]" and "[Client Address]" from the outset. You can change existing source documents to reflect "[Client Name]" and "[Client Address]" using Pathagoras' Search and Replace tool. (Note: Search and Replace will work against all documents in a selected folder, so the replacement can be done quite rapidly.)

Having done that, you now need a tool that allows you to make your existing data records compatible with your new documents.

You can replace a (now) 'bad' variable name with a 'good' variable name using another one of Pathagoras' replacement tools. This one will inspect your entire collection of records and (following the above example) change "[Name of Client]" to "[Client Name]" in each record. You can change up to 3 variables at a time.

Here's how:

From the Instant Database Setting & Options Screen.

- a) Click the "Convert/Create" tab from the bottom set of tabs.
- b) (Optional) If you have activated the IDB 'Tree Service', select (either by typing or by navigating) the folder from which the table will be created.
- c) In the lower set of fields, type the old variable(s) in the top row and the new variables in the bottom row.

(The below image is a bit outdated. You can now replace at one time up to 3 'old' variables with new ones.)

All Settings and Options

MultiChoice *Lists* | Term Works | File Locations | **Prefix Tables** | Document Management | Excel | Miscellaneous Settings

Startup Settings | Document Assembly | **INSTANT DATABASE** | <<Options/Optional Settings>> | <<Repeats>>

Display & Initial Actions

☐ AutoScan after Alt-D?

☐ Auto run IDB after assembly of document.

☐ "Delete if blank" box checked by default

☐ Process Options/Optional and Repeats blocks before displaying IDB screen. ☒ But ask first

☒ Allow "/OR" as multiple choice separator

☐ Display !group! name as first listed item.

☒ Auto Re-scan after call of existing data record.

Manage Files

Label301

'Variable' Boundaries: []

After 'Next>>' Actions

☒ Always scan in Headers, Footers, Text Boxes, etc. for bracketed variables.

☐ Suppress "Save Record" popup. (Data records will not be saved.)

☐ Preserve variables in documents during assembly (allows for editing).

☒ Highlight remaining variables.

☐ Process remaining <<Options/Optional & Repeats*>> text blocks

Mask Maker

☒ Create a Mask (Input Form) automatically from a folder of documents.

Archives | Custom Settings | External Database Settings | Miscellaneous Settings | **Convert/Create** | Password

☒ Convert 2 column table (or two columns from table) into Instant Database Record.

Create/Replace

Folder containing records: c:\Program Files\Pathagoras\IDB5\

Create table of IDB records containing these variables: (Row 1 first, then second row)

[First Name]	[Middle Name]	[Last Name]

Rename 'Old' Variable with 'New' Variable in Existing Records

Replace: [Name of Client] with [Client Name] ☒ Replacements will be made in all records in folder designated above.

The Pathagoras System

Aliases

Part



VII

7 *Aliases*

[Multiple choice variables](#)³⁷ can contain an unlimited number of choices and therefore can get very long. Imagine typing into your document a list of all 50 United States of America. (Don't imagine. . . see the box below.) Not only is the footprint very large, but typing it into multiple different documents where a listing of states may be required borders on ridiculous.

Pathagoras lets you reference long (however you may define 'long') lists via a device we call an '*Alias*'. An '*Alias*' is simply a word or phrase that represents a multi-item list of items. The '*Alias*' is identified and recognized as such by Pathagoras when it is enclosed between two asterisks.

Compare this 'bad' example (a perfectly 'legal' but clearly inefficient way to present a list of the 50 United States):


Travels 'R' Us is pleased to advise that we have completed the itinerary for your summer vacation. On the first leg of your trip, we have scheduled you to visit the great state of [Alabama/Alaska/Arkansas/Arizona/California/Colorado/Connecticut/Delaware/Florida/Georgia/Hawaii/Idaho/Illinois/Indiana/Iowa/Kansas/Kentucky/Louisiana/Maine/Maryland/ Massachusetts/Michigan/Minnesota/ Mississippi/Missouri/Montana/Nebraska/Nevada/New Hampshire/New Jersey/New Mexico/New York/North Carolina/North Dakota/Ohio/Oklahoma/Oregon/Pennsylvania/Rhode Island/South Carolina/South Dakota/Tennessee/Texas/Utah/Vermont/Virginia/Washington/West Virginia/ Wisconsin/Wyoming].

with this 'good' Example:

Travels 'R' Us is pleased to advise that we have completed the itinerary for your summer vacation. On the first leg of your trip, we have scheduled you to visit the great state of [*States*].

The values that you assign to an '*Alias*' are stored in a simple Excel spreadsheet. You easily can open, edit, add to, delete from, and otherwise manipulate this spreadsheet at will. The content of the spreadsheet (like Pathagoras variables) is all plain text. The column headers reflect the 'Alias' (shortcut name) and the entries in rows 2 through (no limit) reflect the values the Alias represents.

The spreadsheet containing the '*alias' entries is named 'multichoice.xlsx'. By default, is located in the Instant Database folder.

 The list of the 50 United States ships with the Demo and Retail versions of Pathagoras. The '*Alias*' is called "States". Therefore, you can copy any of the examples on this page and paste it into a document. Press <Alt-D> to see the described action. To refer to the '*Alias*' in a variable, use [*States*]; in a simple options call it: { *States* }.

Aliases 'in action':

When an '*Alias*' is encountered (remember, the surrounding asterisks make the alias), Pathagoras opens 'multichoice.xlsx' and reads the top row. (In the spreadsheet, the asterisks are not used.) If the alias is present, it captures the entries in the column beneath the term and assigns the choices (each choice being a cell) to the '*Alias*'.

(1) With **Instant Database**, the choices are displayed in a drop down list on the Instant Database screen. (This, of course, is identical to the way multiple choice variables are

presented. In other words, while only the *Alias* appears in your document, it is as if you manually typed the entire list of choices in the body of your document.)

(2) With **Options blocks** (regular or simple), the choices are shown in an overlay screen during initial document processing. (Again, this is identical to the way that Options are presented if each was physically present in the document.)

Notes: When you think of what can be an *Alias*, think broadly. They can represent anything that you can think of: the attorneys in your office, the cities and counties served by your practice, a product line, a list of all Ben & Jerry's ice cream flavors. The list can be used time and time again. The same *alias* can even be used in the same sentence. By changing the language within the variable to reflect a different context, you can get the desired result. Continuing with the above example:

Travels 'R' Us is pleased to advise that we have completed the itinerary for your summer vacation. On the first leg of your trip, we have scheduled you to visit the great state of [Leg1*States*] for four days, and then you will travel to [Leg2*States*] for three. The return portion of your journey will take you through [Leg3*States*] and [Leg4*States*] (two days each).

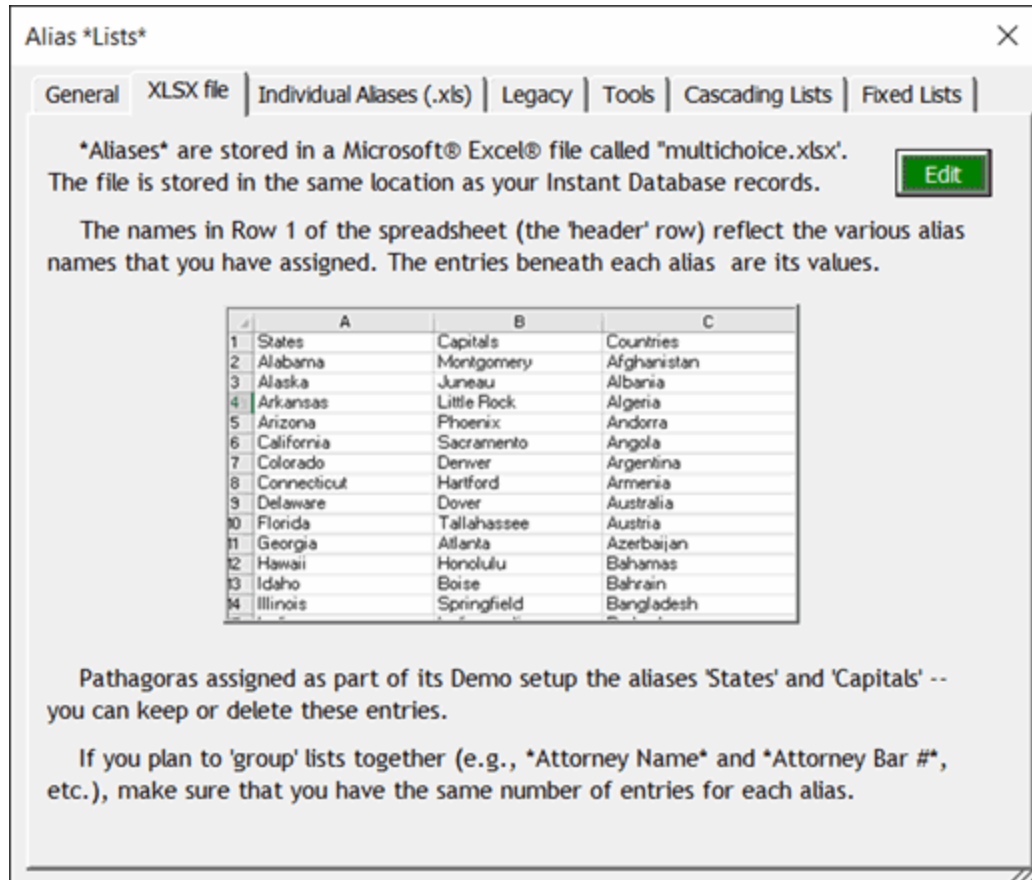
7.1 Creating *Aliases* and Lists

Aliases, and their corresponding lists of values are located in an Excel spreadsheet called 'multichoice.xlsx'. Typically, the file is located in the same folder as your Instant Database records. (Most likely, this location is 'c:\program files (x86)\Pathagoras\IDBs\'. If you are working in a network environment, you or the supervisor may have moved it to a common location on the server.)

The 'multichoice.xlsx' spreadsheet is a standard Excel spreadsheet. You can manually edit this spreadsheet as any other spreadsheet. You can add a new *Alias* by typing any word or short phrase in row one and type a list of choices beneath it. Each choice occupies a single cell. The elements must be contiguous, i.e., no blank columns and no empty cells between entries. (Any blank column indicates the end of the *Aliases*. Any a blank cell indicates the end of the choices for that particular *Alias*.)

Accessing the *Alias* lists (for editing the lists:

1. If you know the physical location of multichoice.xlsx, you can navigate to it manually. (The location of the folder is as indicated in the above paragraphs.)
2. The more classic technique to open multichoice.xlsx is by clicking Pathagoras Features | Authoring Editing Tools | *Alias* Lists). Press the Edit button.



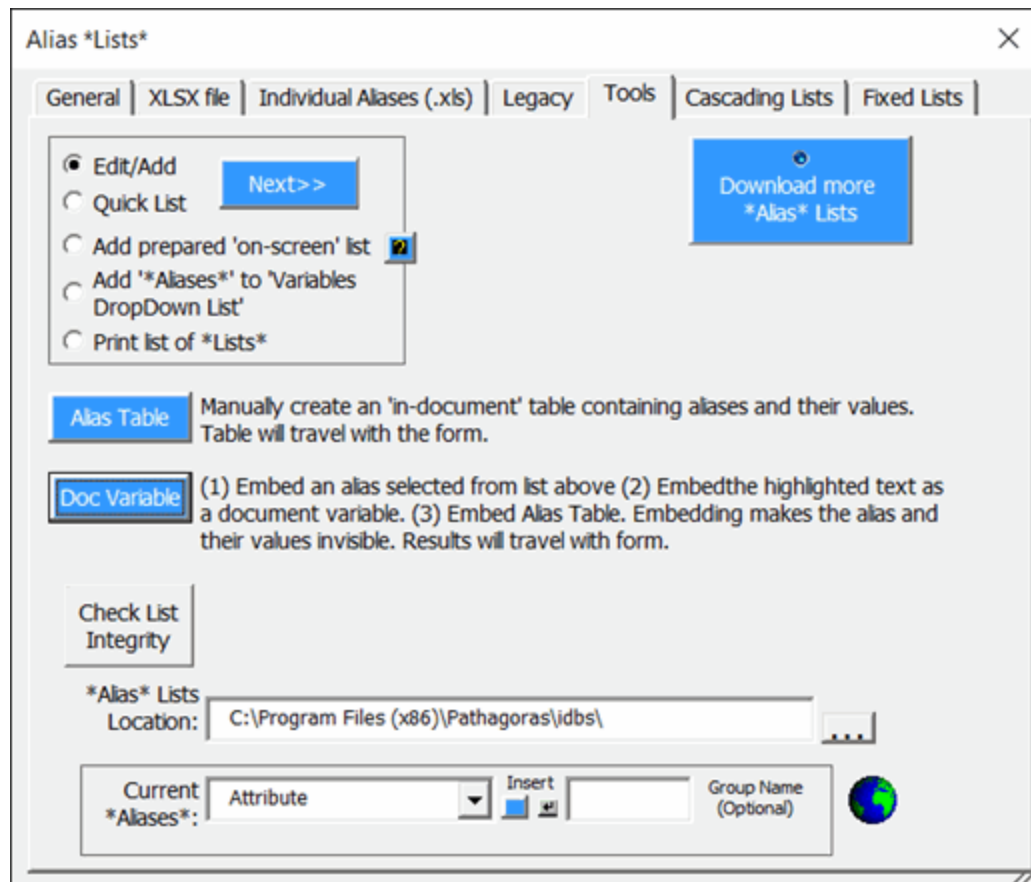
Click the Edit button and the *Alias* spreadsheet will open in Excel.
Edit as needed from there. Very easy.

Alias list rules:

- The first item at the top of a list is the *Alias* name.
- The second through the last items in the list are the values assigned to the *Alias* List.
- There is no limit to the number of *Aliases* (top row entries) you can have.
- There is no limit to the number of choices each *Alias* can have.
- **BLANK COLUMN WARNING:** There cannot be a blank column between *Aliases*. When Pathagoras sees a blank column, it assumes there 'nothing more' to find.
- **BLANK CELL WARNING:** There cannot be an empty cell in a listing of *Alias* values. When Pathagoras encounters a empty cell when reading down a list, it assumes there is 'nothing more' to find. If you must leave a blank, use a period, dash or hyphen as a place-holder.

Creating *Aliases* from within Word:

You can create new *Aliases* and corresponding lists of values from lists and tables in a Word document. It is immaterial how the list is created. You can type it by hand, import if from other sources. Any list of any length can be used. Highlight the list or table you intend to add as an alias. Just click the 'Tools' tab and click the 'Add prepared on-screen list'. Follow the prompts.



If you select 'Quick List', Pathagoras will ask for the name of an *Alias* and then for a listing of up to 10 choices the *Alias* is to represent. See 'Prepared Lists' section of this Manual for the different kinds of lists that Pathagoras can automatically add for you.

The following bullets describe the action of the other options presented in the upper left corner of the screen:

- **The 'Edit/Add' option:**

When you click [Next>>](#), Pathagoras will open the multichoice.xlsx spreadsheet in Excel. Edit as appropriate.

- **The "Quick Add" option:** You can add a list (typically a short list, since you will be manually typing the entries) of multiple choices directly from the keyboard. When you select the Quick Add option and press [Next>>](#), Pathagoras will ask you for the name you want to assign to the variable and then ask you to type up to 10 choices that you want the variable to represent. When done, click OK and Pathagoras will save the choices into your *Alias* collection.

- **'Add prepared on-screen list option:** This is discussed and illustrated in the [next section](#)¹¹⁸.



Pathagoras lets you easily add a new **Alias** and List to your collection 'on the fly'. If you want to create an "Alias" that you know does not currently exist in your collection, simply type in a document a variable representing the non-existing alias. E.g, [**Attorney Name**]. When you scan the document following an Instant Database call, Pathagoras will detect the 'error' and notify you that the **Alias** does not exist. It will offer you the opportunity to create it.

7.2 Prepared Lists

Many users already have content in the nature of lists (or can create that content) in a Word document, Pathagoras lets you add new **Aliases** and corresponding Lists directly from Word.

To add such a list to your **Alias** collection:

- Highlight the list (examples of prepared lists are shown below).
- Click '**Alias**' in the Pathagoras Features | Authoring/Editing Tools.
- Select the 'Tools' tab and choose 'Insert Prepared List' from the options.
- Click [Next>>](#) and follow the prompts (in the first three example lists below, Pathagoras will ask for the **Alias** name you want to assign in the first step. In the fourth example, the **Alias** was already assigned.)

Examples of Lists that can be easily added to your **Alias** collection:

- A **simple row of text**, with choices separated by slashes, e.g.

John Q. Adams/Thomas Jefferson/Abraham Lincoln/Richard M. Nixon/Gerald R. Ford/William J. Clinton

- A **column of text**, e.g.:

John Q. Adams
Thomas Jefferson
Abraham Lincoln
Richard M. Nixon
Gerald R. Ford
William J. Clinton

- A **single column table**

Type each entry into a separate row of the table.

John Q. Adams
Thomas Jefferson
Abraham Lincoln
Richard M. Nixon
Gerald R. Ford

William J. Clinton

- **From a two column table.**

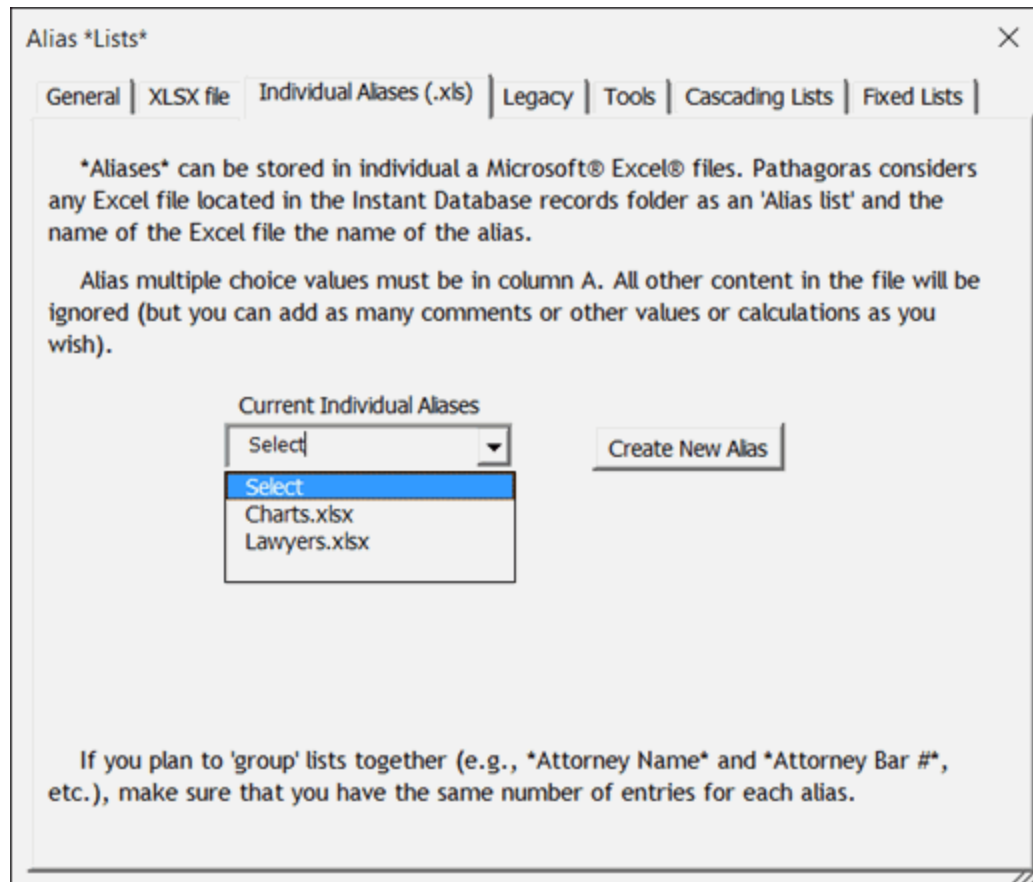
The 'alias' name is provided in the left column. The values (separated by slashes) in the right.

Colors	red/blue/green/orange
Attorneys	John Q. Adams/Thomas Jefferson/Abraham Lincoln/Richard M. Nixon/Gerald R. Ford/William J. Clinton
Attorney Bar Numbers	1111/22222/33333/44444/5555
Attorney Telephones	(202) 555-1212/(205) 334-1452/(354) 555-3468/(890) 555-2009/(556) 444-4444
Sizes	X-Small/Small/Medium/Large/X-Large/XX-Large
Ice cream flavors	chocolate/vanilla/strawberry/Rocky Road/pistachio/cherry swirl/Neapolitan/banana nut
Mailing types	First class mail/First class (return receipt requested)/UPS/Parcel Post/FedEx/DHL/Facsimile transmission/E-mail/Carrier pigeon

7.3 Other *Alias* List containers

There are two other files that can contain *Aliases* and their Lists (in addition to Multichoice.xlsx)

- The first is a spreadsheet that bears the name of the *Alias* itself. Two requirements for this kind:
 1. The name of the spreadsheet must match the name of the *Alias*. (In other words, if you want to refer in your document to your *Alias* as "Attorney Name", the spreadsheet name must be "Attorney Name.xlsx" (or .xls)
 2. The spreadsheet must be saved to the folder in which your default *Alias* file is saved. By default it is stored in the Instant Database folder. (You can determine the current location of your Instant Database folder by pressing <Alt-D> (for Instant Database) and looking toward the bottom of the screen. The full path to IDB files is shown there in a text box. You can highlight the path and then copy the path into your clipboard by pressing Ctrl-C.) Each value must be on a separate line.
 3. You can view your existing 'individual' aliases by clicking the Individual Aliases tab. You can also create a new 'individual alias' via this tab.



- New Pathagoras 2019: The second is a 'custom' alias list that you can associate with a particular document(s) via the Custom Properties of the document. Custom Lists are discussed at this link.

7.4 Presenting *Alias* Lists

As the previous examples suggested, you can use **Aliases** in a variety of ways. Just use the proper enclosure to signal Pathagoras what you intend.

***Alias* in a Variable:**

If you wish the **List** to be completed by the end user after the document has been assembled and as it is being personalized (i.e., when the user presses <Alt-D>), present it as a variable:

[*States*]

***Alias* in a robust Options command:**

If you wish the **List** to be processed as the document is being assembled, present it as Options text. Here, the **List** is "states"

<<*Options**States*>>. (The doubling up of the asterisks is correct. The first asterisk closes the Options command. The second asterisk opens the **alias** call.)

***Alias* in a simple options call:**

For simple options, just use curly braces (don't forget the asterisks): {**states**}

Use an *Alias* in different locations to represent different values:

If your document uses the same variable in several different contexts, you can use the same *Alias*. Just make the variable 'unique.' That can be easy to do for a simple variable. Just append a number or other character at the end. E.g., "[*List*1]", "[*List*2]", "[*List*3]" etc. Below, we use '*States*' twice, yet have a unique variable assigned for each 'leg of the trip.'

Travels 'R' Us is pleased to advise you that we have completed the itinerary for your summer vacation. On the first leg of your trip, we have scheduled you to visit the great state of [*States*1].

You will stay there for the first week and enjoy all the beauty this state has to offer. Then, you will travel to [*States*2] where you will enjoy the magnificent sites that this incredible state has to offer. We hope that you enjoy your trip.

Can I use !Groups! with *Aliases*?

Absolutely. This is the most powerful way to pair up a name with an address and with a phone number, or any other kind of association you can think of. Just precede each list that you want 'grouped' with a !groupname!. Make sure that (1) there is an identical number of options in each list and (2) that 'complementary' selections are in the same relative positions in each List.

E.g.:

[!atty!*Attorney Name*]
 [!atty!*Attorney Bar Number*]
 [!atty!*Attorney Direct Line*]
 [!atty!*Attorney E-Mail*]
 [!atty!*Paralegal*]

We have provided a few pages below a more elaborate example of *Aliases* and !Groups!.
[Click here to go directly to that page.](#)^[122]

Can I cascade *Aliases*?

Yes. *Aliases* can be nested/cascaded to 2 levels for Options, 1 level (currently) for variables. See Cascading *Aliases*.

Display *Aliases* in a DropDown List

Want to be able to instantly click in a value contained in a List? Keep the List in an always-on, always active DropDown List. [Click this link](#)^[160] for more information.

7.5 *Aliases* and !Groups!

!Groups!

The !Group! function works with *Aliases* and <<*Options*>>. (As explained [elsewhere](#)¹³⁶, a !Group! allows a user to select an element from one member of the group that results in the automatic selection of the proper entry from other members of the same group.)

Consider the following. Type (or copy and paste) the following into a document:

The capital of [!St!*States*] is [!St!*Capitals*].

Press <Alt-D> to call up the Instant Database. <Scan> the document to display the variables. Select a state (or a capital) from the drop down lists at the right of the Instant Database screen. You should see how the grouping enables the selection of one item to automatically set to the other.

OTHER EXAMPLES:

- **In a law office setting:** You might have a list of cities in which you practice. You might also have a list of Clerks, and a separate listing of their addresses, with whom you file pleadings. You could also have a list of Sheriffs from whom you request service of process. Let's further assume that you have created individual *Aliases* for each element (*city*, *clerk*, *sheriff*, *clerkaddress*, *sheriffaddress*, etc.)
- **In a non-law office setting:** You might have a list of corporations with whom you conduct business, and a list of contact points within each business to whom you send the request for proposals, etc. Let's further assume that you have created individual *Aliases* for each element (*customer*, *contact person*, *contactaddress*, etc.)

Let's now assume that you have created a form document that contains these variables. Note that each variable that you want to change 'in tandem' is preceded by a !Group! name:

```
Jurisdiction: [!city!*SampleCity*]

[Date of Letter]

[!city!*SampleClerk*]
[!city!*SampleClerkAddress*]

Re: [Client Name] vs. [Defendant Name]

Case Number: [Case Number]

Please file the enclosed documents among the other papers:

[List of Documents]

Thank you for your kind attention to this matter.

Sincerely,

[AttorneySignatureBlock]
```



In the above example, the !groupname! we chose to link the various variables is '!city!'. Please understand that the term you select for a !groupname! is not important. It could have been a non-sense word like !grzb! or a number like !123!. The !groupname! is in no way related to the variables it modifies. It is only an identifier used to tie the 'grouped' variables together.

When you run the Instant Database routine against the above text, all bracketed variables will appear in the left side of the Instant Database screen. The potential values for each variable will be displayed in a dropdown list at the right. Select any one of the !city! variables and the parallel selection of the others in the group will be automatically selected for you.

You can use *Aliases* not only with variables (as shown above) but with robust <<*Options*>> and {Simplified Options} blocks as well. Here is the structure:

Robust Options: <<*Options*!GroupName!**Alias*>>



The double asterisks between the GroupName and the Alias are mandatory. The first asterisk closes the administrative section of the Options 'call to action'. The second one (and the one following) identifies the *Alias*. The double asterisks are not required in the below two examples.

Simplified Options: {!GroupName!*Alias*}

Variables: [!GroupName!*Alias*]

See Also: [Groups](#)¹³⁶

7.6 *Aliases* and Administrative Text

It is important for us to make sure you understand how Pathagoras marks out what it calls 'administrative text'¹⁹⁸ and address the confusion that often arises when administrative text concepts collide with *Aliases*.

Administrative text is that part of an Options/Optional/AskOptions/AskOptional block that is either pure Pathagoras 'command' code or which is added as prompts and choices to provide guidance to the end user.

All administrative text is removed when the block is processed. The 'third' asterisk in a typical command line (if a third asterisk exists) marks the end of the administrative text.

An 'alias' call is a piece of text also surrounded by asterisks that similarly represents a list of choices.

<<*AskOptions* . . . :

An <<*AskOptions . . . >> call is pure administrative text. None of it will remain in the completed document. It is all 'command' and 'prompt' text. Therefore, it ends with an asterisk. So a proper <<*AskOptions* . . . >> would look like this, with two asterisks at the end.

<<*AskOptions(radio)*!Signer!*LoanOfficer**>>

To get a better 'visual' on this, imagine replacing **LoanOfficers** with Joe/Mike/Mary/Susan. You would have this:

```
<<*AskOptions(radio)*!Signer!Joe/Mike/Mary/Susan*>>
```

Hopefully you see how the '3rd ' asterisk closes the administrative text. (It's sometimes easier to 'pretend' the stars in **alias** are not really there.)

<<**Options** . . . :

An <<**Options** block, on the other hand, is administrative text closed by an asterisk, followed by text, all or some of which text may remain in the document. Therefore, in your setup, the Administrative text is <<**Options*!Signer!* and no more. The rest of the block is either going to be your list of names "Joe/Mike/Mary/Susan", or the alias representing those names. Since here, we want to represent those names using an alias, this is the proper construct:*

```
<<*Options(radio)*!Signer!**LoanOfficer*>>
```

Again, imagine the replacing **LoanOfficers** with Joe/Mike/Mary/Susan. You would have this:

```
<<*Options(radio)*!Signer!*Joe/Mike/Mary/Susan*>>
```

It is confusing at first (and maybe at second, third and fourth), but there is a method to the madness. Once you can visualize the 'stars' of an alias as being integrally connected, and the asterisk that closes administrative text is an entirely different entity.

I recognize that it would have been easier to not use asterisks in both routines. However, when programming a 'plain text' system such as Pathagoras, there are only so many keyboard characters at my disposal.

So long story short. This is the code you want (colors only for emphasis; optional and never requiredJ)

```
<<*AskOptions(radio)*!Signer!**LoanOfficer*>>
```

. . . .

Blah blah blah

```
<<*Options*!Signer!**LoanOfficer*>>
```

blah blah blah

7.7 *Alias* Files Location

The physical location of your **Aliases** is initially in the same folder as your Instant Database records. (This location is initially "C:\Program Files (x86)\Pathagoras\IDBs".) However, you may, for any reason, change this location. If you do, you must 're-point' to its location.

Your goal in this exercise could be for one of three purposes:

- (1) just to see where the *Alias* file is currently located;
- (2) to move the physical storage location of the current records; or
- (3) to reset the pointer to the location where the *Alias* file you want to use is currently stored.

You can determine the current 'pointer value' for your *Aliases* and open the *Alias* file ('multichoice.xlsx') in several ways.

Manual Navigation:

If you know its physical location, just navigate to multichoice.xlsx and open (or move) it.

Via the Pathagoras Features drop down list:

Pathagoras Features | Authoring/Editing Tools | *Alias* Lists. Click the Edit button on the resulting screen

From the Utilities/Settings screen:

Click the <Show All Settings> button from the face of the Utilities/Settings screen and then click the *Alias* Lists tab.

To move the file, or reset the pointer (and optionally move the content of your *Alias* Lists), click the 'navigate' button (the one with the three dots) and follow the instructions. Navigate to the desired location and lock in the new location by pressing "OK".

When you have finished re-pointing, and if an *Alias* file exists at both the original and the target locations, Pathagoras will ask whether you want to combine the two lists (and if so, how duplicate entries should be handled) or keep the original or the target List. Respond as appropriate.

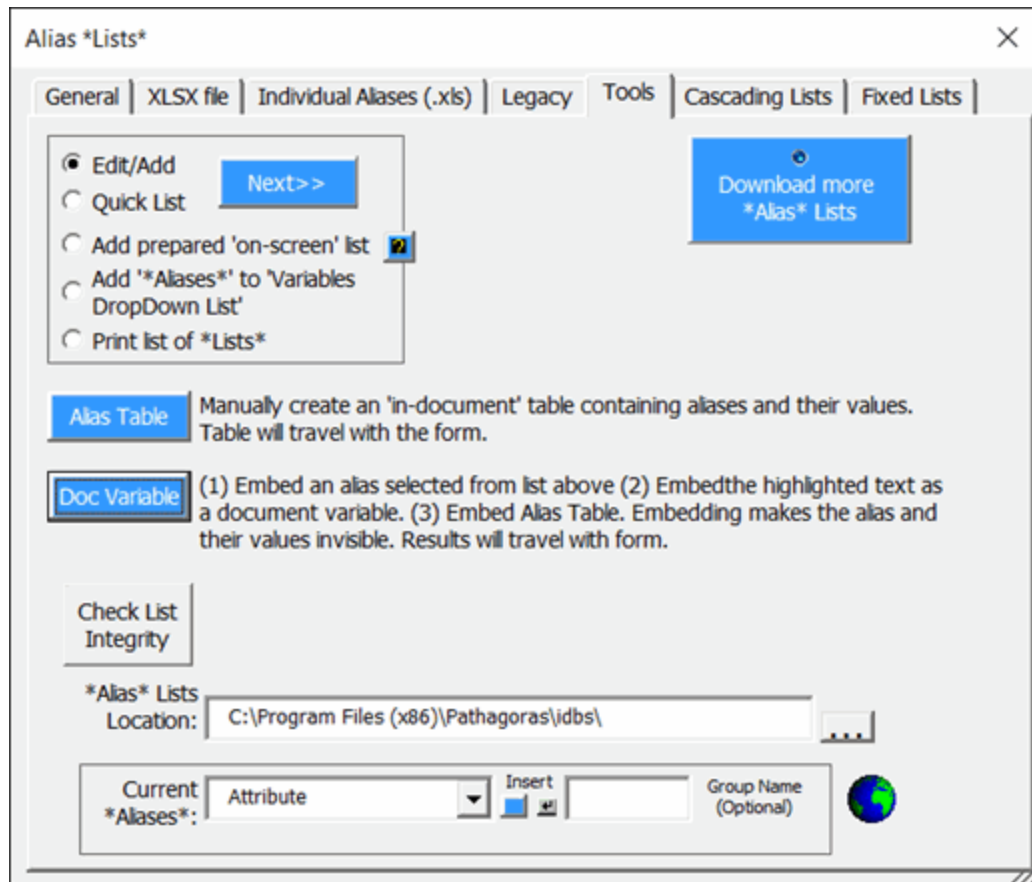
7.8 *Alias* Table (Embedded)

Typically, your *Aliases* are stored at you system level (either on your computer or if you have networked everything, on your server) in a file called "multichoice.csv". They can also be saved at Excel files (.xls or .xlsx). [See this link.](#)^[119]

However, if that file is not available to all users, the call to an *Alias* will fail. (This can happen when users are from different offices or are not otherwise pointed to a central server.)

To overcome that 'problem' and make your *Aliases* more 'portable,' you can store an *Alias* (or several *Aliases*), along with the list of choices that the *Alias* represents, within your document. Pathagoras allows you to paste the aliases you are using into the current document into an '*Alias* Table'. The table sits at the bottom of the document and can be easily seen and edited by the recipient user. (It can also be easily removed before sending to a client or customer.)

To create the *Alias* Table, display the *Alias* List screen (Pathagoras Features | Authoring/Editing Tools | *Alias* Lists). Click the More button to expand the screen.



Choose the **Alias** you want to embed in the document from the Current **Aliases** drop down list and then click the **Alias** Table. (You can alternatively embed the **Alias** and its values as hidden Document Variables. This is a good choice only if you know you or the recipient will not need to edit the multiple choice values for the *Alias*.)

7.9 **Aliases** as DropDown Lists

You can easily use the terms in an **Alias** List as DropDown List content. Perfect for when the terms in your List are short and represent a meaningful collection.

Benefits: No need to create individual documents for elements in your Lists. You can also use the Lists to quickly create sentences, numbered or bulleted lists.

[See this page](#)¹⁶⁰ for more information, examples and directions.

7.10 Sharing Your Lists with the World

Give

You can help to improve the world and make a difference in the lives of others by sharing **Alias** Lists that you have created. (Yes, that is tongue in cheek, but we would like your lists nevertheless.) Just send 'universally usable' lists that you have created to:

lists@pathagoras.com


I will post what you send to the website (the page noted below) where it can be downloaded and used by others.

Receive

Conversely, you can download lists that have already been posted to the website. Here are the steps:

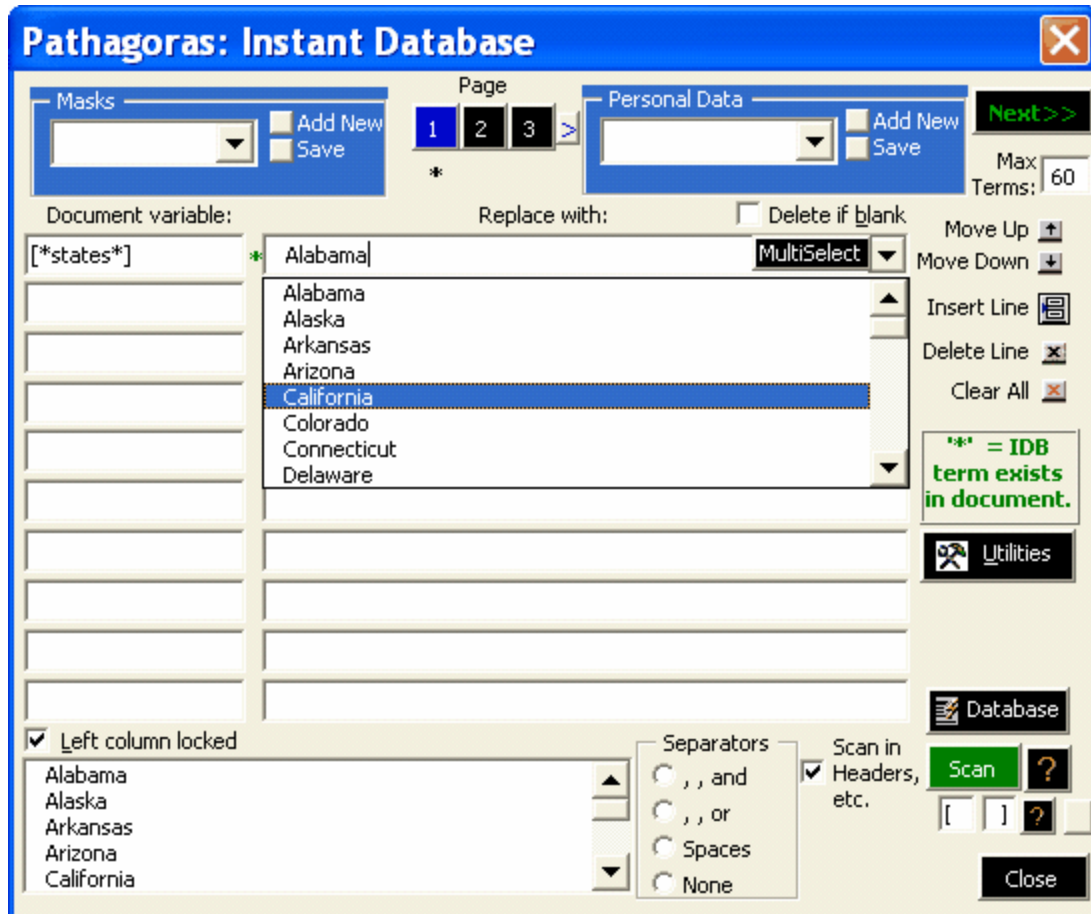
1. Visit this page: www.pathagoras.com/multichoice1ists.html.
2. Scan the current listing. If you see one that you like, highlight and copy it;
3. Return to Word and paste the list onto a blank page;
4. Highlight the entire list;
5. Click the Pathagoras Features menu and select Authoring/Editing Tools;
6. Select *Alias* Lists from the menu and select the Tools tab;
7. Click the 'Add prepared 'on-screen' list' from the upper left quadrant;
8. Click Next. Give the list an appropriate 'alias'. Click OK. That's it.

7.11 Miscellaneous *Alias* information

 **'Natural slashes':** Let's say your *Alias* text will contain 'natural slashes'. Perhaps you want to present the end users with a series of dates in the format 00/00/0000 from which to choose or an address contains an unavoidable 'slash'. You must use '/OR' as the separator between choices.(e.g., "01/28/1953/OR07/07/1988/OR6/22/99"). You must check the 'Use /OR' switch found in the Instant Database settings screen (Utilities/Settings|All Settings|Instant Database).

Visual Depictions of an *Alias* in action.

Just in case you haven't tried any of the above examples on your own computer, but you would like a quick peek at the [*states*] *Alias*, here is a 'look-see' of what the results would be:



Instant Database display of the [*states*] variable list.

Note also the multi-choice possibilities at the bottom of the screen.

You can select one, some or all of the list elements.

You can also indicate the separator, if any.

The use of **Aliases** with Options may be useful when a fluctuating number of variables needs to be brought into the document under construction. For example, in a Will, there might be several children, and a variable (e.g., [Name and Birthday of ChildX]) needs to be provided for each child. You could create a **Alias** called "children" and the values in the list might be:

[first child]/[first child] and [second child]/[first child], [second child] and [third child]/[first child], [second child], [third child] and [fourth child]/[first child], [second child], [third child], [fourth child] and [fifth child]/[first child], [second child], [third child], [fourth child], [fifth child] and [sixth child]

Here is how the above might render when {**children**} is encountered during document assembly. (The same result would obtain using <<**Options**children***>>).

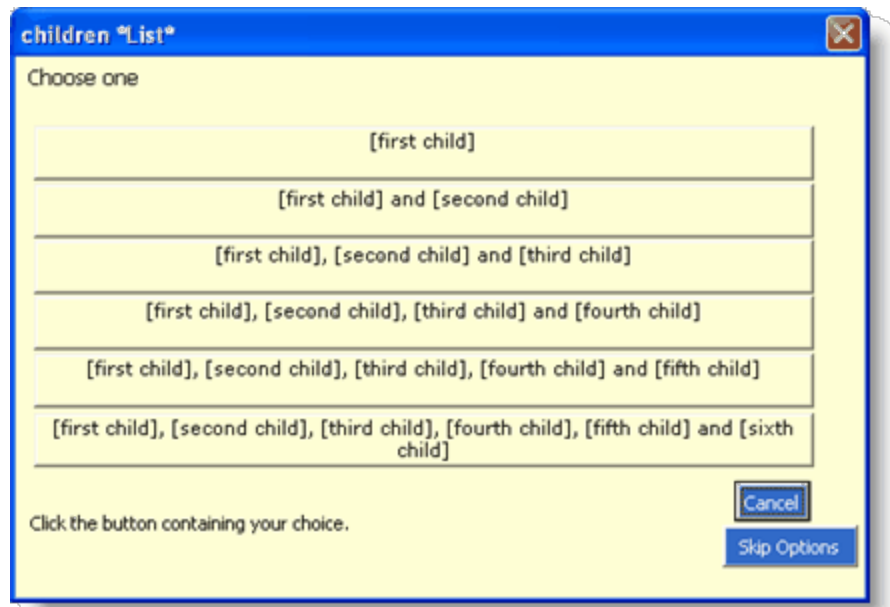


Figure 5. { *Children* } / < *Options* **Children* >> rendition in Pathogoras' options module.

Can I create Addresses (or other multi-line) MultiChoice items?

Yes. Pathogoras handles multi-line items without a problem. If you plan to use the MultiChoice list in an Instant Database screen (which does not allow true 'Enters', you must represent 'Enter' with "<P>". Pathogoras will 'decode' the <P>s at replacement time. See [<P>aragraph Functions](#)^[102]

E.g.:

Big Store<P>123 Main Street<P>Anytown, PA 23454/Little Store<P>324 Oak Lane<P>My Town, VA 43234/etc.

Can I set one of my values to the 'default' selection?

Yes. Just precede the default value with the '#' (hashtag) sign in the Excel spreadsheet. (If using a 'legacy' list, the '#' should be placed immediately preceding the default term. E.g., "Apples/Bananas/#Cherries/Dates"

7.12 Document Assembly with *Aliases*

The *Alias* tool has some pretty powerful aspects. Having a single word comprise a list of 2, 20, 200 or 2000 separate elements is powerful in its own right.

But if an *Alias* let you select several of its choices to complete a signature block, or an order form, why not use it to build documents?

Drawback:

You must select items from the Alias list in the order presented.

7.13 'Sentence' Assembly

The simple, straightforward way that Pathagoras handles *Aliases* makes it a perfect vehicle for 'sentence assembly.' Simply cobble together two or more elements of an *Alias* List and you can create elaborate sentences in the same way that Pathagoras lets you create elaborate documents.

Document assembly, as we have described elsewhere, suggests one of these two techniques (and combinations of the two, when appropriate):

- 'Paragraph assembly.' This is document assembly starting from a 'blank slate, and pouring selected paragraphs onto the page to create the desired text.
- 'Template assembly.' This type of assembly starts with an overbuilt document containing all possible text combinations. By completing an interview, and answering prompts, any unnecessary language is deleted and appropriate text retained.

Regardless of the technique adopted, Pathagoras allows you to continue to insert additional text to achieve the needs of the client, customer or patient.

The two techniques above both deal primarily with the assembly of paragraphs.

But paragraphs end with Enters and carriage returns, not readily lending themselves to building sentences. Further, the terms and clauses used in 'document assembly' are stored in individual documents.

Sometimes, however, you just want to construct your text from simple words and phrases. These terms are typically short (single words or simple phrases) and for that reason you don't want to devote an entire document to preserve just that short snippet.

Pathagoras provides you a third assembly technique. We call this '**Sentence Assembly**'. It allows you to assemble text from short snippets of text. The snippets are the terms saved at terms associated with *Aliases* as discussed in previous pages of this section. So, if you have an alias *States* representing the 50 United States of America, you can ask Pathagoras to present them in a selectable listing. Choose from the list and you can easily create a sentence or other lists.

By way of example, a psychologist may want to describe the appearance, attitude or affect of a patient using terms such as 'cooperative' or 'belligerent' or a dozen or so other phrases. Typing each word to the screen is possible (and is likely the real life practice). But if the terms have been saved into an *Alias* list, Pathagoras' 'Sentence assembly' tool can be used to build a quick description of the patient, a list of prescribed medications, or anything else. (Of course, if the *Alias* doesn't yet exist, you can easily [create it](#)¹¹⁵.)

So this is a possible collection of *Aliases* illustrating the above:

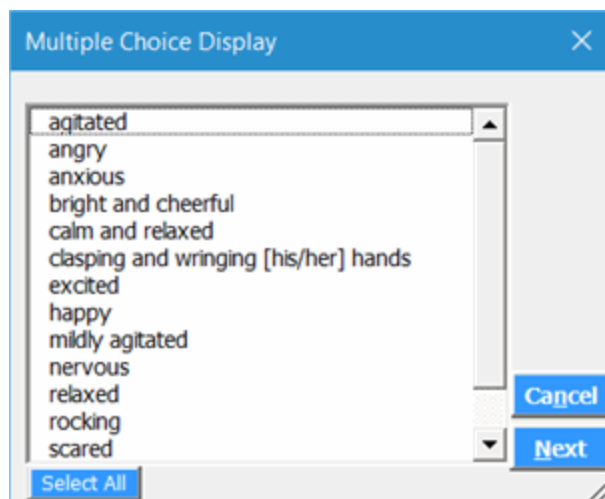
Appearance	Medication Names	Attitude
agitated	Alprazolam	agitated
angry	Amantadine	angry
anxious	Amitriptyline	argumentative
bright and cheerful	Amoxicillin	belligerent

calm and relaxed	Amphetamines	bright and cheerful
clasping and wringing her hands	Aripiprazole	calm and relaxed
excited	Atomoxetine	combative
happy	Augmentin 625 mg	cooperative
mildly agitated	Benzotropine	critical
nervous	Biperiden	derogatory
relaxed	Buprenorphine	scornful
rocking	Bupropion	thankful
scared	Buspirone	
tapping [his/her] feet	Carbamazepine	
tearful	Chlorpromazine	
	Citalopram	
	Clomipramine	
	Clonazepam	
	Clonidine	
	Clopidol Acetate (Acuphase)	
	Clopidol Acuphase	

There are several ways to 'call' the *Alias*.

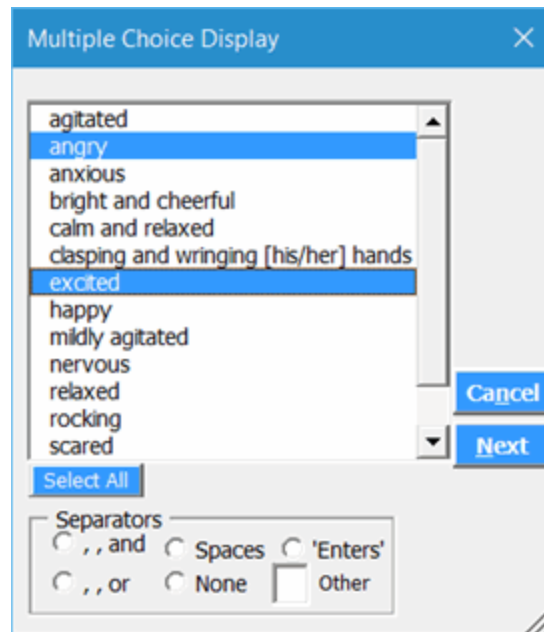
<<*Options**Appearance*>> in the document body is the 'classic' method. You can also use the simple options method: {*Appearance*}.

When the document is processed, Pathagoras will present a list of all of the *Alias* terms in a selection list, such as:



Select 1, 2, 10 or all items.

If you select more than one item, the connectors options will display:



When you hit the next button, the selected items are quickly cobbled together into a nice, compact sentence.

If you want a numbered or bulleted list, make sure the 'call' is made next to a number or bullet. Select 'Enters' as the connector.

You can also double-click on an item and the selected item will be inserted onto your editing screen.

Display *Alias* as DropDown List

You can display the terms of your *Alias* List in an 'always on' DropDown List for easy selection and insertion into your document. [Click here](#)¹⁶⁰ for the steps on how to do so.

NOTES:

While the above examples use words and short phrases, Pathagoras can easily handle much longer terms and even complete sentences. So, this section could have been called "Paragraph Assembly"

Formatting is controlled by where the items are inserted in your document, not the 'style' of text in the *Alias* source.

The tool was developed primarily to speed up the assembly of 'group-able' items. But as you can see from the examples, it can also be used as a 'speller helper' ('Medication Names') and a thesaurus.

7.14 'Aliases as Actors' List

You can assign ^Aliases* to a list of possible actors, and select those actors from a list.

This is explained in the 'Equivalency' section under [Instant Database Functions](#)⁸⁷. Click here for examples and setup instructions.

The Pathagoras System

Groups and !GroupNames!

Part



VIII

8 Groups and !GroupNames!

Sometimes, the value assigned to one multiple choice variable suggests the answer to a subsequent multiple choice variable. For example, if “he” is selected as the value for multiple choice variable [he/she/it], then, “him” or “his” may will be the selection for another multiple choice variable further down in the document if the same ‘actor’ is being referenced.

For example:

[He/she] went with [his/her] dog to the pet store because [he/she] wanted to buy [him/her] a new collar. Blue is [his/her] favorite color.

(Okay, grammarians, the last [him/her] and [his/her] sets could be better structured to prevent noun confusion, but just play along with me here.)

Note that the first decision quite directly suggests the responses to the next two. (It does not suggest the response to the third and it may or may not suggest the response to the fourth).

Without a way to group the variables, the typical action of the Instant Database replacement scheme would assign the selection of the [he/she] multiple choice to all appearances of [he/she] throughout the document. That result, of course, is not 'okay.'

Pathagoras provides the 'way.' Simply add a short prefix to each related variable. We call this prefix a 'group name' and this group name allows you to connect related sets of multiple choice variables.

To add a group name to a multiple choice variable, type a letter, word or phrase between exclamation marks at the very beginning of each variable in the document you want to be in the group.

e.g., [!groupname!variablename].

So the above could be rewritten:

[!Owner!He/she] went with [!owner!his/her] dog to the pet store because [!owner!he/she] wanted to buy [!dog!him/her] a new collar. Blue is [!owner!his/her] favorite color.

(Perhaps the last group name is supposed to be “[!dog!his/her]”?)

Once assigned, a ‘!group!’ links together all variables within the same group. The result is that the selection made for the first member of the group will trigger the answer to the remaining members of the group. This is so even when the variables are not identical. (The above example illustrates that. Of course, "his/her" is not identical to "he/she".

After the document is scanned, the multiple choice provided appear in a drop down list at the right. Drop down the list (1) and make a selection (2).


The other members of the group (but *only* of the specific group) are automatically chosen (3) below.

As you are designing groupings, don't limit yourself to simple gender choices such as those above examples. Think big!

The [!actor!man/woman] went to the [!actor!ladies'/men's] section of the department store to buy a [!actor!skirt/pair of pants] for [!actor!his/her] [!actor!wife/husband].

Try it. Copy and paste any of the above examples into a document. Run Instant Database (<Alt-D>) against them and see the results.

Keep the following in mind:

 Keep the following concepts in mind:

- Groups are not limited to just pronouns. As the above examples illustrate, you can use group names for any multiple-choice collection.
- While an answer to the one element of the group needs to be provided by the end user, it does not matter which group member you click first.
- The result displayed for group members is typically based on the *position* of the answer in the list. It is not a calculated value.
- The group name can be anything. Short is better. A single word, or even a single letter, will work.

- The group name being what Word sees to determining the 'case' (ALL CAPS, Upper And Lower, lower case) of the replacement text, of the group name will control. The examples provided above illustrate this concept.
- Groups are not limited to just variables. You can use the !group names! method for <<*Options*>> and <<*Optional*>> text blocks. [Click this link](#)^[214] for more information.



To help insure consistent spelling of your !GroupNames! throughout your documents, you can append to the bottom of your Variables DropDown List a collection of !GroupNames!. Just display a document containing your favorite !groups!.. Drop down your Variables List and select "Add Document's !Groupnames! to List. See [this page](#)^[160] for more info.

The Pathagoras System

DropDown Lists

Part



IX

9 DropDown Lists

We put the topic of DropDown Lists at the head of the pack of features to emphasize that this tool is not dependent upon your having 'Pathagorized' documents. It is an extraordinary feature of the program aimed solely at speeding locating your primary documents. DropDown Lists do away with most navigation issues every user encounters with Word. Even if you have not Pathagorized a single document, you can shave off many minutes a day, and hours a week, in document production time when you create one or two DropDown Lists and use them regularly.

One of Pathagoras' most powerful document assembly tools is also one of its simplest, both in setup and in use. It is the **DropDown List**.

A "drop down list" in general is any of those lists that reside at the top of your Word editing screen that give you quick access to Word settings. For example, Word presents various 'styles' and 'fonts' in drop down lists. Simply point and click to a new style or font and the texture of the document changes.

Pathagoras' **"DropDown Lists"** work in much the same way, but instead of changing the look of the document, you use the elements of the List to insert selected blocks of text (including whole documents) into your document under construction.

Here is a brief description of what these Lists are and what they can do.

- A Pathagoras DropDown List is a standard drop down element that resides at the top of your editing screen.
- A DropDown List reflects the contents of a folder. It is not the folder itself, but just a pointer to, and a listing of, its contents.
- DropDown Lists allow you to retrieve documents, text snippets, images and other items with simple 'point and click' action.
- **It is not required (or even important) that the documents in a DropDown List be 'Pathagorized'.** DropDown Lists can, and should, be used to help you with accessing all frequently used documents, not just your 'Pathagorized' ones. Our advice is 'create DropDown Lists now,' and Pathagorize the documents later (using the time saved when you are using DropDown Lists).
- The target folders of DropDown Lists are standard Windows folders.
- You can display up to 10 DropDown Lists at a time. (The currently visible Lists are referred to as a 'Collection'.)
- You can maintain up four separate 'Collections' of DropDown Lists. You can easily switch among your Collections via the Collections panel. (With judicious use of 'Collections,' therefore, up to 40 DropDown Lists can be simultaneously maintained.)
- Once assigned, no navigation is required to retrieve documents (or other files) reflected in a DropDown List.

- The target folder can be *any* folder. The target folder can contain any file.
 - In most cases, the folder will contain Word documents (either entire documents or building block type files).
 - But folders can also contain text files, images (.jpg, .gif, .tiff, etc., files), Excel® spreadsheets, PDF files, Word Perfect® documents or . . . , well, you get the picture -- anything.
- A DropDown List can also reference the contents of a glossary (a single document containing dozens or hundreds of separate terms). The same 'non-restrictions' apply to glossary DropDown Lists as apply to those representing folders.
- You can activate a 'Tree Service' feature, allowing the display of the contents of the parent folder and, in two clicks, the contents of any child folder beneath.
- Creating each DropDown List takes 30 seconds tops. Once created, the List remains always active, always visible and always ready. When you exit Word and then return, so do the DropDown Lists.

Create DropDown List from Document Assembly Screen

Create DropDown List from Clause Selection Screen

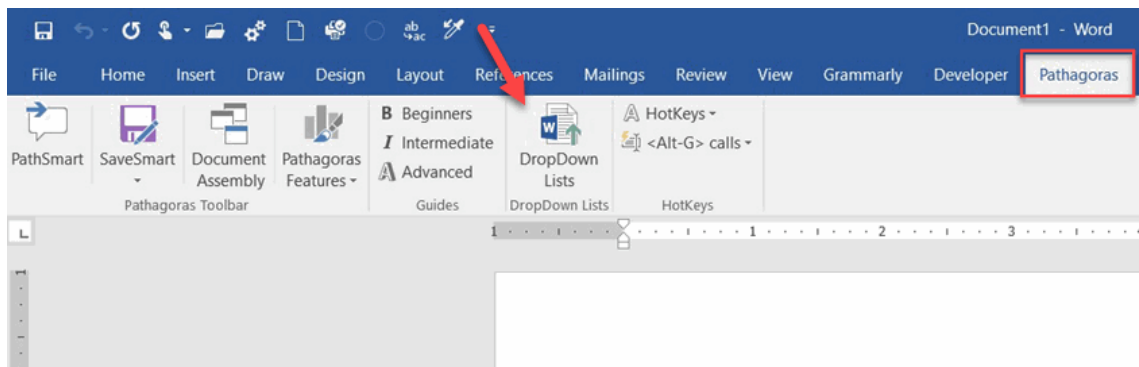
[Create DropDown List 'free hand'](#)¹⁴¹

Creating and using DropDown List 'Collections'

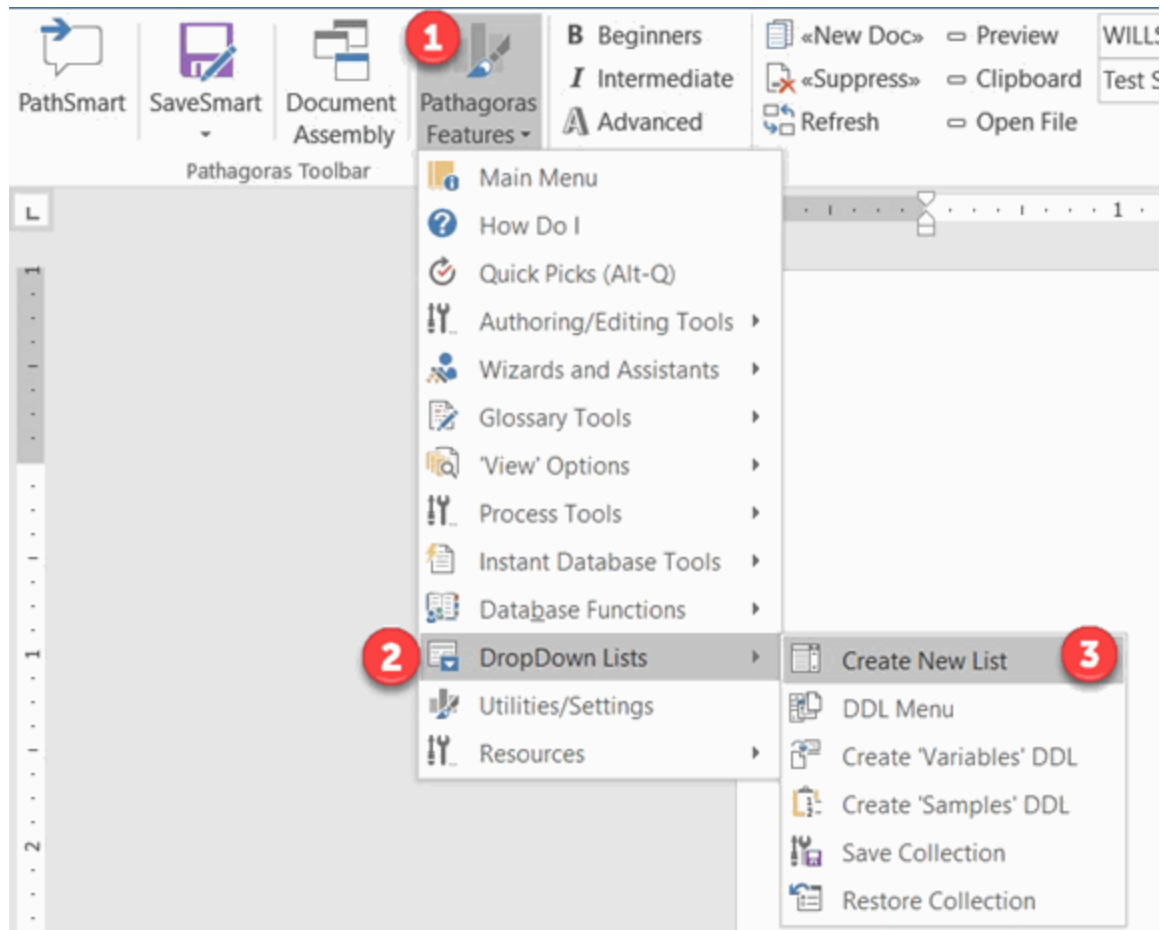
9.1 Creating 'Free Hand'

You are most likely to create your first DropDown List 'freestyle.' In other words, you will manually navigate to source content for the List.

In the Pathagoras tab, click button (the 5th from the left) that is titled 'DropDown Lists.'

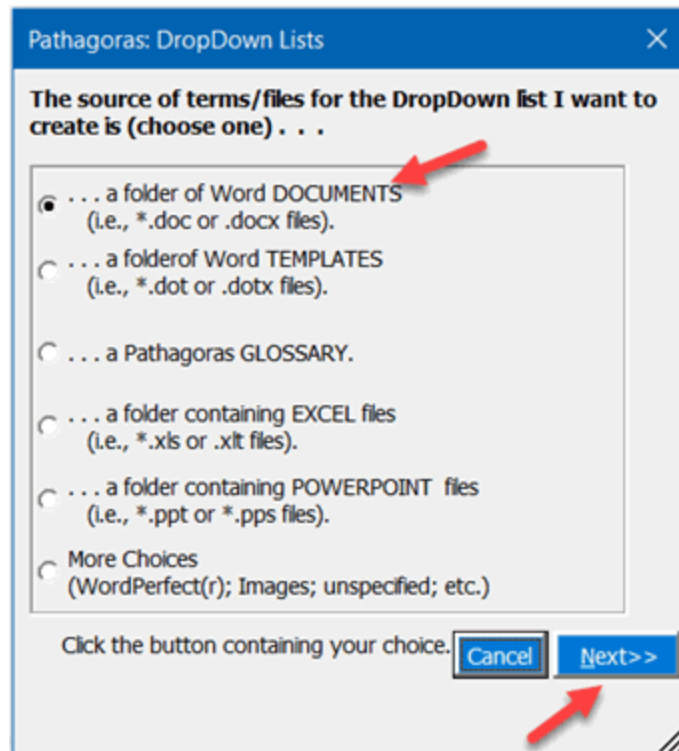


An **alternative** starting point for creating a DropDown List is the 'Create DropDown List' element in the Pathagoras features menu.

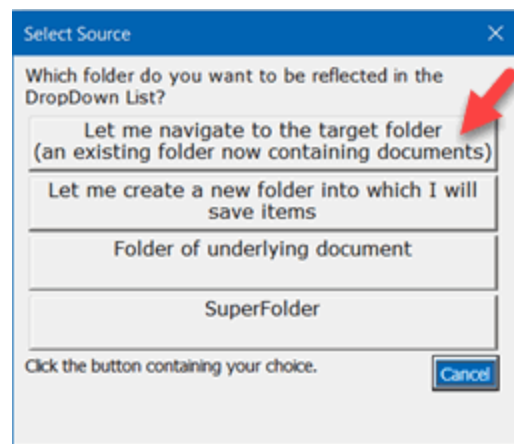


. Follow the prompts, which will ask you to:

- Select the type of content you want the DropDown List to contain. While you will have many choices, for most instances you will select the first option, 'Word Documents'.



- You will next be asked HOW you want to select the target folder of documents. Unless you know the others apply, select the first option: 'I will navigate.'



- Navigate to INSIDE OF the folder that contains the files that you wish to be inserted into the DropDown List. (By folder, we are referring to any standard Windows folder that already exists on your computer or across your network.) Select one item in the folder (it doesn't matter which one -- you are locking in the folder, not the selected document) and press OK.
- (If your target source is a Pathagoras Glossary, navigate to, and simply select, the Glossary you desire. Press OK.)

- When prompted, give the list a title. The default title will be the name of the selected folder or glossary. But feel free to name it anything you wish.
 - After you have created your first DropDown List, you will be offered one more options -- where to place the list. 'Add second (third, etc.)' or 'Replace' an existing list. Make the appropriate choice. (Tray 1 refers to the top row of Lists, up to 5, and Tray 2 refers to the lower row of Lists, up to 5.) If this is your first list, no 'where' question is posed.
 - That is all! In just a few seconds, you will see the DropDown List, containing the content of the folder or glossary in the DropDown List area.
-

OTHER COMMENTS:

- The above steps assumed your DropDown List would reflect documents in a folder. But when you display the 'Type of List' screen, note the great variety of items a List can contain. Note also that there is a second page of options. Don't hesitate to experiment with different kinds of Lists.
- Because a DropDown List is so easy to create, delete and recreate, don't get 'hung up' on which folder or glossary you select for your first one. You have 10 readily available (and 40 in total if you implement 'Collections'). You can overwrite any of them at will. Just do it!
- Pathagoras doesn't care whether a DropDown List contains links to complete documents, to individual clauses, to pictures, spreadsheets, PDF files, or whatever. If Word has the capacity to insert a particular file type, it can be the contents of a DropDown list. We recommend that you experiment with many types of Lists.

See Also:

[Using DropDown Lists](#)  148

[Image Assembly](#)  163

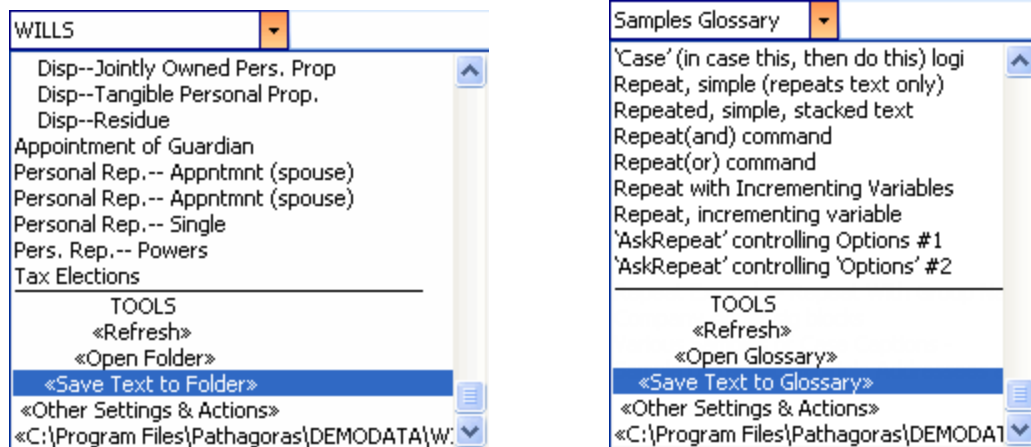
[PDF Assembly](#)  163

9.2 Adding Content to DropDown Lists

You can add clauses to a folder or glossary represented by an existing DropDown List using the list itself. Here are the steps.

- Highlight with your cursor the portion you want and then click «Save Text to Folder» or «Save Text to Glossary», as appropriate.
- Pathagoras does the rest. Provide a name and a subject and you are done!
- If you want to add the entire document on your editing screen to the DropDown List, highlighting is not necessary. Simply click the appropriate «Save Text» item.
- Don't worry about 'Folder' vs. 'Glossary.' Pathagoras knows which is which, and will only display the proper option based on the source of the List.

- There is absolutely no navigation required on your part. You don't have to Refresh the List after you add an item.




Adding text to a folder (left) or to a glossary (right) is done with point & click simplicity.



To further illustrate the elegance of this feature, we propose the following exercise:

1. Create a DropDown List for a folder to which you intend to add text (we'll call this the 'incomplete folder').
2. Display a document that contains some text that you want to add to the 'incomplete folder.'
3. Highlight the text you want to move into the folder. Click the «Save Doc to Folder» entry in the DropDown List.
4. Repeat.

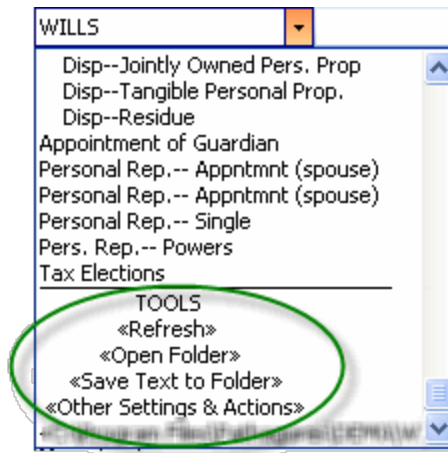
 We emphasize that when you 'add content' to a DropDown List, you are only adding a new document *to the folder* that the Drop Down List represents. Then Pathagoras re-reads the folder content and re-displays the List, including the newly added term. If you look at the source folder you will see your addition there. Further, if you perform a 'standard' document assembly routine on the same folder, the new term(s) will appear in the Clause Selection Screen.

Note: As an alternative to the above, you can manually add documents to the folder using any of the other methods that Word makes available. However, the new document won't automatically appear in the DropDown List when you next display it. Be sure to click «Refresh» to update the content.

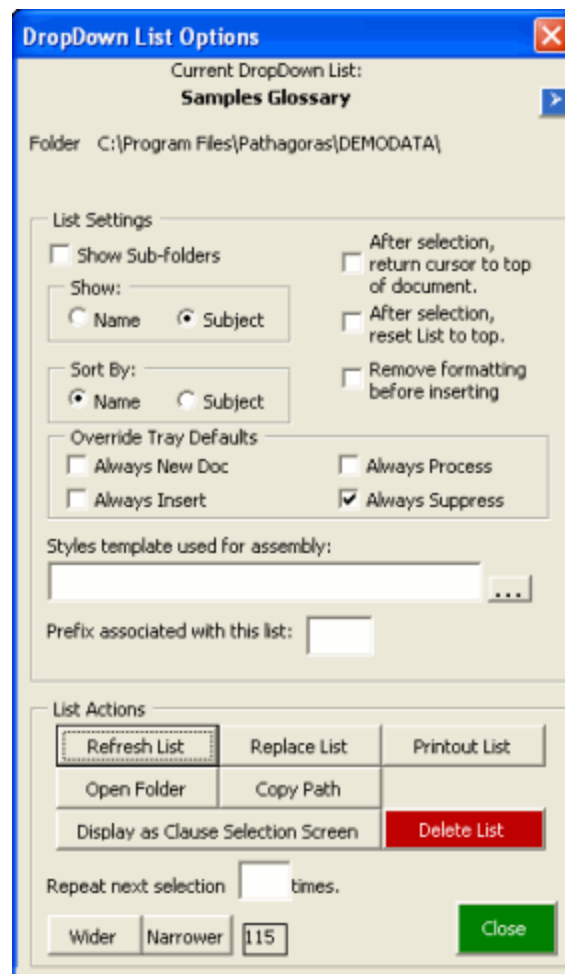
9.3 Deleting a DropDown List

To delete an existing DropDown List:

1. drop down the DropDown List you want to delete.
2. scroll down to and click «**Other Settings & Actions**» at the very bottom of the List.



3. Click the red 'Delete List' button at the bottom of the screen.



9.4 Repointing a DropDown List

If you move the contents of the folder to which a DropDown List currently points, the List will (of course) become dysfunctional. You will need to repoint the list so that it can 'grab' the files from the new location.

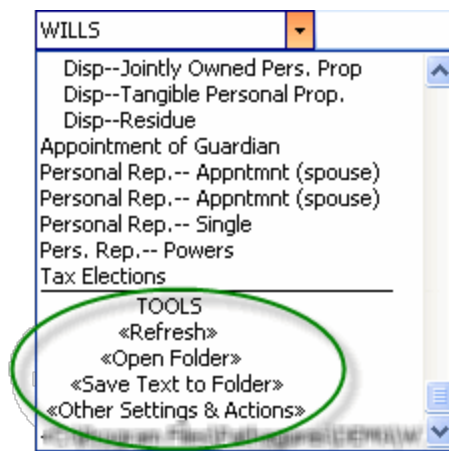
A couple of options exist.

Repoint via the Libraries & Books screen:

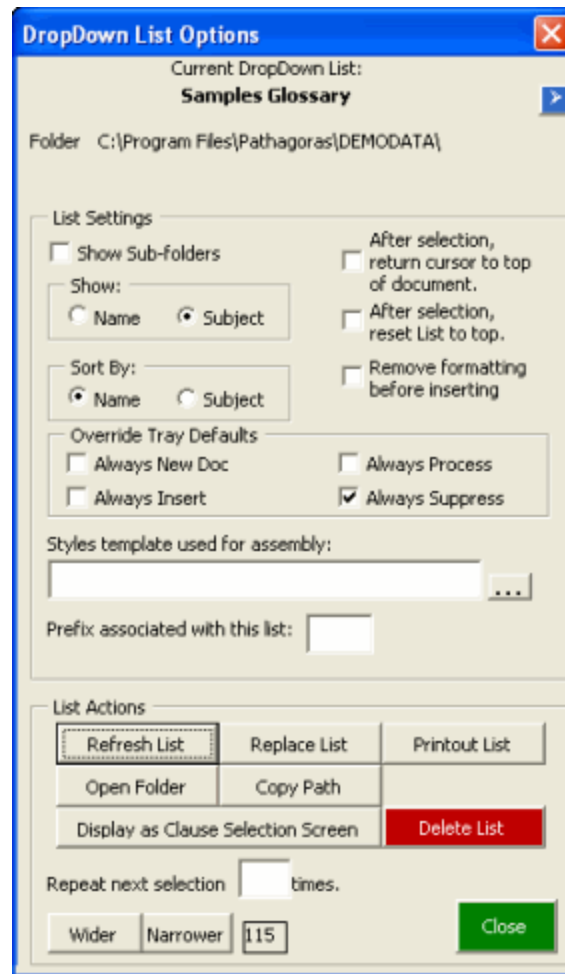
If the list reflects the contents of an existing book, you can call up the Document Assembly 'Libraries & Books' screen (the one that appears when you click the Document Assembly button. Click once on the book name. Then click the "Create DropDown List" option. Pathagoras will instantly create the List and ask which position you want the List to occupy. Select the appropriate 'tray' and 'position' of the current list. That is all.

Repoint via the actual DropDown List:

As a practical matter, the repointing function is actually the 'replacement' of the new pointer for the old. Drop down the DropDown List. Scroll down to and click **«Other Settings & Actions»** at the very bottom of the List.



This is the result"

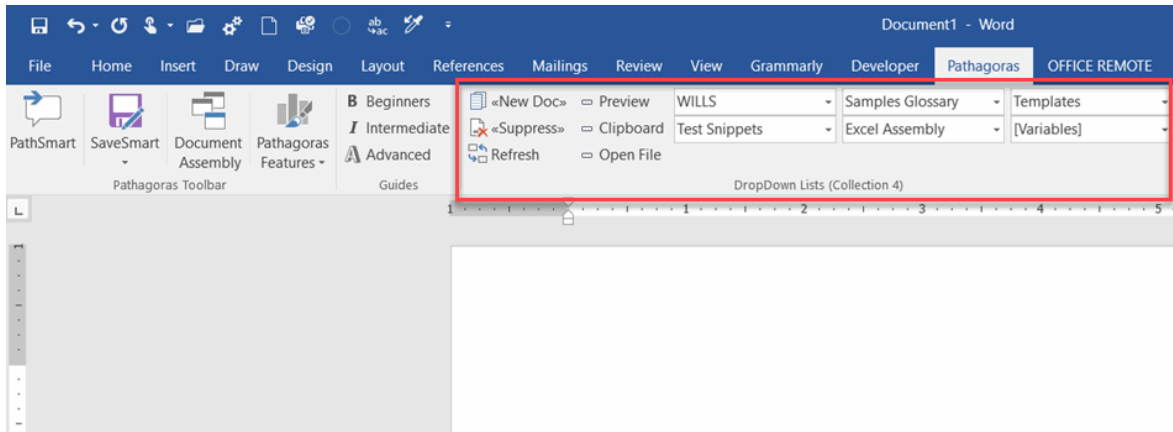


Click the 'Replace List' button. Follow the steps to navigate to inside the folder at the new location. Select one file from the folder's content (it does not matter which one). Click 'OK' to lock in the folder name and you are done.

9.5 Using a List

Using DropDown Lists

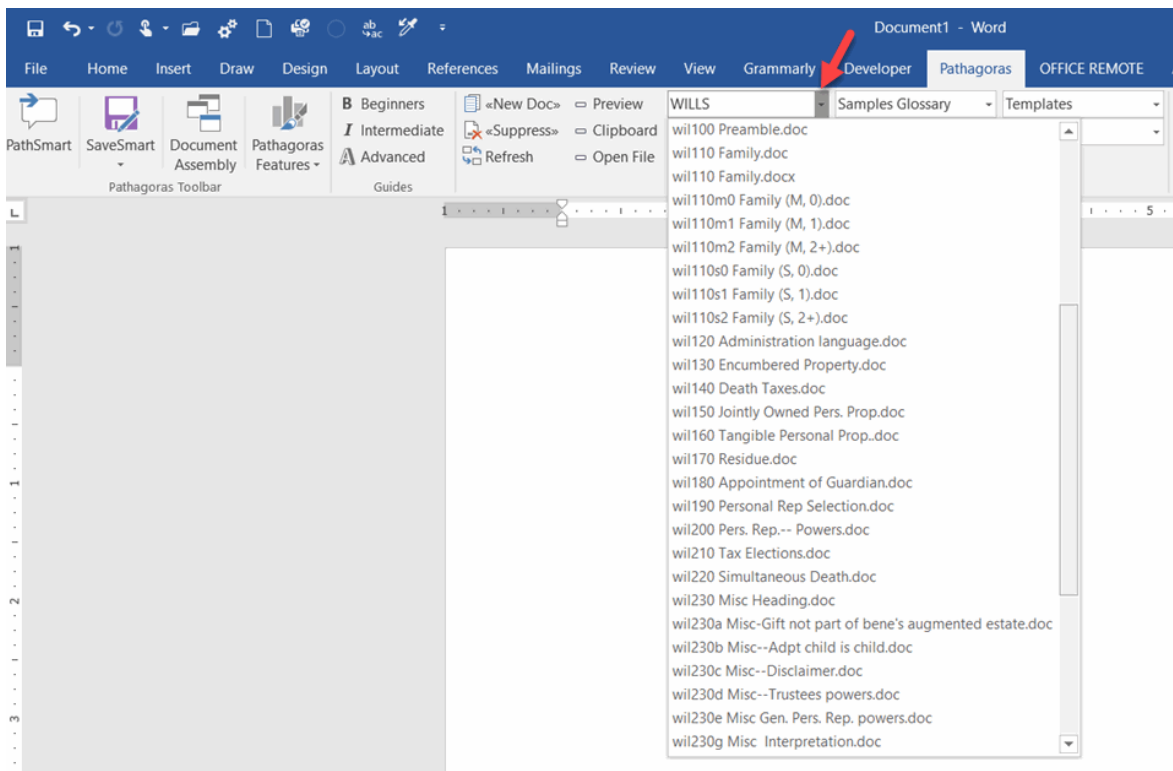
Regardless of which method you used to create the DropDown List (methods discussed in earlier sections), Pathagoras will transfer the names of the documents in the target location into a DropDown List. It is now on your screen (look in the menu area). Once it is on screen, it is ready to use.



The DropDown lists displays in the Menu area.

Point and Click

Point and click to insert any item contained in any DropDown List. Depending upon the state of the «NewDoc/Insert» toggle at the left side of the DropDown List Controls Panel the selected clause will be inserted at the cursor point in the open document, or a brand-new document will be created using the selected item. (If «NewDoc», the document will be identical in every respect to the original, but it will just be a copy. You can tell because the name of the document will be something like "Document 2".



DropDown list 'dropped down'. Select a clause.

«NewDoc/Insert» toggle

In the upper left corner of the DropDown List section is a button that is initially set to «NewDoc». This means that the next document you click on in your List will be inserted as a new document. (You can

hover over it to see its action.) The NewDoc will be an identical copy of the original in every respect. The only way to tell that it is not the original is to check out its name, which will be 'Document 2', 'Document 3' or the like. This prevents the possibility of an accident overwrite of the original.

You can toggle the button to «**Insert**» by simply clicking on it (and then back to «**NewDoc**»). Inserted text typically takes on the style characteristics of the receiving document

«NewDoc/Insert» override

If you always want (or never want) a particular behavior for a DropDown list, you can set an override such that a selection will always be NewDoc or always Insert, regardless of the toggle. To set the override, click the «Other Settings and Actions» item at the foot of the list. Then select the desired override in the resulting screen.

<Alt-G> recall

If you happen to know the name of the document that resides in one of your DropDown Lists, you can type the name and press <Alt-G> and Pathagoras will find and insert it. This quite literally makes hundreds of terms (or however many you have in your combined DropDown Lists) instantly available. (If you cannot remember the names, you can print out lists and keep the full or edited down lists at your side.

(BETA) You can direct Pathagoras to search for the document's assigned **subject** instead of its name. But you must set a toggle found in the 'expanded' DropDown List options screen (click the blue right pointing arrow at the top right of the screen.). Check the box that says 'Search by Subject.'

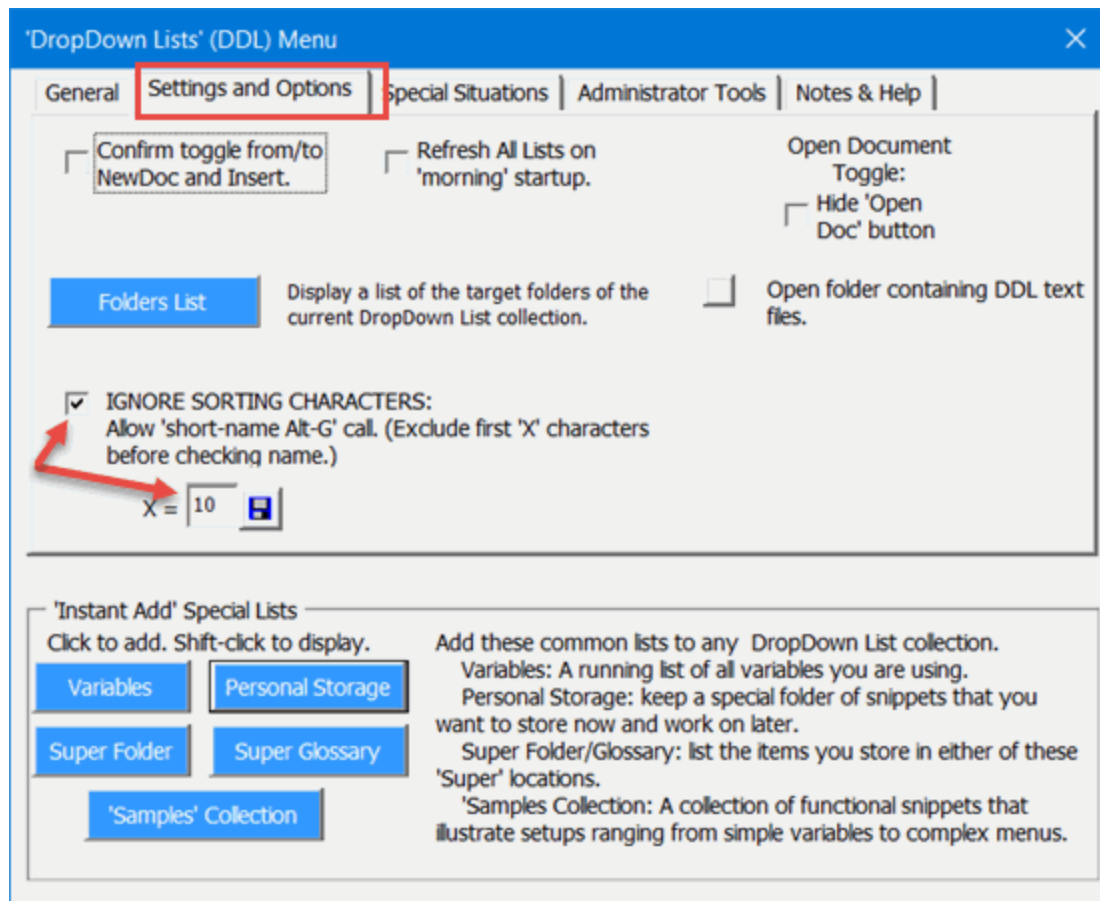
Sorting Characters and <Alt-G>

Many firms use sorting characters in front of the substantive document name so that the documents can be listed in a desired alphanumeric order. Sometimes the sorters are a simple digit or two: "01 Preamble"; "02 Family". Sometimes there are 'Dewey Decimal' style: "002.045A.001 Preamble"; "002.045A.002 Family".

The screenshot shows the 'DropDown List Options' dialog box with the 'GENERAL' tab selected. The 'Folder Name' is set to 'C:\Users\Admin\Documents\Pathagoras Transfer\DEMOTATA\GENERAL\'. The 'List Settings' section includes options for 'Show Sub-folders', 'Show: Name' (selected), 'Sort By: Name' (selected), and checkboxes for 'After selection, return cursor to top of document.' (checked), 'After selection, reset List to top.' (unchecked), and 'Remove formatting before inserting' (unchecked). The 'Override Tray Defaults' section is highlighted with a red box and contains four checkboxes: 'Always New Doc' (unchecked), 'Always Insert' (checked), 'Always Process' (unchecked), and 'Always Suppress' (unchecked). Below this is the 'Styles template used for assembly:' field with a dropdown menu. The 'Default Mask/Intake form:' and 'Prefix associated with this list:' fields are also present. The 'When inserting text:' section has two radio buttons: 'Keep source formatting.' (selected) and 'Adopt destination formatting.' (unchecked). The 'List Actions' section at the bottom contains buttons for 'Refresh List', 'Repoint/Replace', 'Printout List', 'Open Folder', 'Name/Subj Editor', 'Display as Clause Selection Screen' (highlighted in blue), 'Delete List' (highlighted in red), and a 'Close' button. At the very bottom, the text 'GENERAL (*Document*) ShowNamesSortByNameToDocTopNoFolders' is displayed.

NewDoc / Insert Override

Regardless of the nature or length, Pathagoras allows you to exclude the first X characters from a clause search, enabling you to type just the substantive text and <Alt-G> to recall the term from the DropDown. The 'X' characters to exclude setting is found at "**Pathagoras Features | DropDown Lists | DDL Menu | Settings and Options**".



When set, you can type 'Preamble' <Alt-G> even tho' full name of Document may be "001.990.10 Preamble"

OTHER COMMENTS:

- Since you can have up to 10 Lists presented simultaneously, it can be quite easy to assemble complex documents from a wide variety of sources. Drop down a list, select an item. Drop down the same list or another list, select an item, and so on. You can mix images and charts with regular text.
- A DropDown List ("DDL") is a document assembly tool. It presumes that you are building or adding to a new document. Only a copy (and never the original) of the selected item is inserted into the active document. But all formatting from the original is maintained.
- Pathagoras doesn't care whether a DropDown List contains links to complete documents, to individual clauses, to pictures, spreadsheets, PDF files, or whatever. If Word has the capacity to insert a particular type of file, it can be fed into a DropDown list.

- A third toggle exists in the <<**Insert/NewDoc**>> rotation. The command is "Insert Name". When selected, the name of the document, enclosed within << and >> markers, will be inserted. In addition to making it easy to copy the name of a document into your current document (instead of having to copy it from other source) it allows you to create Clause Sets of multiple documents. Clause Sets are discussed at this link. The third toggle is 'off' by default. It can be switched on in **Pathagoras Features | All Settings | DropDown Lists**.



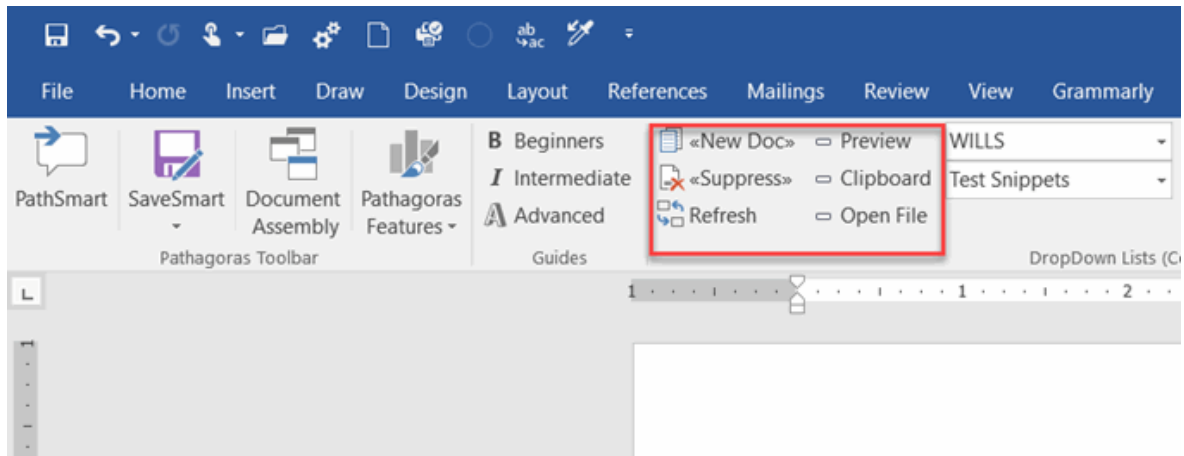
Sometimes you want to make a document available to your end users, but don't want those documents displayed in the list. You can place those clauses in a subfolder called "Hidden" beneath any folder associated with a DropDown List. When Pathagoras searches for a clause in a DropDown List, it will automatically look in the parent folder, and if not there found, in a sub-folder called 'Hidden'. (This feature might be used when you have placed one or more <<[document calls](#)>> into a source document. Pathagoras can still find the called text, but it is not listed in the primary dropdown.

9.6 DropDown Control Panel

The DropDown List Panel houses the various DropDown Lists along with the control buttons that allow you to personalize the various Lists to your specific needs.

Note: Before the first list is created, there is only a single button labeled "DropDown Lists". Only after you have created your first list does the panel expand to display the remaining elements.

At the left side of the DropDown List panel are 6 toggle buttons that control how the DropDown Lists (or items you call from a DropDown List) will behave.



«NewDoc/Insert/Insert Name» Toggle

At the upper left of the DropDown List panel, you should see a button showing either «**NewDoc**» or «**Insert**» or «**Insert Name**». The button's title indicates the expected action. The button is a toggle. Click it once to select the 'other' choice(s).

- «**New Doc**»: a new document will be created, and the selected clause inserted as the first element in the new document; or
- «**Insert**»: the next selection you make from any list will be inserted into the current document at the current cursor location or
- «**Insert Name**»: the next selection you make from any list will transfer the actual name of the selected document into the current screen at the current cursor location. You will be given

the option to insert the document's short name or its full name (with the full address to its folder location). **Note:** This toggle is optional, and turned off by default. Activate it via **Pathagoras Features | Utilities/Settings | All Settings | DropDown Lists**

«Process/Suppress» Toggle:

As you point-and-click in text, Pathagoras default action will be to 'process' any <<*Optional*...>>, <<*Options*...>> and <<*Repeat*>> blocks and to call in any Clause Sets that reside in the inserted text. Automatic processing, however, may on occasion prove inconvenient, especially if you are testing certain actions and do not want the options text 'touched,' or if you want to call in several terms and you want to delay processing until all are present.

Click the Process/Suppress toggle as needed to control whether processing occurs automatically.

The button's title ("Suppress" in the above example) indicates the expected action.

- **«Process»:** Any 'optional' or 'repeat' text, and any other text brought in from the source document within <<double angle brackets>> will be processed.
- **«Suppress»:** Any 'optional' or 'repeat' text, and any other text brought in from the source document within <<double angle brackets>> will be processed, but rather will remain in the assembled document intact and untouched.

The button is a toggle. Click it once to select the 'other' choice.

See Also: Suppress Processing

Refresh: This button should be needed only in the rare situation where you have created a new DropDown List, but Pathagoras has failed to display it. (If you know that a List is not up to date, click the «Refresh» button near the bottom of the List.)

Preview: The classic action when you click on an element in a DropDown List is that it will insert a copy of the selected text at the desired location (into a new document or into an existing document, depending upon the status of the NewDoc/Insert toggle discussed above). But sometimes you want to be able to *preview* the document before committing to it. When 'On' (indicated by a magnifying glass), Pathagoras will display the content of the document you select next in a preview window, along with any comments and usage tips associated with the document. Additional action buttons allow you to insert the text (if it is what you wanted) or to move on to preview another clause. (As noted, the Preview action will show any text that has been placed in the Comments section of the document's Properties section. You should consider adding Comments to your documents to help the end user understand the usage rules for the clause. The raw text is nice to see before committing to a selection, but well worded Comments could be more valuable to the end-user. Click here to read more on Document Comments.)

Clipboard: When 'On' (indicated by a clipboard), Pathagoras will send the text of the next item you select from a DropDown List into your system's clipboard. (It's the same action as highlighting text and pressing 'Copy'.) From there you can paste it into the current document, any other document, or any other program (just like any text in your clipboard can be used).

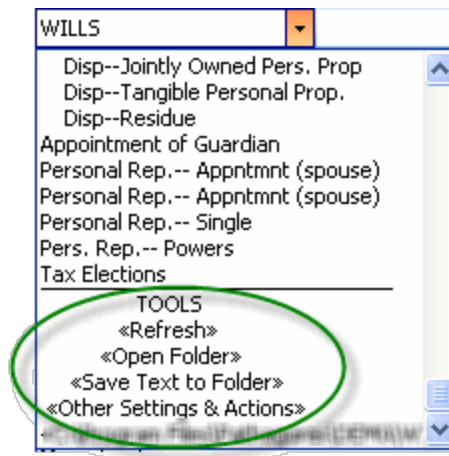
Open File: When 'On' (indicated by the open file icon), Pathagoras will open the original of the next item you select from a DropDown List. (The 'preferred' document assembly action is to display a

copy of the text, never the original. But when only the original will do, such as when you want to edit the original text, the Open File toggle can be a real time saver. Just remember that you have the original document open.) (System administrators can 'hide' this Open File toggle. If you do not see it, but want it, ask your system administrator for access.)

9.7 DropDown 'Other' Settings

At the foot of each DropDown List is a series of options that make each list even more useful and flexible.

Here is what you will see when you scroll to the bottom of a DropDown List:



The 'Below the Line' features, Initial Display
 (The very last line, blurred in the sample above, is the full path name
 of the folder or glossary to which the List points.
 Use it as a quick reference to locate the source of the List items.)

- **«Refresh»:** If you have added or deleted items to the folder to which the DropDown List points, you should 'refresh' the List to bring it up to date. It takes just a second or two.
- **«Open Folder»:** Pathagoras knows where everything is. So, if you want to view the contents of the folder to which the DropDown List points, don't navigate to it. Just click this item and you will be taken directly to it. Open documents, edit them, re-save them. All from this one line.

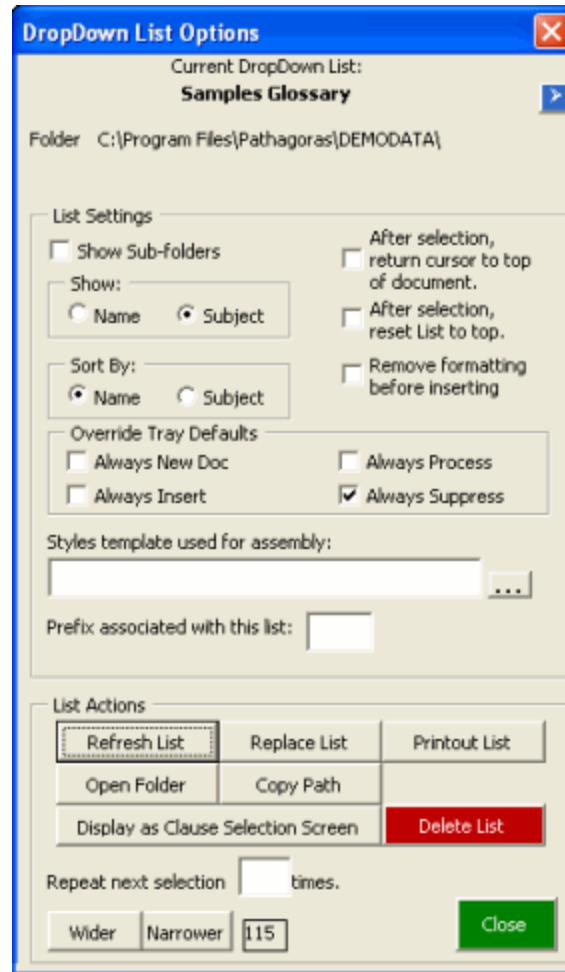
New in 2020.1: Open the target folder by typing the positional name of the List followed by <Alt-G> (for 'g'et). The top tray numbers are 1 thru 5, and the second tray 6 thru 10 (regardless of whether all positions in tray 1 are used, the first list in tray 2 is '6'). So to call the 3rd List in tray 1, type 'ddl<Alt-G>' (not case sensitive). To call the second list in Tray 2, type 'ddl7<Alt-G>'.

<Open Folder> This is also a great general purpose navigation tool. In addition to being able to go directly to the target folder of the DropDown List, you can start what might otherwise be a cumbersome navigation process by using this as a shortcut.

- **«Save Text to Folder»:** Let's say that you have created (or copied from another source) a section of text (or a complete document) and that you want to save into the folder to which the DropDown List points. Simply click this entry and in no time at all, you can save that text. No navigation. (If the DropDown List points to a glossary, this item will read «Save Text to Glossary». Same idea. Same ease of use.)

See [Adding Content to DropDown List](#)^[144].

- **«Other Settings & Actions».** When clicked, it displays an extensive list of options that control the look and feel of that List.



The 'Other Settings & Actions' Display

List Settings.

- **Show Sub-folders:** Turn the Tree Service 'on' and the DropDown List display links to the sub-folders beneath the parent. (Sub-folders are displayed at the top of the list between curly-braces.) Click on a sub-folder and the list will be re-drawn, containing the files of the selected sub-folder. Any sub-sub-folders will be displayed at this level as well, along with an "{..{Up} }" entry so that you can return to top of the tree.

➡ The folder display possibilities here are quite literally endless. By strategically assigning parent folders, you could conceivably access every folder and sub-folder without ever leaving your editing screen. All navigation as envisioned by Windows could be eliminated.


- Display Names/ Display Subjects: Click as appropriate.
- Sort by Name/Sort by Subject: Click as appropriate.
- List Reset after Insert / No Reset after Insert: This simply indicates what shows after you click on a clause in the list and the list is 'retracted'. If 'Reset' is chosen, the title of the list displays. If 'No Reset' is chosen, the selected item displays, making it a bit easier to (perhaps) choose an item further down in the list.
- Return Cursor to Top of Document / Leave Cursor at End:
 - ✓ Choose the former if you want the program to reset itself to the top of the document (so that you can easily review the document from top to bottom)
 - ✓ Choose the latter if you want the insertion point (and the display) to be at the document bottom so you can easily add additional text.
- Remove formatting: Sometimes you don't want the clause you are about to insert to contain the formatting with which it otherwise has been saved. Here is the way to insert truly unformatted text.

Toggle Overrides.

In the DropDown List area of the editing screen, there are two toggle buttons that control whether the selection will cause a new document to be created (vs. inserting the selection into the current document and whether the <<Options/Optional/Repeat>> blocks (if any) within the inserted text will be processed or (temporarily) ignored. If you know that the clauses in a certain DropDown List should always be handled in a particular fashion, you can set that in this section. Regardless of the toggle button setting, the setting in the individual DropDown List will be honored.

Other Settings:

- Assign a Template: Assign a template to any documents created from a term in the list. A template is typically a blank document that contains headers, footers, margins and styles unique to the type of document contained in the list). Once a template is assigned, it to be laid down before any text is inserted in the following instances: (1) the New Doc toggle has been selected and (2) when the item called from the list is the very first item on an otherwise blank page. (Irrespective of the assigned template, if you call in a term into an otherwise blank document that itself has headers and footers, the headers and footers of the recalled document will display.)
 - **Refresh List:** If you have added more items to the folder or glossary, refresh the List to include the new additions. It takes just a second or two.
 - **Replace List:** Place another, completely different, folder or book in place of this DropDown List.
 - **Open Folder:** Don't navigate to the folder to see its full contents. Just click this item and you will be taken directly to it. Open documents, edit them, re-save them. All from this one line.

- **Copy Path:** Places the name of the DropDown List's folder into your clipboard memory. Handy when you want to navigate to the folder. (But don't forget about the 'Open Folder' button.)
 - **Display as Clause Selection Screen:** Create a Clause Selection Screen from the entries in the DropDown List. Very helpful when you want to quickly select and assemble multiple items in the list. Imagine—document assembly of image files. Or Word Perfect® files!
-  If you want to create a sub-set of this list, display the list as a Clause Selection Screen. Then select just the clauses you want to appear in your DropDown List. Choose the Create DropDown List option and press Next. Instantly, you have a shorter DropDown list containing a hand-picked selection of terms.
- **Create Printouts:** Need a hard copy of the contents of the DropDown list? Click this entry to transfer the information to a Word document for editing/printing.
 - **Delete:** Delete the DropDown List from the display.

9.8 DDL Variables List

DropDown Lists provide two-click access to a variety of items: documents, clauses, text snippets (a/k/a building blocks), Excel data, PowerPoint presentations, image files, etc. The end-product is a signable letter, contract, pleading, Will, etc. The target audience is typically the client or customer.

But 'source-document authors' (here we refer to the author of the original or 'source text' used to create the client documents) can also benefit from DropDowns. Several authoring tools can be presented via DropDown Lists. We discuss in this section the DropDown Variables List. This List gives you two-click access to the variables you have adopted as office standards for your forms. It can dramatically speed up the process of 'Pathagorizing' your documents.

Creating the DropDown Variables List

Two methods are available:

1. Via the Instant Database Screen"

- Display the Instant Database screen and select a mask or a client/customer's record that contains the variables you want to add to the Variables List.
- Click 'Power Tools' (the red button) and then click the "Print/Transfer/Export" button. If a List does not already exist, Pathagoras will ask for the location (row and slot) where the List should be placed.
- If a list already exists, you will also have the option of replacing the existing List with just the variables in the mask or record you have called, or to add any new variables to the existing List.

2. Via the 'Collections' Menu (ribbon)

- Click the Pathagoras tab and locate the 'Collections' section toward the right of the ribbon.
- Click the 'Menu' button.
- Select 'Variables' toward the bottom of the new screen. Pathagoras will add a new DropDown List called (appropriately enough) 'Variables.' (If a list of variables already exists

from your having followed these steps in an earlier setting, those variables will populate the new list. Otherwise, Pathagoras will create a new list of variables. It will insert [Client Name], [Client Address], and [Client City, ST ZIP] as a starter list. These variables can be kept or deleted following the Editing procedures below.

Now, insert Variables in 2 clicks!

- **Insert single variable:** Use like any other DropDown List. Just make sure that the cursor is at the location where you wish the variable to be inserted. Point. Click.
- **Global replacements:** Let say you have on your editing screen a document that you want to 'Pathagorize.' Currently it has the name 'John Doe' in several locations throughout the document. You want to 'neuter' the document to create an office form.
 1. Highlight one instance of a word or phrase (i.e., John Doe) that you want to convert to a variable.
 2. Click on an entry in the Variables DropDown List that you want John Doe to become.
 3. Pathagoras will ask if you want to replace all other instances of the same text in the document with the selected variable. If you do, click 'Yes.' The replacement of all 'John Does' with the selected variable is done in a split second.
 4. As with all replacements performed by Pathagoras, the program will preserve the style of the text --case, bold, italics, color, etc. -- when replacements are made in the other locations.

Adding Variables to the Variables List

- **Highlight and Add:** A single variable, or a group of variables, that currently reside on you editing screen can be quickly added to your List.
 1. Highlight the variable(s). Click on the Variables list and select the 'Add Variables to List' element at the bottom.
 2. To add more than one variable at a time, type a list of the variables in a stack down the left side of the page. Highlight the stack and the click the 'Add Variables to List' element at the bottom.
 3. If you are editing a document that contains variables you wish to add, but they are not stacked, use the 'Via the IDB Screen' method described in the section that immediately follows. See #3 in that section.)
 4. That's it. Your new variables have been added to the bottom.
- **Via the Instant Database (IDB) screen:**
 1. Display the Instant Database screen. (You can press Alt-D to call up the IDB screen).
 2. Select an existing record or any mask from the lists in the upper right or upper left drop downs.
 3. As an alternative to using an existing record, you can click the <Scan> button to place the variables in the currently open document into the IDB screen.

4. Regardless of how you populated the IDB screen, click the red 'Power Tools' button.
5. Click the 'Insert into DropDown List' button. Choose whether you want to replace the existing List, or add any new variables to the List.
6. All done.

- **Manual Add:**

Pathagoras allows you easy access to the file that contains the Variables. Using the steps outlined below under "Editing the Variables List" you can manually insert (remove, rename, etc.) variables at your pleasure.

Editing the Variables List.

- You can easily access the file for editing in these two ways:
 1. **Via the Variables List:** Click the 'Open List for Editing' element near the bottom of the List itself.
 2. **Via the Collections menu:** Shift click on the 'Variables' button.

Pathagoras will open the document called Variables.txt as a Notepad (not Word) file. (It is just easier to edit a simple file such a Variables.txt with this editor.) Edit as appropriate and then save and close the file.

If you need a more powerful editor (i.e., Word) to edit the file (because let's say you want to search for and replace text in a large file, or you want to alphabetize your entries), you can easily copy all or part of the content from the Notepad screen and paste it into Word . Edit as desired. Copy and paste the results back into the Notepad screen, save and close.

Don't forget to click «Refresh» in the Variables List.

(You can also navigate to and edit the Variables.txt file manually. As suggested above, we recommend that you edit the list with Notepad, Microsoft's simple text editor.)

Techies: The Variables List is physically stored in a file called 'Variables.txt' located in the same folder that your other Instant Database records are stored. By default, the location is:

32 bit computers: c:\program files (x86)\Pathagoras\IDBS

64 bit computers: c:\program files\Pathagoras\IDBS

The actual current location is displayed when you press Alt-D to display the IDB screen, It's at the bottom of the screen. (You can double click the text box displaying the path to open it.)

Currently there is only one Variables List available. If you wish to categorize the List (i.e., separate your Estate Planning variables from your Litigation Variables, etc.), you can do so by editing and rearranging the contents of Variables.txt using any of the editing techniques discussed above. You can insert Category captions (perhaps in ALL CAPS). You can even indent the variables within each category for a more visually pleasing display.

E.g.,
 GENERAL
 [Client Name]
 [Client Address]
 ESTATE PLANNING
 [Child@1 Name]
 [Child@2 Name]
 REAL ESTATE
 [Grantor]

Refreshing the List

If you know you have added a variable (or several) to the List, but it doesn't appear when you drop it down, click the «Refresh» entry near the bottom of the List. This tells Pathagoras to reread the Variables.txt file.



*If you find yourself using your Variables List frequently, you should elevate the List to your Quick Access Toolbar (QAT). That way you will have immediate access to your variables without having to first click on the Pathagoras tab. To add the List to the QAT, right click on the List and select 'Add to Quick Access Toolbar.' Note: If you have placed your Variables DropDown in more than one Collection, make sure that the List is in the same relative location in each. If you change Collections, the QAT will display the DropDown that occupies the same position as the List from which you created the QAT entry.



!GroupNames!

To help insure consistent spelling of your !GroupNames! throughout your documents, you can append to the bottom of your Variables DropDown a collection of !GroupNames!. Just display a document containing your favorite !groups!. Drop down your Variables List and select "Add Document's !Groupnames! to List.

9.9 DDL *Alias* Lists

DropDown Lists typically point to documents. Word documents in particular. But other sections described how you can point DropDown Lists to Excel spreadsheets, images, Word Perfect documents and others.

You can also point your DropDown Lists to an *Alias* list. An *alias* is a single word or phrase that represents longer list of possible choices. Think "States" representing the 50 United States. 'States' is the alias. The list comprises the names of the individual States.

A DropDown List can refer to any of your existing *Aliases*. Click here for steps on how to create a DropDown List from an existing Alias list. .

You can insert any term from the List simply by dropping down the list and clicking the item. If you want to select multiple items from the list in a single action, first click the 'Create Multi-select Display' ' option at the bottom of the List. With Multi-select activated, you can build sentences, and numbered or bulleted lists.

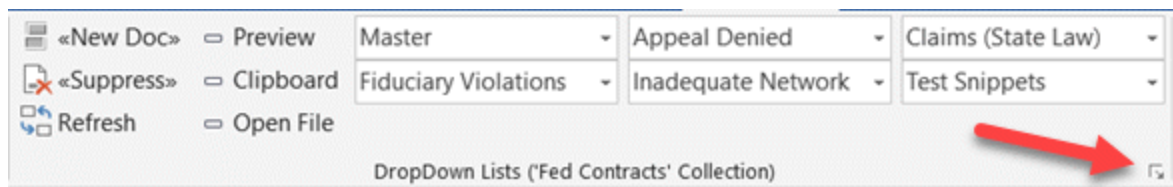
It's one more way that Pathagoras lets you build documents from just about any source. See ['Sentence Assembly'](#)¹³⁰ for more information.

9.10 Community DropDown Lists

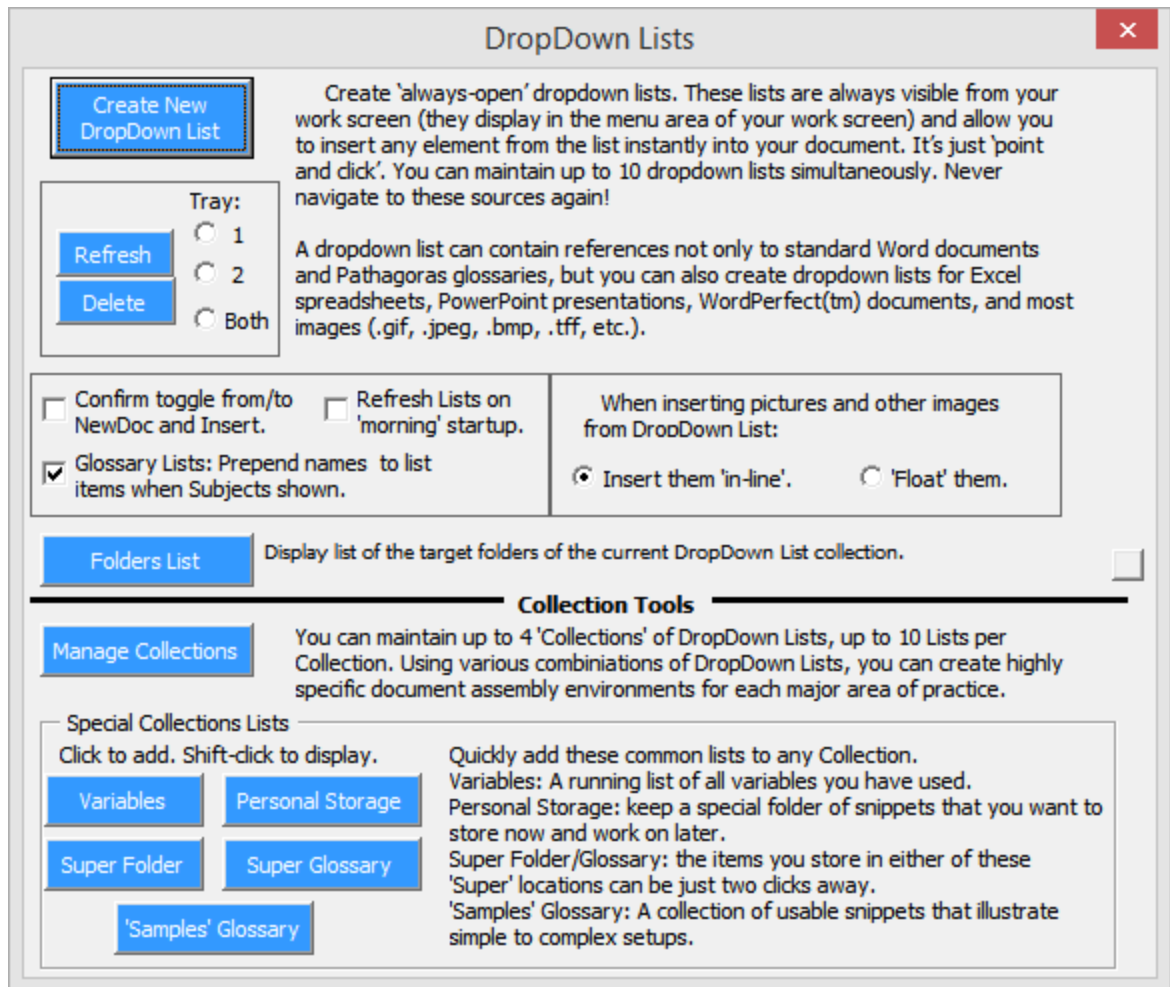
Pathagoras offer several other 'special' lists that you may want to add to one or more (or all) of your collections. We call these 'Community DropDown Lists.' Their content is more general in nature (cover sheets, signature blocks, notary jurats, pleading styles, and the like.) Because of their general content, it is appropriate that you consider including them in multiple Collections. Pathagoras makes it easy to do so. These community lists include:

- You **Super Folder** and **Super Glossary**
- **Samples Glossary**
- **Your Variables.** (if displayed you can insert any variable that you have save in the List into your document in just two clicks. No more guessing how you spelled the variable. The previous page is dedicated to the DropDown Variables list.)
- **Personal Storage**

You can manually navigate to any of these lists, but Pathagoras has made access to them much easier via the Collections Menu (visible in the Pathagoras toolbar area). Click the Menu element.



Click Menu to display the DropDown List Menu, shown below



The Special Collections Lists are shown at the bottom.

Super Folder: The SuperFolder is a folder typically containing your 'general' documents . It and its sibling SuperGlossary, below, allow you quick, no navigation access to the content you store there. If you have assigned a SuperFolder, it's a good idea (but not mandatory) to reference it with a DropDown List. (If you have not previously assigned your Super Books, Pathagoras will give you that opportunity. To learn more about the benefits of **SuperBooks**, click the link.)

Super Glossary: The SuperGlossary is a glossary typically containing 'general' clauses. It and its sibling SuperFolder, above, allow you quick, no navigation access to the content you store there. If you have assigned a SuperGlossary, it's a good idea (but not mandatory) to reference it with a DropDown List. (If you have not previously assigned your Super Books, Pathagoras will give you that opportunity. To learn more about the benefits of **SuperBooks**, click the link.)

DropDown Variables ¹⁵⁷: This is discussed fully in another section of this Manual.

Personal Storage: This list points to a Windows folder into which you can temporarily place documents, clauses, building blocks and text snippets that you wish to preserve, but don't quite know where the ultimately belong. So, save them now and process them later. Consider this just a temporary storage location.

9.11 Debugging the List

There are two main reasons why the entries in the DropDown List may not appear as you expect them to:

- **Issue:** "I know that a file is in the folder (or clause is in the glossary) to which the List points. But that file or clause does not appear in the DropDown List. Why not?"

➤ **Solution:** The List just needs to be **refreshed**. Click the «Refresh» entry near the bottom of the DropDown List

- **Issue:** "My DropDown List is set to display the subjects of the terms in my glossary, and I have assigned subjects to all of my terms. However, for several entries in the List, the term '(no subject)' appears. Why?"

➤ **Solution:** This issue has been reported only when a 'glossary' is the object of the DropDown List. The cause is that there is an extra line between the Subject (the blue text in the glossary just above the actual term text) and the beginning of the actual term text. (For the glossary to be properly read, there can be no lines between the red name, the blue subject and the beginning of the actual text of the term.)

You can fix this by displaying the glossary and *manually* removing the extra line.

--Or--

You can run Pathagoras' '*Structure Checker*' against the glossary. This is the better alternative since it will not only fix the problem at hand, but it will check the entire glossary structure for you.

Here are the steps to run the Structure Checker:

1. Display the glossary. (The quickest way is to click the «Open Glossary» item from the DropDown List or Open Glossary from the Document Assembly 'Libraries & Books' screen.)
2. With the glossary 'on display,' click the Pathagoras dropdown features list and select "Authoring/Editing Creation Tools".
3. Click "Glossary Tools".
4. Click the <Structure & Integrity> button.

9.12 WordPerfect, PDF and Image Assembly

Word Perfect Assembly

Pathagoras is capable of 'document assembly' of Word Perfect® files assigned to a DropDown List. We do want you to understand however, that the resulting document is a Word, not a Word Perfect document.

Individual files can be 'clicked in' one at a time using just the List.

However, true assembly using these Word Perfect documents can be accomplished by clicking the <<Clause Sel. Screen>> entry toward the bottom of the DropDown List. Pathagoras will

transfer all elements of the List into a standard Clause Selection Screen, with the selectable items will be listed in the left panel.

Just like in assembly of standard Word documents, you would select some or all of the Word Perfect documents and move them to the right panel. When done, press **Next>>**.

See Also:

Word Perfect®

Folder to Glossary

Word Perfect® is a registered trademark of the
Corel Software Corporation

PDF Assembly

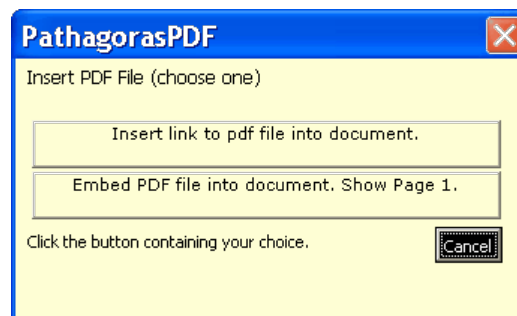
As mentioned in other pages, Pathagoras allows a DropDown List to be populated with PDF files as well as document files. Therefore, Pathagoras is capable of 'PDF assembly' as well as document assembly.

There are a few limitations of which you need to be aware when it comes to working with PDFs.

When you click on a PDF from a DropDown List, you will be presented the option of inserting a link to the file or the actual file. See below figure.

Link to PDF: A link is no more than that. Make sure that the recipient of the link has access to the folder to which the link points.

Embed PDF: The other choice is to actually embed the PDF in the Word document. This makes for true document portability. It can also make for a very large file, as the Word document now contains not only the Word text but the entirety of the PDF record. You can embed ('assemble') an unlimited number of PDF files into your Word document.



Viewing the embedded PDF: When you click the embed option, an 'image' of the first page of the PDF file will be placed into your Word document. To view the actual file, simply double click on the image. Your PDF viewer will activate and show all the PDFs.

Printing the PDF: Word cannot directly print an embedded PDF. If you printed a document into which you embedded a PDF file, you will get all the Word text and the first page (the one that you

see when you embed the file). To print the entire contents of the PDF, you must view it (see above paragraph) and print using your PDF viewer.

Converting Word files to PDF: You cannot convert your Word document into a PDF and keep the embedded PDFs intact. If you convert your Word document containing embedded PDF files into a new PDF file, only the *image* of the first page of each embedded file will convert. If each PDF file is itself one page, this is not bad news, but it is little different from image assembly. If you need to transmit the Word document with the embedded PDFs as a single entity, don't convert it to PDF.

See also:

[Creating a DropDown List \(Free Hand\)](#)¹⁴¹

IMAGE ASSEMBLY:

As mentioned in other pages, Pathagoras allows a DropDown List to be populated with image files as well as document files. Therefore, Pathagoras is capable of 'image assembly' as well as document assembly.

Images can be 'clicked in' one at a time, perfect for situations where an image is inserted and a description is then typed after the entry.

True image assembly can be accomplished by clicking the <<Clause Sel. Screen>> entry toward the bottom of the DropDown List. Pathagoras will transfer all elements of the List into a standard Clause Selection Screen, with the selectable items listed in the left panel. Select some or all of the images and press **Next>>**.

9.13 Double Duty

DropDown Lists pull double (if not triple and quadruple) duty.

- DropDown Lists can be used to select complete documents for processing. Typically the New Doc toggle will be set.
- DropDown Lists can be used to insert that 'one last clause' that makes the document complete. Typically the 'Insert' toggle will be set.
- By clicking the <<Other Settings and Actions>> element at the bottom of the List, you can convert the List into a fully functional Clause Selection Screen, allowing you a whole new range of clause/document assembly options. [Click this link](#)¹⁵⁴ to read more.
- DropDown Lists can function as 'text expanders.' The name of any entry in a List can be typed to the editing screen. Press Alt-G and Pathagoras instantly finds the clause and inserts it into your document.
- **Document disassembler.** The **Save text to Folder** element makes it incredibly easy (not to mention incredibly fast) to highlight and save snippets (words, sentences, paragraphs and even large swaths of highly formatted text) into the DropDown List's folder. Highlight your text, click 'Save text to Folder' from the desired DropDown List, give the term a name, and it's done.

- **Clause Set** builder. If your DropDown List contains clauses that can be used to build a complete document, you can 'pre-build' one or more documents using only clause references. Make sure the 'Insert Name' toggle is set as you are clicking in the individual clauses. Save the clause set as a document in the DropDown List folder. Next time you need the 'simple will' (or whatever you created), just click in the clause set. Pathagoras handles the rest.

The Pathagoras System

Excel Spreadsheets as Datasource

Part



X

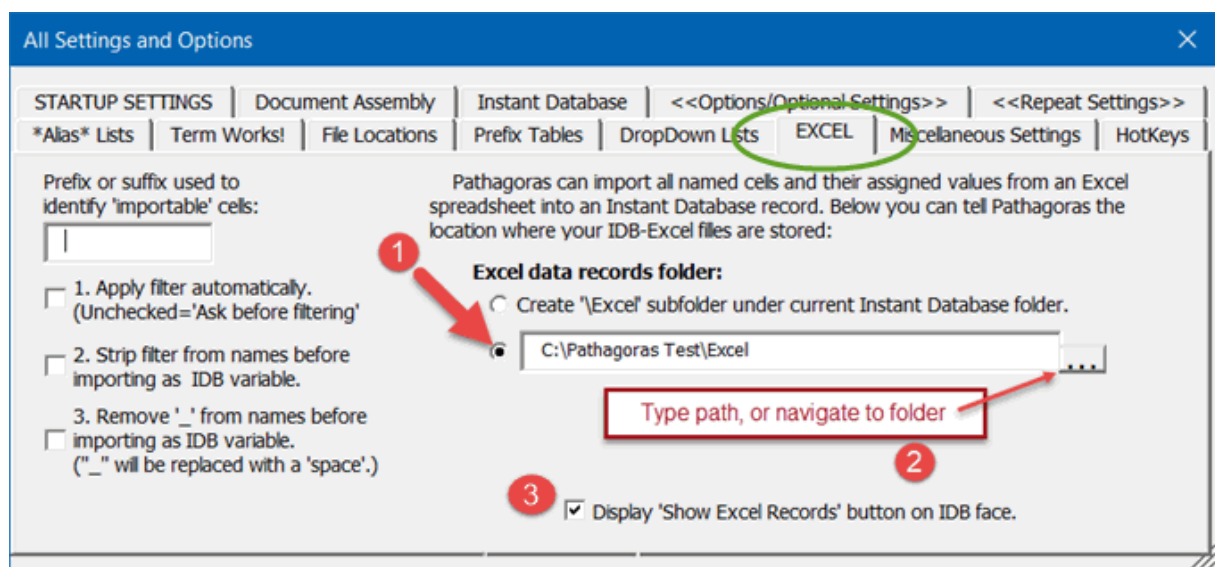
10 Excel Spreadsheets as Datasource

An Excel spreadsheet can be a data source for Instant Database data. Beginning with version 2016.2, it can also be a primary datasource in its own right.

Location of Files: The 'default' location for Excel files as an IDB datasource is a folder called 'Excel' directly beneath IDB records folder. (To determine the current [IDB Files Folder](#)^[106], click link.)

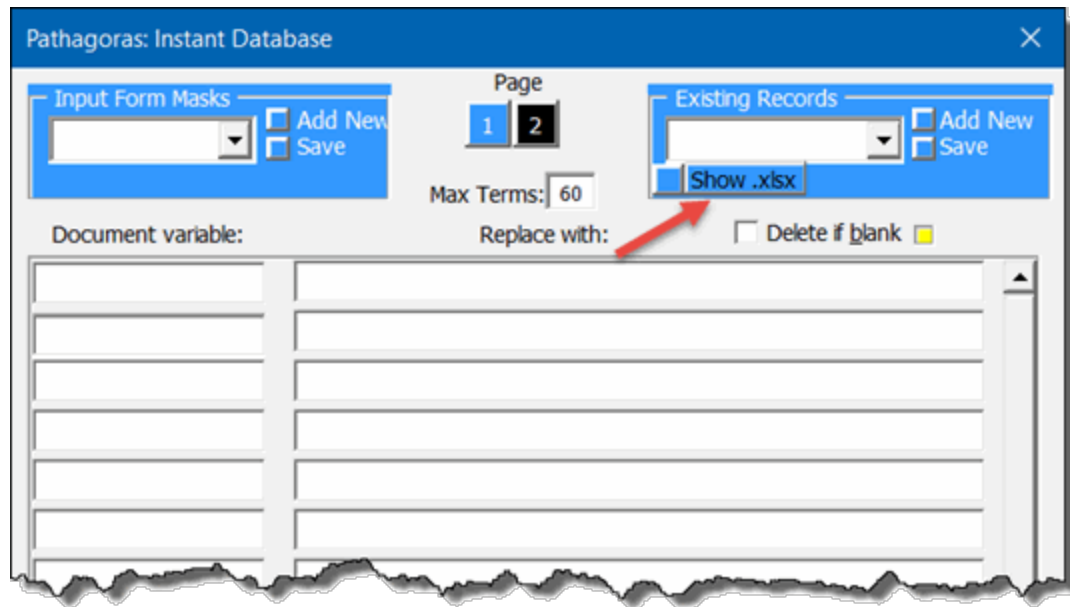
However, you can place your Excel files folder in any convenient location of your choosing. To set that location, follow these steps:

navigate to the **Excel** tab in the All Settings window (Pathagoras Features | Utilities & Settings | All Settings). Here is the screen you will see:

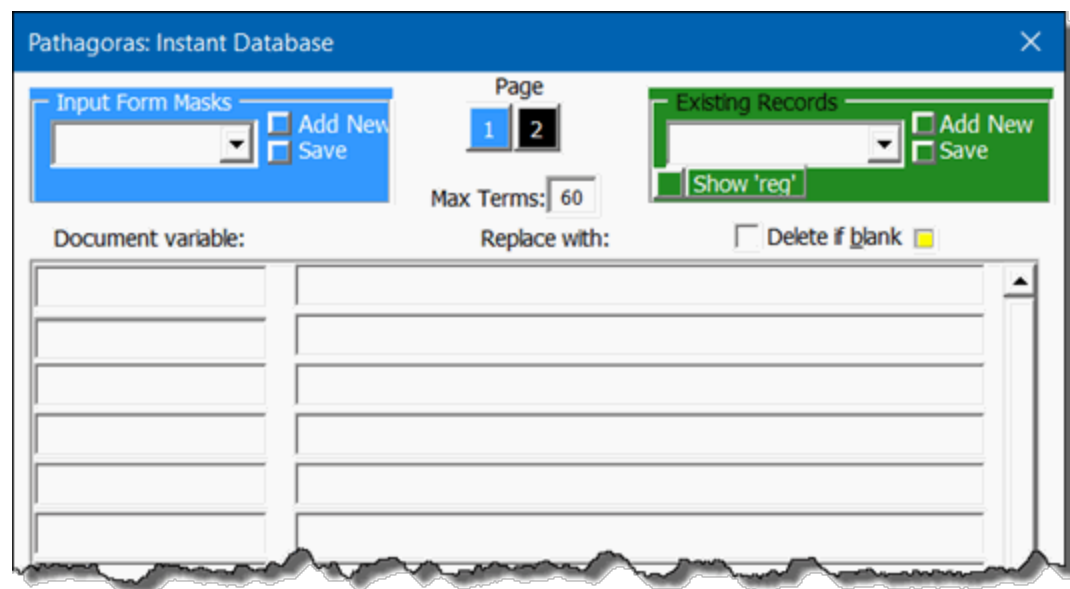


When the Instant Database Screen is displayed, the 'regular' records are gathered and displayed in the Existing Records dropdown. But click the Show .xlsx button and the section is transformed

....



... the Existing records box turns green (Excel's 'brand' color) and



... only the records in the Excel folder selected in step 1 are show. Select it and Pathagoras accomplishes two more steps. (1) Pathagoras determines whether the record is a simple 2 column record or a more elaborate spreadsheet with named fields. If the later, Pathagoras presents the names of the Named Fields in the left column and the values that reside in those named fields in the right column.

Any new variables in the document that did not exist in the .xlsx file will be appended to the bottom of the new record (just like with 'regular' records).

Notes:

- There is no 'write back' capability to the source Excel file. (You can, of course, save the data to a new IDB record. Indeed, this is the better choice since you likely will be adding variables to the record over time.)

Rules:

- If the style of the spreadsheet you are reading is simple (that is, two columns, with the variable name in column A and the variable value in Column B, or two rows, row 1 being variables and row 2 their corresponding values), just about an value can be used in either column. (There may be exceptions, but none that we know about at this writing.) You don't even need the brackets indicating that column A (or row one) is a [variable]. Pathagoras will add brackets automatically.
- However, if your spreadsheet contains named cells, a few special rules apply. Named cells allow you greater flexibility in the layout of the spreadsheet, but Microsoft imposes some rules that Pathagoras -- and you - must observe.
- Named Cells issues:
 - If a spreadsheet has even one named cell, Pathagoras will see it as a 'named cell spreadsheet.' That means if you intended a 'simple' (two column) structure of data and have 100 separate rows of data, but accidentally have created, or left in, a single named cell, that single cell will trump the 100 rows of data. So, if you are getting strange results, check the spreadsheet for named cells and delete them. (In later versions of Pathagoras, you will be able to tell Pathagoras to ignore named cells.)
 - The 'name' of a named cell in Excel can contain only letter (a to z) and numbers, and an underscore. No spaces or special characters. If your Named cells contain the '!' mark signifying a group name or the '@' sign used by Pathagoras for incrementing variables (e.g., [Child@1 Name]; [Child@1 Name DOB], you will discover that Excel (not Pathagoras) replaces those symbols with an underscore. (Spaces will be replaced with underscores as well).
 - Ancillary to the above bullet, you should not try to use mimic in Excel Pathagoras' 'fancy' Instant Database tools (e.g., multiple choice variables and !groups!) in your spreadsheets. They won't work there, and the conversion back to Word may not be pretty. Use Excel tools to perform Excel calculations, but feed the results back to Word/Pathagoras using simple variable names.
 - EXCEPTION: As part of the Excel to IDB conversion process, Pathagoras will look for a pattern of 'text underscore #' (e.g., "Child_1_Name"). If detected, Pathagoras will replace the underscore preceding the number with a '@' sign. So while the name of the Excel cell might have to be [Child_1_Name], Pathagoras will convert it to [Child@1 Name]. So don't hesitate to assign your incrementing variables to named Excel cells.
 - Pathagoras will, by default, alphabetize the results when it reads a spreadsheet of named cells.

The Pathagoras System

**Conditional Text
(Options/Optional)**

Part



XI

11 Conditional Text (Options/Optional)

Conditional Text refers to those blocks of text that remain in, or are deleted from, the template, depending upon certain conditions. Those conditions can be an answer to a simple 'Yes/No' prompt that you create, or as the result of a more sophisticated and layered question and answer sequence.

Consistent with Pathagoras' plain text approach, conditional text blocks are created using straight from the keyboard characters typed at the location of the target text. The conditional text remains as you typed them, where you typed them. If and when you have to edit them, you know where they are. If you have teach them to others, or copy and paste them as sample text for other projects, they are where you expect them to be. If you want to add another layer of cascading or branching logic, you know where to start.

Pathagoras uses two types of conditional text blocks. They are both created with, and denoted by, the boundary markers “<<” and “>>”. (*Pathagoras provides a 'simplified' version of both of these blocks. They are discussed starting at [this link](#)^[202]*)

1. **“Optional” text:** You should think of 'optional' text as 'take it or delete it' text. At document assembly time, the program will highlight the text, pause and ask “Do you want to keep this?”. The user need only respond “Yes” or “No” to tell Pathagoras whether the text block should be retained. 'Yes,' the boundary marks are stripped and the target text remains. If 'No,' the text block (including boundary markers) is deleted and the extra space is closed up. If automatic paragraph numbering is in place, the below numbers are appropriately decremented.

Creating 'Optional' text block: Simply type boundary markers '<<' and '>>' around the text (the amount of text does not matter). Type command `*Optional*` just inside the opening boundary marker.

```
<<*Optional*The widgets you have ordered are not currently in stock. We will
ship them as soon as possible. If we have not shipped within 5 days of this date,
you will have the option to cancel the order.>>
```

2. **“Options” text:** This type allows the user to select among several choices -- i.e., options. At document assembly time, the choices are presented to the user as selectable checkboxes. The user selects one or more of the displayed choices.

Creating 'Options' text block. Simply type boundary markers '<<' and '>>' around the text that comprises all of the options. (The amount of text does not matter. Pathagoras can handle as many choices as you need. The size of each individual option, or the number of paragraphs in a particular option, is likewise unlimited.) Type a forward-slash between each possible choice. Type the command `*Options*` just inside the opening boundary marker.

<<*Options*Per your request, the widgets will be shipped by Federal Express. We will bill you for the extra cost of shipping./Per your request, we will send the widgets by standard ground transport. This may take 3 to 5 additional days./As per your request, we will hold the widgets for pickup by your courier./The widgets you have ordered are not currently in stock. We will ship them as soon as possible. If we have not shipped within 5 days of this date, you will have the option to cancel the order.>>

When Pathagoras encounters the optional text block (#1 above), it will highlight the text in the document and ask if you want to keep it (Figure 1).

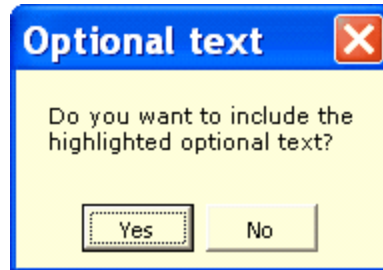


Figure 1 Optional Text dialog.

If you select <Yes>, the boundary markers are removed and the text remains in the document.

If you choose <No>, the entire text block is deleted from the document.

When Pathagoras encounters the options block (#2 above), it will parse out the individual choices and display them onto buttons on a selection screen. (If the text is too long to fit, only the first 200 or so characters of the option will display.) Checkboxes are shown at the left of each choice so that you can choose more than one option, if desired. If you want just a single choice, click on the 'bar' containing that choice.

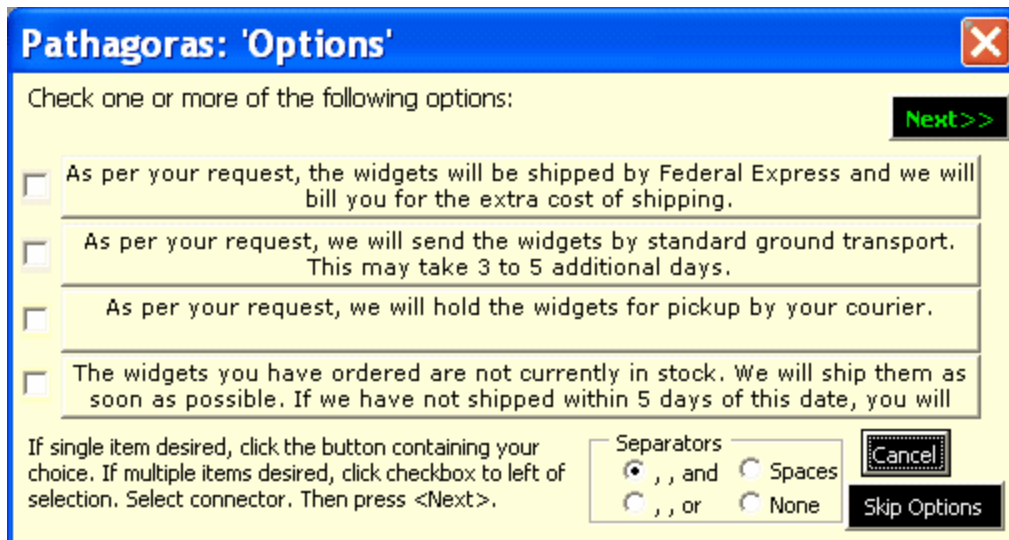


Figure 2 Options block dialog.

Note that the actual option text is provided (subject to space constraints.)

 **Notes:**

- Creating an <<*Options*>> or <<*Optional*>> text block is all 'plain text'. No fields, no codes. Just remember that the 'administrative' section of an Options or Optional text block must end with an asterisk.
- The above examples are the 'purest' forms of conditional text. Pathagoras allows more robust approaches to creating and displaying options and optional text. That is done by adding 'arguments,' 'prompts' and 'hover-over' text which can be used to restrict or explain choices, to add appropriate punctuation to the final selections and otherwise lead to a better final draft. These tools are also intended to enhance the experience for the end user. These 'robust' additions are discussed and illustrated in the next sections.
- When you have nine or fewer *Options* in an <<*Options*>> block, the display is as shown below:

Pathagoras: Options

Select one or more of the following options:

Group: Flavors

☐ chocolate

☐ strawberry

☐ vanilla crème

☐ Rocky Road

If single item desired, click the button containing your choice. If multiple items desired, click checkbox to left of selection. Select connector. Then press Next>>.

If there are 10 or more options, the display is a more compact form that looks like this:

Multiple Choice Display

☐ Alabama

☒ Alaska

☐ Arkansas

☐ Arizona

☐ California

☒ Colorado

☐ Connecticut

☐ Delaware

☐ Florida

☐ Georgia

☐ Hawaii

☐ Idaho

☐ Illinois

Separators

☐ ,, and ☐ Spaces ☐ 'Enters'

☐ ,, or ☐ None ☐ Other

- **Where to place the slash?** Typically, the placement of the slash between choices is easy. When words or sentences are being separated, the slash goes after the last character of the

preceding choice. But when the choices are 'paragraphs long', a little experimentation and trial and error might be needed. The logic built into Pathagoras as to how to process the keeping/elimination of selected/discarded text is complex. Factors such as paragraph indentions, centering, automatic paragraph numbering, and other style considerations makes it impossible to articulate a hard and fast rule regarding where to place the slash. Just perform a couple of test runs using the [Process button](#)^[197] to see if the result is what you expect. Be sure to test all options.

While <<*Options/Optional*>> text blocks are easy to create manually following the instructions provided above, you can easily build them automatically using the [Create Options Assistant](#)^[194].

Any of the text blocks shown above can be copied and pasted into any Word document for testing and experimentation. To test the action of any <<*Options/Optional*>> routine, place the cursor in your Word document immediately to the right of the closing bracket and press Alt-G.

Click here: www.pathagoras.com/sample for a collection of *Options/Optional* text blocks samples

which can be copied into a Word document and tested.

See also:

[Create <<*Options*>> Assistant](#)^[194]

[Administrative Text](#)^[198]

[Single \(Radio button\) vs. Multiple Selections \(Checkboxes\)](#)^[179]

Structure Checker

Suppress processing of <<*text blocks*>>

[*Aliases*](#)^[114]

11.1 'Optional' Text with Prompts'

The action of Pathagoras when it encounters **pure** <<*Optional*>> text is to highlight the entire text block and overlay the question with "Do you want to keep this highlighted text? Yes or No." For short blocks of text that can be quickly read and understood, this probably will work fine. But when the text blocks get longer, or if the context is not apparent, you will probably want to add a more meaningful question to which the user can respond 'Yes' or 'No'. Add such a prompt immediately after the word *Optional*. You must end of the prompt with an "*" to not the close of the administrative text.

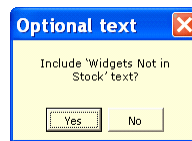
Structure:

<<*Optional*(prompt)* . . .>>

Example:

```
<<*Optional*Include 'Widgets Not in Stock' text?*The widgets you have ordered
are not currently in stock. We will ship them as soon as possible. If we have not
shipped within 5 days of this date, you will have the option to cancel the order.>>
```

When encountered during document assembly, the prompt after the command and before the asterisk that closes the administrative text is presented instead of the default question shown in the previous page.



Optional Text (displaying user provided question.)

Notes:

- Make sure that the prompt is answerable 'Yes' or 'No,' with the 'Yes' answer resulting in keeping the target text, and 'No' resulting in deletion of the text. (Note--the colors in the above and the following examples are for illustration purposes only.)
- If you decide that the question needs reworking, simply open the source document and edit the question. Since there is no hidden coding or fields or links to ancillary programs, there is never an impediment to perfecting your source documents.
- **Administrative Text:** The text to the *left* of the last asterisk in an <<*Options*>> or <<*Optional*>> text block is referred to as the 'administrative text', as distinguished from the actual choices found to the *right* of that last asterisk.) We talk about 'administrative text' in other sections of this Manual.
- **Substantive Text:** The text to the *right* of the 'administrative text' in an <<*Options*>> block is the text that will remain (if chosen) in the final document.
 - The '**Negative Optional**': It is possible to provide a 'false' value result to optional text. The 'false' value will be selected if you choose "No" or "False" to the question presented. To provide the 'false' part, simple type **/NEGOPT** (no quotes, color not important, CAPS mandatory) at the end of the Optional text block and type the 'negative' language you want to be included in the document if the user provides a "No" or "False".

```
<<*Optional*!Apples!* An apple a day keeps the doctor
away./NEGOPTThose wormy apples didn't keep my doctor
away.>>
More examples on how to best use NEGOPT can be found at this page.
```

See also:

[Create <<*Options*>> Assistant](#)

Structure Checker

/NEGOPT

11.2 'Options' Text with Prompts

The default action of Pathagoras when it encounters <<*Options*>> text is to present each choice (up to the first 200 character) to the user for selection. If the options are short and otherwise self-explanatory, this should not be a problem. But if the text of each option is lengthy, this can make for a very busy, wordy and confusing screen.

In the latter case, with just a little more typing, you can provide succinct and more meaningful 'prompts' to the user. These prompts will appear in place of the actual choice as the text on the buttons. Properly written, the user can respond more accurately and quickly to the prompts in making a selection.

To add prompts to your Options text, list them immediately after the word *Options*. Separate each prompt with a slash (making sure that you have as many prompts as you have text options). As always, and close the list of prompts -- part of the administrative text section -- with an "*"

Structure:

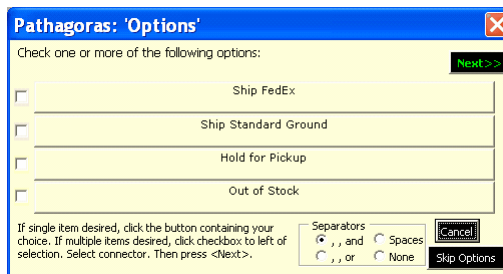
```
<<*Options*(prompt1/prompt2)(etc.)* . . .>>
```

Example:

```
<<*Options*Ship FedEx/Ship Standard Ground/Hold for Pickup/Out of
Stock*As per your request, the widgets will be shipped by Federal Express and
we will bill you for the extra cost of shipping./As per your request, we will send the
widgets by standard ground transport. This may take 3 to 5 additional days./As
per your request, we will hold the widgets for pickup by your courier./The widgets
you have ordered are not currently in stock. We will ship them as soon as
possible. If we have not shipped within 5 days of this date, you will have the
option to cancel the order.>>
```

When encountered during document assembly, the prompts between the asterisks are presented instead of the actual series of options. (See example of 'no prompts' on the previous page).

The above example will yield the result shown in the figure below:



Options block, with descriptions provided.

NOTES -- "and/or", "he/she" (etc.) and fractions: By default, Pathagoras uses the forward slash '/' character to determine the boundaries of each option in an <<*Options*>> block. However, if you use the forward slash as 'normal text' within an <<*Options*>> text block (e.g., a fraction like '1/2' or a non-variable phrase such as 'he/she' or 'and/or'), you must use "/OR" as the separator within that block. Consider the following.

```
<<*Options*Chocolate/Vanilla/Mixed: 1/2 chocolate and 1/2 vanilla.>>
```

In the above example, Pathagoras cannot tell where the 'real' choices end. It will see the above block as 5 choices ("Chocolate", "Vanilla", "Mixed: 1", "2 chocolate and 1" and "2 vanilla") and not the intended 3.

As noted above, the workaround requires only that you add the word "OR" to the slash (to create "/OR") to denote the choices. So the above block should read.

```
<<*Options*Chocolate/ORVanilla/ORMixed: 1/2 chocolate and 1/2 vanilla.>>
```

A few more notes:

- "/OR" must be in CAPS.
- If you use shorthand text prompts (described at the very top of this page) to depict each option, you must use "/OR" in the prompt as well as in the <<*Options*>> block body.
- "/OR" is only required when a natural slash exists within the same <<*Options*>> block. It is not needed otherwise.
- While the 'rule' is that you must have as many prompts as you have options, the exception to the rule is that you can have a single prompt as 'introduction text' to an Options block. Therefore, this is a 'legal' construct:

```
<<*Options*Choose your favorite flavor:*Chocolate/Vanilla/Strawberry.>>
```
- <<*Options*>> text *in tables*: <<*Options*>> text blocks work well within a single cell of a table, but you cannot cross table cells. That is, the opening "<<" marker cannot be in one cell and the closing ">>" marker in another. If you must 'cross' cells, consider 'paragraph assembly.' Each block of text comprising an option is saved as a single clause. You can select the appropriate clause from either the Clause Selection Screen (along with others that can make up the entire document) or from a [DropDown List](#)^[140].
- The options to the right of the 'administrative text' can be 'end text' choices (such as chocolate and vanilla) or they can be references to other text, including calls to documents, if enclosed within 'double angle brackets'. See [Calls to Other Documents](#)^[307] for more information and examples.



The %OR and ^OR (PercentOR and CaretOR) Separators: To determine the proper scope of a particular option within a multiple choice section of a document (choices being indicated by a slash (/) or slashOR (/OR'), Pathagoras would convert slashes in nested Options and variables text within the parent block, replacing those slashes with other character sets. (Otherwise the slashes separating nested blocks would be seen as separators for the parent.) For complex and heavily nested documents, however, the process of converting 'in' and then back 'out' added a significant amount of processing time. The judicious use of a %OR and ^OR dividers (which we have dubbed 'super-separators') to denote choices at the 'top-most' levels will avoid Pathagoras taking time to convert nested text. Read more at this link

See also:

[Create <<*Options*>> Assistant](#)^[194]

Structure Checker

[*Aliases*](#)^[114]

[Single \(Radio button\) vs. Multiple Selections \(Check-boxes\)](#)^[179]

[Calls to Other Documents](#) ³⁰⁷

11.3 Hover-over Text



Hover-Over text: Sometimes even a prompt cannot perfectly communicate what you intent to the end user. You can provide up to 256 more characters of 'hover-over' text for each prompt. This hover-over text will appear when the user moves the cursor over each prompt. It will display only during the time the cursor is 'hovering over' the selection.

To add hover-over text, simple type two plus signs ('++') followed by the hover-over text at the end of each option. Make sure the hover-over text is within the administrative section of Options block (i.e., before the third asterisk).

Example:

Effective Date: <<*Options*!TypePOA!Springing++Use when delegation of power effective only upon disability of grantor/Immediate++Effective when signed and delivered*This Power of Attorney shall become effective only upon my disability as certified by any of my attending physicians./This Power of Attorney shall be effective immediately upon its delivery to my Attorney in Fact.>>

See also: [Hover-over AskTables](#) ²⁶⁰

11.4 'Radio' argument

If you look at the result in the previous examples, you will see check-boxes next to each option. The user can choose one or more than one of the available choices.

However, there are times where you need to restrict the user to making a single choice. In these cases, use the '(radio)' argument. At processing time, single select 'radio buttons' are presented to the users instead of checkboxes.

To designate 'radio buttons,' simply add the text “(radio)” adjacent to the word 'Options'. Parentheses required. Since the argument is part of the command, the closing "*" will appear to the right of the argument.)

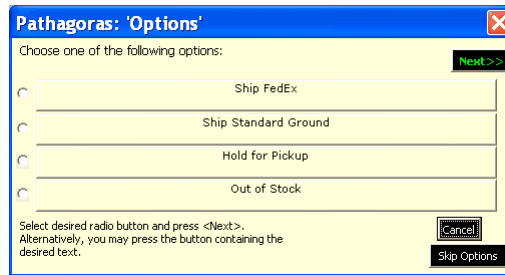
Structure:

```
<<*Options(radio)* . . .
```

Example:

```
<<*Options(radio)*Ship FedEx/Ship Standard Ground/Hold for Pickup/Out of Stock*As per your request, the widgets will be shipped by Federal Express and we will bill you for the extra cost of shipping./As per your request, we will send the widgets by standard ground transport. This may take 3 to 5 additional days./As per your request, we will hold the widgets for pickup by your courier./The widgets you have ordered are not currently in stock. We will ship them as soon as possible. If we have not shipped within 5 days of this date, you will have the option to cancel the order.>>
```

This is the result:



Options presentation with 'radio' buttons.
Only one of the choices provided can be selected.

11.5 'and' and 'or' arguments

While you can use them with 'stand-alone' <<*Options* blocks, they are designed for use with <<*Options* calls triggered by an <<*AskOptions* call when the individual <<*Options* blocks require different responses, one or more with 'and' or 'or' connectors.

```
<<*AskOptions*!colors!red/blue/green/yellow*>>
```

```
<<*Options(and)*!colors!*apples/blueberries/grapes/bananas>>.
```

When processed, the user selects one or more from the AskOptions prompts. Pathagoras then returns the appropriate fruit with commas appropriately inserted and an 'and' between the penultimate and final choice.

11.6 . . .to call documents

The text with an option or optional block can be 'real' text (such as was shown in the above examples). This is the rather 'classic' approach. But text can be *references* to other documents that you want Pathagoras to find and insert, following a true 'document assembly' pattern where substantive text is stored in individual documents and called in to a new document when needed. This approach minimizes the need to find all documents that use a particular block of text when a correction is needed. Change just the one document, and the all future documents that use that text are automatically up to date.

By way of example, an <<*Options*>> block can read like this:

```
<<*Options*As per your request, the widgets will be shipped by
Federal Express. We will bill you for the extra cost of shipping./As
per your request, we will send the widgets by standard ground
transport. This may take 3 to 5 additional days./As per your request,
we will hold the widgets for pickup by your courier./The widgets you
have ordered are not currently in stock. We will ship them as soon as
possible. If we have not shipped within 5 days of this date, you will
have the option to cancel the order.>>
```

Using document calls, the the same block can read like this:

```
<<*Options*<<shp101>>/<<shp102>>/<<shp103>>>>
```


where shp101,102 and 103 point to text stored as a document in a folder or glossary called shp101.docx, shp102.docx and shp103.docx. The 'double angle brackets' signal Pathagoras to make the call to the appropriate document. See ['Document Calls'](#)^[307] for more information and more examples.

More discussion and examples:

As stated above, a primary goal of any document assembly program is to let a single block of text be used in multiple documents. The simplest example might be letterhead text that might change with the addition of a partner, or the change of the firms address or phone number. It could also be text that contains a statutory reference or quotation that changes after each legislative session. Or it can be a clause that now is present in 25 documents but in which you recently discovered a typo. Regardless, if that text needs to be changed, and you are using `<<document calls>>`, only the source text needs to be changed. All future documents that need that text will necessarily contain the most up to date version.

An `<<*Options* . . .>>` block is a perfect tool for inserting external document text into the document you are currently building. Instead of 'classic' text, as the option between the slashes, use a Document Call. Documents Calls are discussed in greater detail at [this link](#)^[302] but we offer a short lesson here:

If the name of a known document is typed (or otherwise brought) onto your editing screen and is enclosed within double angle brackets ('DAB'), Pathagoras will locate that document and insert it's text in place of the Call. (The double angle brackets are not chevrons. Rather, they are plain text double angle brackets. E.g. `<<my document>>`).

So if Pathagoras sees `<<my document>>` anywhere in the document under construction, it will locate and place the desired text onto the screen precisely where you asked for it. How fast: blink of an eye fast!


`<<*Options* . . .>>` blocks. can, as discussed in previous pages, present a wide variety of text choices, from single words and sentences to multiple paragraph and pages. But they can also offer to you a list of Document Calls. This feature opens up a wide range of document assembly possibilities. By properly constructing the `<<*Options . . .>>` block, you can pose a simple question to the end user which, when answered, calls in an entire document.

For example, let's assume the following `<<*Options*>>` text block resides in the source clause of a letter being written to a purchaser of goods. The purpose of the clause is advise the letter's recipient what the shipping costs would be in various situations. We also assume that 'Full Charge.doc', '20pct discount.doc', '50 pct discount.doc' and 'Free shipping.doc' (which you see below as the document names enclosed within the double angle brackets) are existing documents in the our system.

```
<<*Options(radio)*Order Amount:$0 -
$100/$101-$200/$201-$500/$501 + *<<Full charges>>/<<20pct
discount>>/<<50 pct discount>>/<<Free shipping>>>>
```

When the above text block is encountered during Pathagoras' top-to-bottom 'processing' of the document, Pathagoras will present onto a pop-up form the prompt text: '\$0 - \$100' '\$101-\$200' '\$201-\$500' and '\$501 +' for selection. Make the appropriate selection. Based on that selection,

the appropriate <<document name>> value is returned to the screen (albeit only briefly). When Pathagoras rescans the document, it will encounter the DAB text, signalling Pathagoras to find the document and insert its text in place of the <<document name>> entry.

 A 'document' can be any thing you want. It can be a simple signature block saved out as a document. It can be a multi-page order form. It can be a 100 page trust instrument. If Word can handle it, so can Pathagoras.

'Fill-in' as a choice: If there are situations where you need greater flexibility, you can 'fill-in' with a DAB document name. E.g., <<*Options(**radio**)*Order Amount:\$0 - \$100/\$101-\$200/\$201-\$500/\$501 +/user-choice*<<Full charges>>/<<20pcnt discount>>/<<50 pcnt discount>>/<<Free shipping>>/<<Fill-in>>>>. If you select the Fill-in choice, you will be prompted for a document name. Type just the name (no extension, no path) and Pathagoras will find it for you if the document is in your [hunt path](#)³⁰⁴.

11.7 'Cumulative' argument

Pathagoras provides an alternative approach to using the <<Repeat>> command. We call it the Cumulative argument for Options text. When a choice is made, it 'accumulates' all lower choices. More below.

With 'Repeats', you start with a section of text, perhaps including variables. When the document is processed, you will be asking Pathagoras to duplicate the text X number of times, and appropriately increment any variables within the duplicated text.

With the 'cumulative' argument to an Options command, you pre-list the greatest number of options reasonably foreseeable in the options block. When the document is processed, you will be asked for the 'highest' element you want. The selected and all lower positioned elements are retained. Any remaining Options above the selected position are deleted.

"Cumulative"

The '**cumulative**' argument to the <<*Options*>> command is a great alternative to the [Repeats](#)²²⁴ command and to other alternatives which involve listing of sequentially larger and longer choices.

With '**cumulative**', you would prepare a list of choices. Each choice reflecting an increasingly higher count of a specific variable (plus surrounding text). Think 1 child, 2 children, 3, etc.). However, each choice anticipates including all of the lower choice. If you select item 4 from the list, all selections up to and including 4, will remain in the document. All items 'above' the choice are deleted. Proper commas and connectors can be inserted with the '**cumulative,and**' modifier so that the result is a proper 'sentence.'

Example: Let's say you are creating a Will. As you are composing the 'family' clause, you pre-list the maximum number of children you are likely to need:

```
<<*Options(cumulative)*One child/Two children/Three children/Four children/Five
children/Six children*
```

```
[Child1Name] born [Child1DOB]/
```

```
[Child2Name] born [Child2DOB]/
```

```
[Child3Name], born [Child3DOB]/
```

```
[Child4Name], born [Child4DOB]/
```


```
[Child5Name], born [Child5DOB]/
```

```
[Child6Name], born [Child6DOB]>>
```

"Cumulative, and" (used in the more 'prose' settings)

```
<<*Options(cumulative,and)*One child/Two children/Three children/Four children/Five
children/Six children*[Child1Name] born [Child1DOB]/[Child2Name] born [Child2DOB]/
[Child3Name], born [Child3DOB]/[Child4Name], born [Child4DOB]/[Child5Name], born
[Child5DOB]/[Child6Name], born [Child6DOB]>>
```

When the command is processed, and you select (for example) 'Four children', all text blocks up to and including your choice will be inserted. Commas and a final 'and' connector will be inserted since the **(cumulative,and)** modifier is being used.

 **NOTE:** If you have prepared an Interview including <<*AskOptions* . . .>> command for the related !group!, the proper modifier for the interview *Ask* is "(radio)". This is to insure that only one choice can be made when the Interview is presented.

So for the above example, in the Interview section, use

```
<<*AskOptions(radio)*!cdn!One child/Two children/Three children/Four children/Five
children/Six children*>>
```

The body text (which need not include the prompt text) can be simply:

```
<<*Options(cumulative,and)*!cdn!*[Child1Name] born [Child1DOB]/[Child2Name]
born [Child2DOB]/[Child3Name], born [Child3DOB]/[Child4Name], born
[Child4DOB]/[Child5Name], born [Child5DOB]/[Child6Name], born [Child6DOB]>>
```

Tables

The cumulative modifier can be used with tables. Each row of a table is deemed an option. (With tables, no slash is needed between the rows of the table. Slashes are still needed between the prompts. *Prompts are mandatory if a !groupname! value is not used.*

<<*Options(cumulative)*Header/One Child/Two Children/Three Children/Four Children/Five Children*

Child Name	Age	Born in Year
[Child-Name@1]	[Child-Age@1]	[Child-DOB@1]
[Child-Name@2]	[Child-Age@2]	[Child-DOB@2]
[Child-Name@3]	[Child-Age@3]	[Child-DOB@3]
[Child-Name@4]	[Child-Age@4]	[Child-DOB@4]
[Child-Name@5]	[Child-Age@5]	[Child-DOB@5]

>>

Another approach to the above is to place the header outside of the table. Pathagoras will join the header with the body if you have the 'Join Adjacent Tables' feature turned 'on':

Child Name	Age	Born in Year
------------	-----	--------------

<<*Options(cumulative)*!cdn!*

[Child-Name@1]	[Child-Age@1]	[Child-DOB@1]
[Child-Name@2]	[Child-Age@2]	[Child-DOB@2]
[Child-Name@3]	[Child-Age@3]	[Child-DOB@3]
[Child-Name@4]	[Child-Age@4]	[Child-DOB@4]
[Child-Name@5]	[Child-Age@5]	[Child-DOB@5]

>>

In the immediately preceding example, the groupname !cdn! is being used to set the number of rows that will be kept. The example assumes it was set previously, either in a previous <<*Options* or <<*AskOptions*

The 'None' Option

Sometimes you don't want *any* of the options. E.g., 'None' may be the correct choice for the 'How many children?' question. Pathagoras process the "None" choice when it is the only one chosen (by deleting the others), but when a 'higher' choice is made, excludes/excises it from display when anything above it is chosen. So this is an appropriate and functional expansion of an above example:

```
<<*Options(cumulative,and)*!cdn!*None/[Child1Name] born [Child1DOB]/
[Child2Name] born [Child2DOB]/[Child3Name], born [Child3DOB]/[Child4Name],
born [Child4DOB]/[Child5Name], born [Child5DOB]/[Child6Name], born
[Child6DOB]>>
```

A similar approach would be to use prompts, the first one being a prompt that suggest 'none', and the first text selection being just a slash (indicating 'no text'). E.g.,

```
<<*Options(cumulative)*No Accounts/1 Account/2 Accounts/3 Accounts/4
Accounts/5 Accounts*/[Account #1]/[Account #2]/[Account #3]/[Account #4]/
[Account #5]>>
```

Along the same lines, if you use AskOptions in a 'top of document' interview, you might use:

```
<<*AskOptions*!accts!No Accounts/1 Account/2 Accounts/3 Accounts/4 Accounts/5
Accounts*>>
```

and in the body of the document:

```
<<*Options(cumulative)*!accts!*/[Account #1]/[Account #2]/[Account #3]/[Account
#4]/[Account #5]>>
```

Note in the above two examples, you don't use 'none' as an 'in body' choice -- just a slash.

The 'cumulative' argument is only used in body text.

Don't forget the [groupname](#) if you use AskOptions.

Limits: The maximum number of elements you can process with the 'cumulative' argument is 9. Use <<*Repeat* if you need more than 9 elements.

11.8 'Connector' argument

By default, when Pathagoras presents the screen on which you select one or more options, as soon as you select a second option, Pathagoras will present a series of 'connectors' from which you can pick to indicate that you want commas and the word 'and' or 'or' to automatically be inserted between the selections.

```
<<*Options(connector)*Apples/Bananas/Cherries/Dates>>
```

However, if you don't typically want these choices presented, you can change the default in the Pathagoras Features | All Settings | Options/Optional' Uncheck the Display Connectors. Then if you decide you do want those options displayed for a particular variable, just add the 'connector' argument

11.9 Fill-in the blank

Sometimes you want to provide a value for a document without having to use the Instant Database. The Instant Database allows you to provide values for [variables] enclosed within plain text square brackets, and save the variables-to-values pairings in a database that can be repetitively reused.. Pathagoras offers that with the 'fill-in' option of the options tool.


In most cases, when you are using <<*Options* blocks, you are supplying the text from which you wish your users to choose:

```
My favorite fruit is <<*Options*apples/bananas/peaches>>.
```

But occasionally you want to allow the end user (or yourself) an opportunity to provide an alternative value as a response to the Option. Beginning with version 2017.1, Pathagoras lets you do that. You can provide a 'stop' phrase in your list of options. **By default, the stop phrase is 'fill-in'.**


```
My favorite fruit is <<*Options*apples/bananas/peaches/fill-in>>.
```

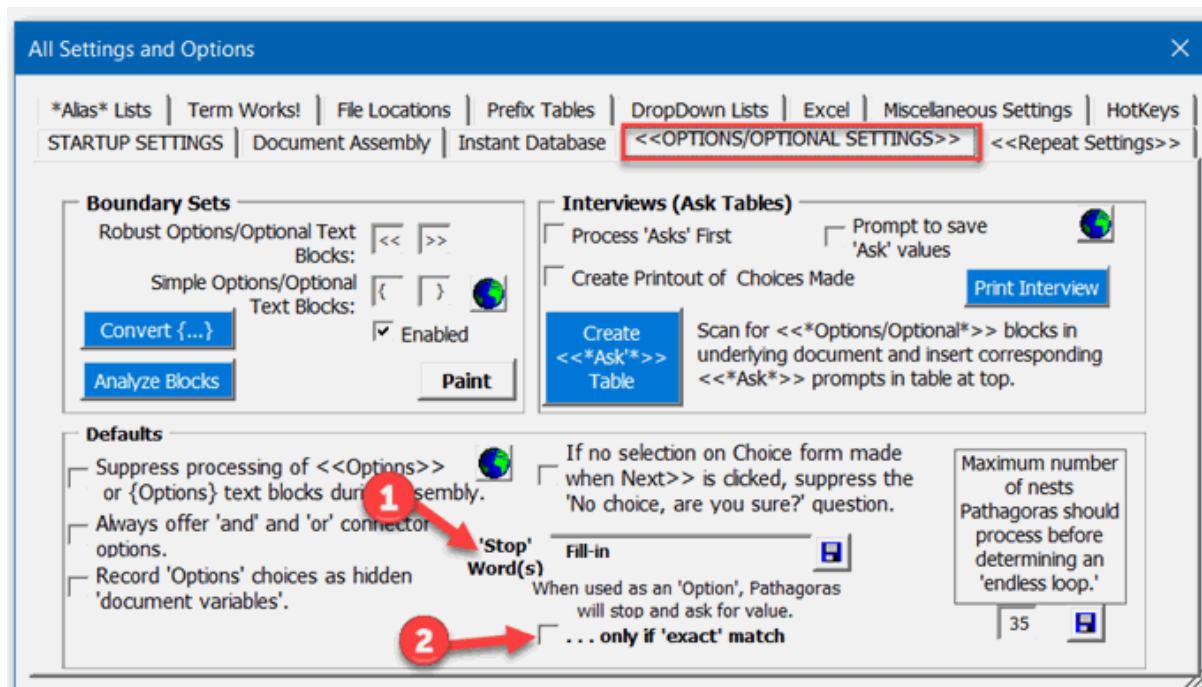
If stop phrase is selected, Pathagoras will pause and present a text box for completion. Provide a value response and press Okay. Pathagoras inserts the value at the point of the Options block (erasing all other options and administrative text in that block).

 The term 'Options' obviously connotes two or more choices. But Pathagoras offers a singular exception (pun intended) for the 'Fill-in the blank' term. If you decide not to pre-place one or more options, this is still a properly constructed options block:

```
My favorite fruit is <<*Options*fill-in name of your favorite fruit>>.
```

The last example above illustrates one additional feature of the 'Fill-in the blank' tool. The wording of the prompt must contain the stop phrase, but need not be just the stop phrase. So you can offer a meaningful prompt to your end users.

 'Fill-in' is the default 'stop' term for generation the action described above. However, you can assign any other term you wish. E.g., "Stop", "Pause" "Insert your answer here". To reset the 'stop' phrase, navigate to Utilities/Settings | All Settings | Options & Optionals tab. Insert a different phrase (replacing 'Fill-in') (1). You can also designate whether the 'stop' phrase must be exact or just must be included somewhere in the Options choice (2).



If the '<<...exact match>>' box remains unchecked, any of the below examples will trigger the 'stop and ask':

My favorite fruit is <<*Options*apples/bananas/peaches/fill-in>>. (Multiple choices & fill-in.)

My favorite fruit is <<*Options*fill-in>>. (No choices. Just fill-in.)

My favorite fruit is <<*Options*apples/bananas/peaches/favorite fruit (fill-in)>>.

My favorite fruit is <<*Options*please fill-in your favorite fruit>>.

If the 'exact match' box is checked, only the first two examples above will trigger the 'stop and ask.'

Notes:

- The fill-in feature is also available for the Simple options construct. {Fill-in} and {!group!Fill-in} are acceptable commands. [See more here.](#)¹⁸⁸
- We stated it above, but it is worth repeating. The 'fill-in' values you provide are inserted in place of the <<*Options*>> block (or the specific option if more than one are selected) in the current document, *but they are not otherwise preserved*. This feature, therefore, is not a substitute for Instant Database. But when you don't intend to save values, and don't want to activate the Instant Database routine, you may find this is a perfect alternative to completing variable type text.

11.10 Fill-in the blank . . .

Sometimes you want to provide a value for a document without having to use the Instant Database. The Instant Database allows you to provide values for [variables] enclosed within plain text square brackets, and save the variables-to-values pairings in a database that can be repetitively reused.. Pathagoras offers that with the 'fill-in' option. The robust version of this feature is discussed [at this link](#).^[186] But the 'simple' option (which may actually be your preference due to its, well, simplicity) is discussed below

When you provide a 'stop' phrase as the value in your simple {optional block}, when that item is encountered, Pathagoras will pause and present a text box for you to fill in a value. Then press Okay and Pathagoras will insert the value at the point of the {simple optional} block (erasing all other options and administrative text in that block).

My favorite fruit is {fill-in}.

Notes:

!Groups! [GroupNames](#)^[136] work fine with the 'Fill-in the blank' feature. Each 'fill-in' of the same group will be completed in the identical fashion.

The 'fill-in' values you provide are inserted in place of the {fill-in} call in the current document. *The value is not otherwise preserved.* This feature, therefore, is not a substitute for Instant Database. But when you don't intend to save values, and don't want to activate the Instant Database routine, you may find this is a perfect alternative to completing variable type text.

This fill-in tool is optimized for Pathagoras robust <<*Options*. . . >> as well. [Click here for more](#).^[186]

11.11 Nesting

<<*Options/Optional*>> Text Blocks: Nesting:

Nesting of <<*Options/Optional*>> text block is allowed. As a practical matter it is not advisable to nest beyond two or three levels. This is not because the system itself is limited. Rather this limit is suggested because it gets confusing to the eye to have nestings too deep.

Here is one example of nesting based on the examples used in prior pages. The nest appears in the last option. (Coloring has been added to the elements of the text block for emphasis. You can also add coloring to your source documents to make editing easier. See [Color Markers](#)^[318]):


```
<<*Options(radio)*Ship FedEx/Ship Standard Ground/Hold for Pickup/Out of
Stock*As per your request, the widgets will be shipped by Federal Express and
we will bill you for the extra cost of shipping./As per your request, we will send
the widgets by standard ground transport. This may take 3 to 5 additional
days./As per your request, we will hold the widgets for pickup by your
courier./The widgets you have ordered are not currently in stock. We will ship
them as soon as possible. If we have not shipped within 5 days of this date, you
will have the option to cancel the order. When available, the widgets will be
shipped by: <<*Options(radio)*Federal Express./Standard ground transport./Per
your request, we will hold the widgets for pickup by your courier.>>>>
```

As you are designing <<*Options/Optional*>> text blocks, please note that the text need not be a solid block (as the above examples were). <<*Options/Optional*>> text blocks can be multi-dimensional, just like Word documents can be. You can have single or multiple, paragraphs, add 'character' to the text (color, emphasis, even pictures). You can add any dimension to make the block meet your specific need. (Just a reminder. The various elements of the text block in the below illustration are 'painted' red and blue. These colors are for emphasis only. Colors are not required See [Color Markers](#)^[318].)

<<*Optional*My favorite foods are:

<<*Options*Vegetables

My favorite vegetable is <<*Options(radio)
*tomatoes/potatoes/squash/beans/none of the above>>./

Fruits

My favorite fruit is <<*Options(radio)*peaches/<<*options(radio)*Macintosh
apples/Golden Delicious apples/Granny Smith apples/plain ole
apples>>/plums/bananas>>./

Grains

My favorite grain is <<*Options(radio)*whole wheat/barley/oats/beer>>.

>>

>>

Note: Pathagoras always works on <<*Options/Optional*>> text blocks from the outside in.

Diversion: Copy and paste the above example into a standard Word document. Trigger the processing by pressing <Alt-G> at the very end of the document. Note the actions of the various elements and option styles used, including the use of non-radio options for the top-most question, and radio options for the others.

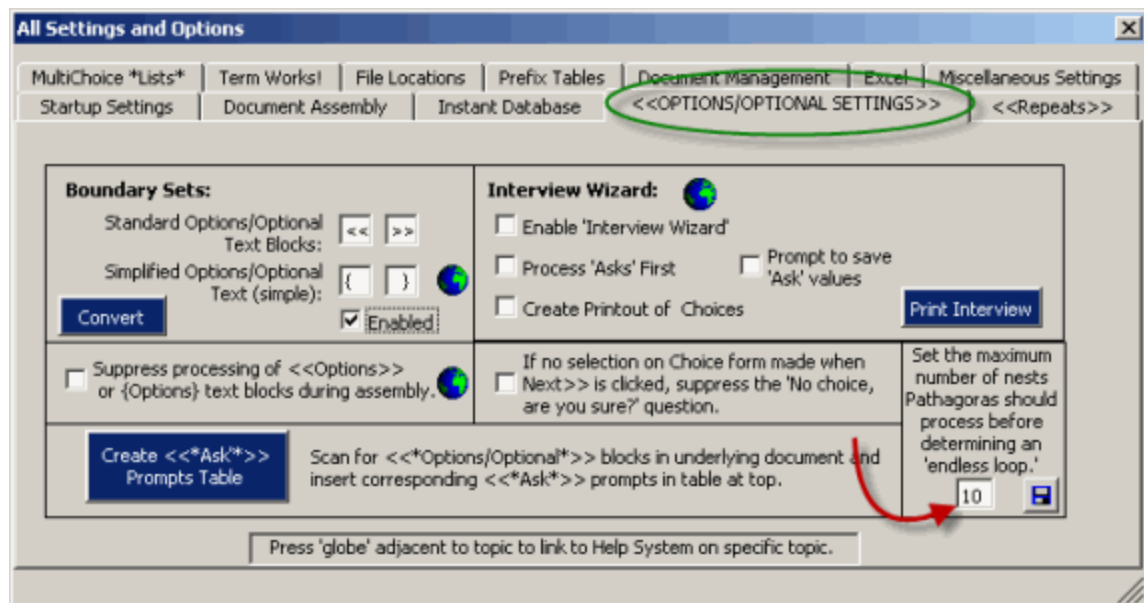


You can nest Options questions to as many levels as you choose. But just because you *can* doesn't mean that you *should*. We suggest no more than two levels of nesting. The problem with nesting is that, the 'deeper' you go, the more difficult the resulting text is to read. Everything in Pathagoras is on-screen, not in hidden fields. This is a plus when you nest once or twice -- the end user doesn't have

much trouble seeing a predicting what will happen if a certain choice is made. But when it gets deeper that, it can be difficult to read. Concentrate on properly constructed questions. That way, you will not have to go very deep at all.

As stated immediately above, there is no 'legal limit' to the number of nests you can create, but Pathagoras needs a cap to tell it when to stop looking for closing brackets. An arbitrary number of 5 nestings was selected. (An error message reporting "Hopeless Imbalance" will appear when the cap is reached. (By 'nesting', we refer both to 'deep' nesting -- where every new nest is a 'grand child' of the previous parent, and sibling nests, where all nestings are child of a single parent, on the same level as each other.)

If you receive the message, and know the structure is correct, you will need to increase the cap. To do so, go to the <<Options/Optional Settings>> tab in the All Settings screen and increase the count to 10 (or more, if needed). (See image below.)



11.12 Processing Order, Delaying Processing

Processing is initiated automatically when a document (or set of documents) is called via the Document Assembly Clause Selection screen or when a document is called from a DropDown List. Processing also begins when the user presses <Alt-P> from the keyboard.

Processing begins when P detects a balanced '<<' and '>>' sets. (We call these 'double angle brackets' or DAB sets). It ends when no more balanced DAB sets can be found. (We say 'balanced' because there can be nested sets of '<<' and '>>' markers. In a properly structured document, only the outer-most set will be 'balanced.'

Once a balanced DAB set is detected, Pathagoras processes the command just inside the opening '<<' (e.g., *Options* or *Optional* or *Repeat*, and any arguments provided).

Processing takes place from top to bottom of the document, from outside to inside. The exceptions are these:

1. If a !groupname! is present in the block being processed, all elements of the group will be processed immediately, regardless of intervening '<<. . . >>' blocks. You can take

advantage of this 'rule,' and set the order of processing in any fashion you desire, by using <<*AskOptions*, <<AskOptional* and <<AskRepeat* blocks at the top of the document.

2. <<*Repeat* commands are processed after *Options* and *Optional* blocks. That is to allow a possible nested <<*Optional*<<*Repeat*. . .>>> block to be processed in case the <<*Repeat* is not kept. An exception to this exception is this: if the <<*Repeat*!group!* . . .>> is called via and <<*AskRepeat*!group!* . . .>> call, it will be processed in the order of appearance in the AskTable.
3. <<Document call>> (i.e., just the name of a document inside of DABs) are processed last. If the document called in by a <<document call>> itself has DAB sets, they are processed in a recursive fashion, and in the order described above.

11.13 "Syntax"

Options/Optional Text:

The anatomy of <<*Options*>> & <<*Optional*>>text blocks.

The setup for <<*Options*/*Optional*>> text blocks is not without its potential complexities. Compared with the setup required by other programs, however, it is undeniably simpler. Because it is plain text, you don't have to fish around for the right field coding and switches. And when you actually see it 'broken down,' it makes even more sense to you.

But precise placement of the various characters that make up the blocks is mandatory. Without the right characters properly placed, Pathagoras will not be able to dissect and process the text. Not to worry too much, however. After the discussion of 'anatomy,' we will show you (1) how to activate a screen which will automate the creation of <<*Options*/*Optional*>> blocks and (2) how to activate a different screen which will check the structure of each <<*Options*/*Optional*>> block in a document.



The coloring that you will see in the samples below is for emphasis only. No coloring is required.



Remember that the 'administrative' section (name, group and prompts, if any) of an <<*Options*>> or <<*Optional*>> text block must end with an asterisk.

Simplest <<*Options*>> block (up to 9 choices separated by slashes):

<<*Options*I have one minor child./I have no children./I have __ minor children.>>

(Notes: Colors are only for emphasis. Without 'prompts' (discussed below), the actual text of each of the options will appear on a selection form during document assembly process. Don't forget the closing brackets.)

<<*Options*>> block with reference to *Aliases*

<<*Options**children*.>>

<<*Options*>> block with "Questions"

<<*Options*Is there one child?/ Are there no children?/Are there two or more children?* I have one minor child to whom I give, devise and bequeath the remainder of my estate./I have no children./I have [number of children] minor children to whom I give, devise and bequeath the remainder of my estate, in equal shares.>>

(Notes: The question — which need not actually be a question — is any text set out between the second and third asterisks. The questions will appear on the selection form during document assembly process.)

<<*Options*>> with groupings:

<<*Options*!Children!* I have one minor child./I have no children./I have __ minor children.>>

<<*Options*!Children!* I give the remainder of my estate to my child./I give the remainder of my estate to the following persons. /I give the remainder of my estate in equal shares to my children, per stirpes.>>

(Notes: The Group name (in this case “!Children!”) is set out between exclamation marks immediately after the second asterisk. An asterisk after the !groupname! closes out the administrative section of the block. Once a selection of the first !group! encountered has been made, all other options blocks within the same !group! will be ‘handled’ in the same fashion.)

<<*Options*>> with groupings and prompts:

<<*Options*!Children!Is there one child?/Are there no children?/Are there two or more children?* I have one minor child./I have no children./I have __ minor children.>>

<<*Options*!Children!* I give the remainder of my estate to my child./I give the remainder of my estate to the following persons./I give the remainder of my estate in equal shares to my children, per stirpes.>>

(Notes: Once the first Options group is encountered and question answered, Pathagoras will automatically process the remaining blocks with the same group name. The selection is made using the same *positions* in the list. Because the processing is done automatically, you do not need to repeat the question for subsequent members of the same group.)

Simplest <<*Optional*>>:

<<*Optional*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>>

<<*Optional*>> with question:

<<*Optional*Is pickup available?*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>>

<<*Optional*>> with groupings and question:

<<*Optional*!pickup!Is pickup available?*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>> (body text . . .body text)

<<*Optional*!pickup!*Be sure bring a crane with you.>>

(Notes: Once the first Optional group is encountered and answered, Pathagoras will automatically process the remaining blocks with the same group name. Because the processing is done automatically, you do not need to repeat the question for subsequent members of the same group.)

Linear summary of the <<*Options*>> block elements. Mandatory elements are noted in red:

<<

Options or *Options(radio)*

!GroupName! (group names allow you to complete all members of the group with the same level selection made in the first member of the group.)

QuestionText/QuestionText/etc* (no opening '"', just a closing one. Either one question per option or one question, period.)

Option1/Option 2/Option Number 3/etc.

>>

Use the '(radio)' suffix within the Options tag if you want to restrict the user to just one of the listed choices. The default is to allow the user to select one *or more* of the up to five possible choices.

Remember that the 'administrative' section of an Options or Optional text block must end with an asterisk.

Linear summary of the <<*Optional*>> block elements. Mandatory elements are noted in red:

<<

Optional

!GroupName! (group names allow you to complete all members of the group with the same level selection made in the first member of the group.)

QuestionText* (followed by an asterisk to close out the administrative section)

blah blah blah (The text that you want to stay or to be deleted.)

>>

See also:

[Create <<*Options*>> Assistant](#)¹⁹⁴

['Administrative Text'](#)¹⁹⁸

Coloring Markers

Structure Checker

11.14 Tables, Rows & <<Options>>

The 'traditional' method of presenting Options text is to separate each choice with slashes.

But sometimes presenting the data in rows and columns makes for a better layout. Pathagoras will interpret the separate rows in a table as individual choices. Example:

Title	Name
Chief Executive Officer	[CEOName]
President	[PresidentName]

<<*Options*!Officers!CEO/President/VP/Secretary/Treasurer/COO/CTO/CFO*

Vice President	[VPName]
Secretary	[SecretaryName]
Treasurer	[TreasurerName]
Chief Operating Officer	[COOName]
Chief Technical Officer	[CTOName]
Chief Financial Officer	[CFOName]

>>

<<*Options(radio)*FedEx/USPS (First Class)/UPS*

Federal Express	The widgets will be shipped by Federal Express. We will bill you for the extra cost of handling and shipping.
United States Postal Service	We will ship the widgets by standard First Class mail. There will be no additional charges.
UPS	We ship your purchase via United Parcel Service

>>

<<*Options*!Entity!Corp/LLC*

<<*Repeat*!Directors!* <hr/> [Director#Name]>>
<<*Options*Owners?/Manager?* <<*Repeat*!Owners!* <hr/> [Owner#Name]>>/
<hr/> Manager >>

>>

11.15 Create <<*Options*>> Assistant

The Create '<<*Options/Optional*>>' text assistant' can help you to properly form the administrative section of each <<*Options/Optional*>> text block. If you are not familiar with how Optional and Options text, be sure to familiarize yourself first with the [<<*Optional*/*Options*>> text blocks](#)¹⁷² section of this Manual.

To activate the Assistant, click the “Create <<*Options/Optional>> Ass't” entry in the Pathagoras drop down features list. (You can also press <Alt-O>.) This screen will appear.

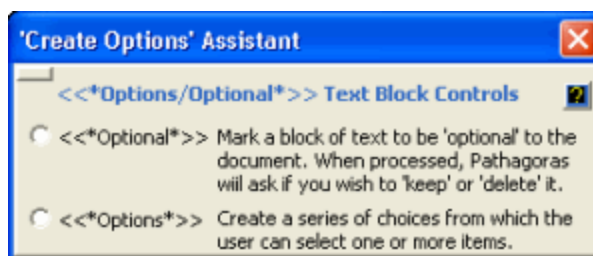


Figure 1. The Initial 'Create Options' Assistant Screen.

There are two distinct types of text blocks with which this Assistant deals.

A. <<*Optional*>>: 'Optional' text is pure 'take or leave it' text.

1. Highlight text you intend to be 'optional'.
2. Display the CVA. Check the <<*Optional*>> control.
3. If you want to add a Group Name (so that all Optional text block throughout the document will be processed together), check that box and provide the Group Name.
4. Click **Next>>**. "<<" and ">>" markers are placed around the selected text and if selected, the Group Name is added as well.

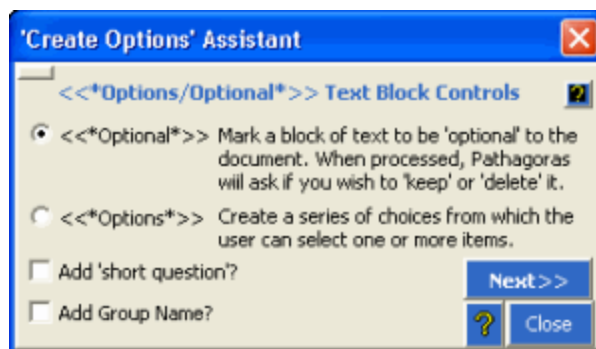


Figure 1. When <<*Optional*>> selected, the screen offers additional choices ('Add Short Question' and 'Add Group Name'), corresponding with advanced features available with <<Optional>> blocks.

B. <<*Options*>>: 'Options' text is that text which provides the user with two or more (up to six) selection from which to choose.

1. Highlight the text which reflects the choices you have composed. (The choices should be separated from each other by forward slashes (/)).
2. Display the CVA. Check the <<*Options*>> control. The menu is expanded to look like Figure 3.
3. If you wish Pathagoras to display 'short questions' in lieu of the entire optional text section when the Options block is encountered during document assembly, check the 'Add 'short questions' box. Select then number of short question holders you want reserved (same as the number of options you have provided).

4. Select '**Checkboxes**' if you want the user to be able to select more than one of the options. Select '**Radio Buttons**' if you want to limit the user to choosing a single option from the ones presented.
5. Press the Next button. Type your 'short questions' in the places indicated.

There you can select whether to add just the opening and closing "<<" and ">>" markers, or place holders for questions you may want to use to guide the user.

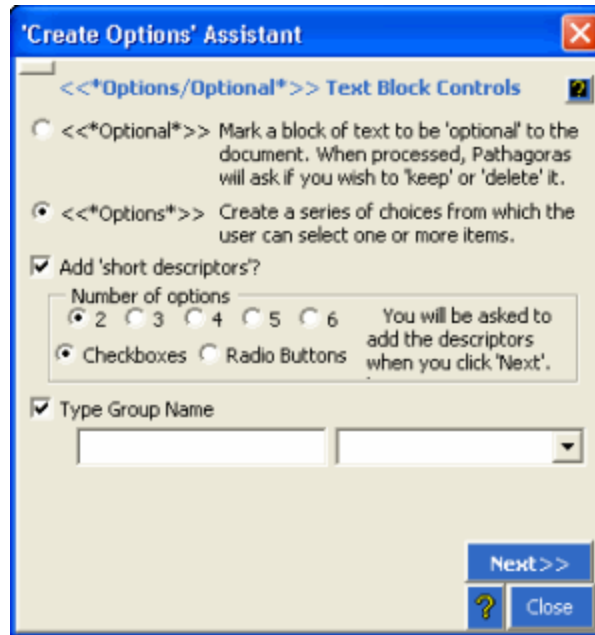


Figure 2. The Expanded Variable Creation Assistant Screen.
(Expanded after the <<*Options*>> radio button was selected.)



Don't be afraid to make a 'mistake' in creating <<*Options/Optional*>> text blocks. After you press an action button, Pathagoras will display an "Undo" button (not shown here). Click it once or twice to take you back to your original text. You can then start over with a different approach.

11.16 Negative Optional (/NEGOPT)

By definition, 'Optional text' is 'keep it' or 'delete it' only. If the answer to the keep this text is 'Yes', 'True' or 'Keep', the entire block is preserved. If 'No', 'False' or 'Delete' the block is deleted.

However, there often is a need to delete the 'yes' choice but allow other text in the document (that reflects the 'no' or 'false' choice) to remain. '/NEGOPT' fills that need.

To enable the negative optional, just add '/NEGOPT' and some 'alternative' text immediately after the 'if true' text, and before the closing >>. Don't add a space after /NEGOPT. Treat it like a slash. (Red color for emphasis only. No coloring is required. CAPS, however, are mandatory.)

If value of the optional block is True, the text before the /NEGOPT is kept and the text after (and including) the /NEGOPT is deleted; if false, the text after the /NEGOPT is kept, and the 'true' part is deleted.

Examples:


```
<<*Optional*!Apples!*An apple a day keeps the doctor away./NEGOPTThose
wormy apples didn't keep my doctor away.>>
<<*Optional*!Bananas!*My doctors tells me that bananas are the perfect
fruit./NEGOPTYes, I have no bananas.>>
<<*Optional*!Cherries!*Please top my [sundae/milkshake] with a
cherry./NEGOPTI don't want any cherries. They have pits.>>
```

It works with 'simple options' as well:

```
{!Apples!An apple a day keeps the doctor away./NEGOPTThose wormy apples didn't
keep my doctor away.}
{!Bananas!My doctors tells me that bananas are the perfect fruit./NEGOPTYes, I have
no bananas.}
{!Cherries!Please top my [sundae/milkshake] with a cherry./NEGOPTI don't want any
cherries. They have pits.}
```

What if the 'positive' answer should be blank and the 'negative' answer is the one that should contain text?

Not a problem. A 'Negative Optional' text block can be used to provide *just* the negative text in the appropriate circumstance. Just leave a blank (a simple slash) for the 'positive' option..
E.g,

```
<<*AskOptional*!YesChildren!Does client have children?*>>
```

(further in document)

```
<<*Optional*!YesChildren!My children's names are [Names of Children].>>
```

```
<<*Optional*!YesChildren!*/NEGOPTIf I should subsequent to this writing, have children,
>>I bequeath my estate to my children in equal shares.
```

NOTE: Except for the 'color coded optional test' example above, the colors in the sample text snippets are for emphasis and illustration only. No coloring is required for any 'coding' of Pathagoras commands.

11.17 Testing & Editing

Pathagoras provides an easy way to test, edit, retest, re-edit your <<*Options/Optional*>> text block structure (and any other text within <<double angle brackets>> without having to assemble a complete project.

The routine is called 'Process'. With the click of the **Process** button, Pathagoras will act upon the current document as if you were the end user initiating a 'document assembly' routine.



Before getting started with 'Process', *save your work!* (If you test -- process -- your work without saving it, you may be disappointed when your efforts are 'tested away.')

- To **process** the <<*Options/Optional*>>, {Simple Options}, <<*Repeat*>> and <<document>> blocks in your document:

Option 1: Anywhere in the document, press <Alt-P. (for 'p'rocess)

Option 2: If that hotkey doesn't work, or you prefer 'pure mouse,' click the Pathagoras Features menu. Click the 'Process' element from the list. (Note, you can elevate this key to Word's Quick Access Toolbar to make it still easier for you.)

- **Structure and Integrity Checkers:** Pathagoras can review a source document (whether final, or simply 'under construction') to see if you are complying with 'the rules.' In many cases, it can automatically (with your permission) fix any errors it finds. So even before 'Processing' the document, run the 'Structure Checker' first and study its results.

To activate the **Structure Checker**, click "Wizards & Assistants | Structure Checker" item from the Pathagoras drop down features menu. (Structure Checker is also a pre-assigned **Alt-Q | My Buttons** element.)



If you have a large document, think about testing only a portion of it. Copy and paste the portion you want to test onto to a new page. 'Process' just that section. Once the selected section is perfected, then test it within the original, complete document.



Because all text within <<*Options/Optional*>> text blocks is 'plain text,' you do not have to have Pathagoras loaded on your system in order to *create* them. You can create and fully edit them anywhere. This allows 'at home' and third party editing. Test them after you get them onto a 'Pathagoras' enabled computer.



This is not a clause 'testing' tip, but rather a document assembly testing tip. As you are testing how your documents are coming together, you may wish to *suppress* the processing of <<Options/Optional>> text block when you call in a document from a DropDown List. Look for the button probably labelled 'Process' and click it. It should toggle to 'Suppress.'" Now, when you call in a document, nothing other than a display of the document (actually a copy) occurs..

Despite the above warnings to the contrary, do not worry *too* much about losing your work while testing. As you perform tasks in Pathagoras, remember that you are always in Word. You can click the Undo button (or press Ctrl-Z) a sufficient number of times and, in almost all cases, restore your work to its pre-processed state.

See also:

Structure Checker

['The' Process](#)^[16] (what we mean by 'processing' a document)

11.18 Administrative Text

Every program has got to be able to distinguish between the commands and the substantive text that a command controls. It's no less true with Pathagoras and its plain text approach. We call the 'non-substantive' areas of a document 'administrative text.'

Administrative text has the following elements:

1. **Boundary Markers.** What demarks the beginning and end of the command block. In Pathagoras it is << and >> for a 'robust Option or Optional call' and for a Repeat block. (This boundary set is also used for a 'document call'³⁰², but because there is no 'command' following the opening boundary, Pathagoras knows its purpose.)
2. A **command** (*Options* or <*Optional* or *Repeat*) (**mandatory**). The asterisks on each end of the command are an essential part of the command.
3. An '**argument**' (optional) which refines the command and tells Pathagoras further goals of the command. E.g., *Options(radio)* tell Pathagoras to display the choices with single-select 'radio' buttons; *Repeat(and;)* tells Pathagoras to separate the repeating text with semi-colons and to add an 'and' before the last repeated text. The argument is within the command section (before the second '*').
4. A **!groupname!** (optional) to tie the answer of the first element of the group with remaining elements of the group further down in the document.
5. **Prompts** (optional) to offer short text snippets to guide (prompt) the user when making a decision regarding Options, Optional or Repeats.
6. If a **groupname** or **prompts** have been added to the command block, a closing asterisk must be placed at the end of the administrative text to signal its close. (If you have neither groupnames nor prompts, this third asterisk is not needed.)

You cannot have characters other than the above within the administrative text area. Pathagoras just won't know what to do with them, and it will fail in processing the block.

Examples. Remember, the colors in the below examples are for emphasis only. They are not needed for proper structure. You can copy and past any (or all) of the below examples into a Word document. Process them by pressing <Alt-P>.

```
<<*Options*Federal Express/USPS (First Class)/UPS>>
```

The administrative text above is simply the command "<<*Options*" and only 2 asterisks are used.

Now imagine a longer version of the above:

```
<<*Options*Per your request, the widgets will be shipped by Federal Express and we will bill you for the extra cost of shipping./Per your request, we will send the widgets by standard First Class mail./Per your request, we ship your purchase via United Parcel Service.>>
```

Without formal prompts, the first 200 characters of each choice will be presented as the text in the 'choices' screen as prompt text. That's a lot of reading. Time to use prompts!

Prompts are used in the example below. Short, sweet, simple. Note the asterisk at the end of the prompts (i.e., the close of the administrative text):

```
<<*Options*FedEx/USPS (First Class)/UPS*Per your request, the widgets will be shipped by Federal Express and we will bill you for the extra cost of shipping./Per your request, we will send the widgets by standard First Class mail./Per your request, we ship your purchase via United Parcel Service.>>
```

If a !groupname! is used, it must appear immediately after the command (and before any prompts). Note that the third asterisk (after the prompts) closes 'all' of the administrative text.

```
<<*Options*!Shipping!FedEx/USPS (First Class)/UPS*Per your request, the widgets will be
shipped by Federal Express and we will bill you for the extra cost of shipping./Per your request,
we will send the widgets by standard First Class mail./Per your request, we ship your purchase
via United Parcel Service.>>
```

If you have nested options/optional blocks, treat such blocks independently, and as if the nested block is 'substantive' for purposes of the outer block. So, while, in the below block, administrative text can be seen as spanning almost the entire first line, for purposes of the outer '*Optional*' block (and this rule), the administrative text ends after 'shipping?' and just before the nested '<<*Options* . . . block.

```
<<*Optional* Are we shipping?*<<*Options*!Shipping!FedEx/USPS (First Class)/UPS*Per your
request, the widgets will be shipped by Federal Express and we will bill you for the extra cost of
shipping./Per your request, we will send the widgets by standard First Class mail./Per your
request, we ship your purchase via United Parcel Service.>>>>
```

[*Aliases* and Administrative Text](#)¹²³: Newcomers (and even not so newcomers) to Pathagoras may become a bit perplexed when *Aliases* (and their stars) are added to administrative text. That's a lot of asterisks at play. The link will take you to a separate article we have prepared on this topic. It's actually quite easy and logical.

The Pathagoras System

{Simple Optional} Text Blocks

Part



XII

12 {Simple Optional} Text Blocks

Pathagoras provides a very easy approach to providing optional text blocks. Simply surround text you want to be 'optional' with {curly braces}.

Structure: {This is the optional text. Curly braces at each end is all there is to it. }

Like its 'big brother,' {Simple Optional Text} offers *two* types of blocks:

1. **“Optional” text:** This is pure ‘optional text.’ At document assembly time, the program will pause and ask “Include this text?” The user need only select “Yes” or “No” to tell Pathagoras whether the text block should be retained in the final document.

{The widgets you have ordered are not currently in stock. We will ship them as soon as possible. If we have not shipped within 5 days of this date, you will have the option to cancel the order. }

2. **“Options” text:** You can convert 'Optional' text to a quick form of 'Options text' to allow the user to choose among several options. Each option is separated from the others by a simple, “/OR” (No quotes. Caps mandatory. Color not required.). At document assembly time, the options are presented to the user on a selection screen. The user selects one or more of the options shown.

{As per your request, the widgets will be shipped by Federal Express. We will bill you for the extra cost of shipping./OR As per your request, we will send the widgets by standard ground transport. This may take 3 to 5 additional days./OR As per your request, we will hold the widgets for pickup by your courier./OR The widgets you have ordered are not currently in stock. We will ship them as soon as possible. If we have not shipped within 5 days of this date, you will have the option to cancel the order. }

i The **"/OR"** separator: The '/OR' separator will present the end user with mutually exclusive 'radio' buttons on the choice selection screen.

i The **"/ANDOR"** separator: If you want the end user to be able to select more than one of the options, separate the choices with "/ANDOR". (No quotes. Caps Mandatory.)

When, during document processing, Pathagoras encounters the optional text block (#1 above), it will highlight the text in the document and ask if you want to keep it (Figure 1).

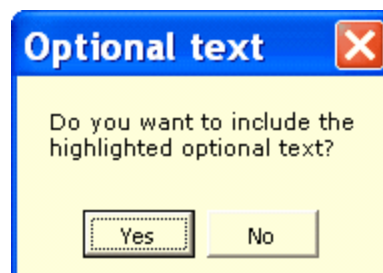


Figure 1. Optional Text dialog.
If you select <Yes>, the boundary markers

are removed and the text remains in the document.
If you choose <No>, the entire text block is deleted from the document.

When Pathagoras encounters the options text block (#2 above), it will parse out the individual choices and display them onto buttons on a selection screen. (If the text is too long to fit, only the first 200 or so characters of the option will display.) Check-boxes are shown at the left of each choice so that you can choose more than one option, if desired.

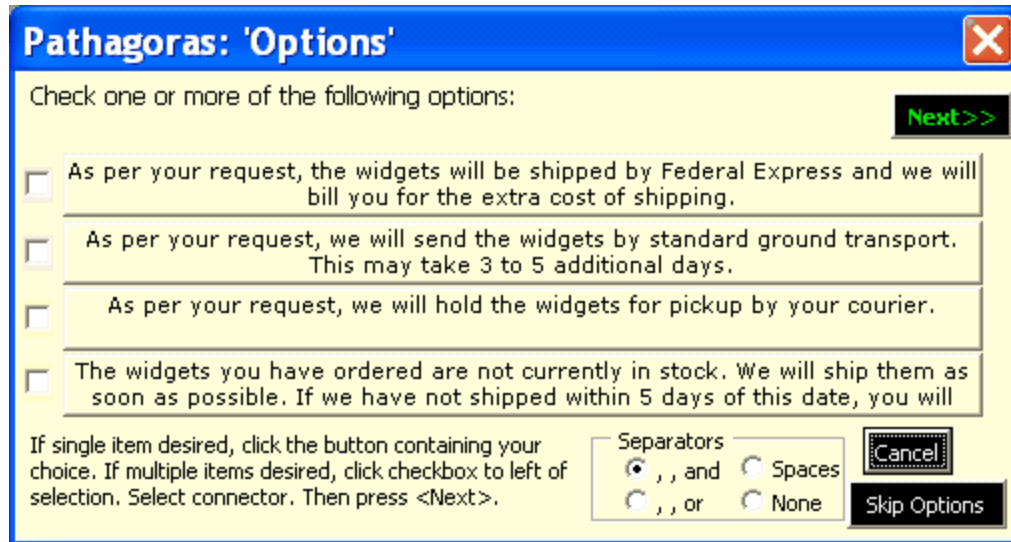


Figure 2. Options block dialog.

Note that the actual text of the option is provided
(subject to space constraints.)

Note that the "/ANDOR" separator was used. It lead to a screen
where the end user can choose one or more of the options.

!GroupNames!: Like its big brother, you can insert a !Groupname! to the beginning of related blocks of optional text. This will cause each {Simple Optional} and {Simple Options} containing the same !Groupname! to behave in a similar fashion.

Examples:

With Simple Optional text:

{!Disclaimer!We disclaim all liability if you try to shave your head with our chainsaw. } (blah blah blah . . .) and furthermore {!Disclaimer!we also disclaim responsibility if you try to use our hedge trimmer to trim your fingernails. }

With Simple Options text:

We suggest that the color for the living room be {!colors!red/ORyellow/ORburnt orange} and the contrasting trim color be {!colors!yellow/ORpurple/ORroyal blue}.

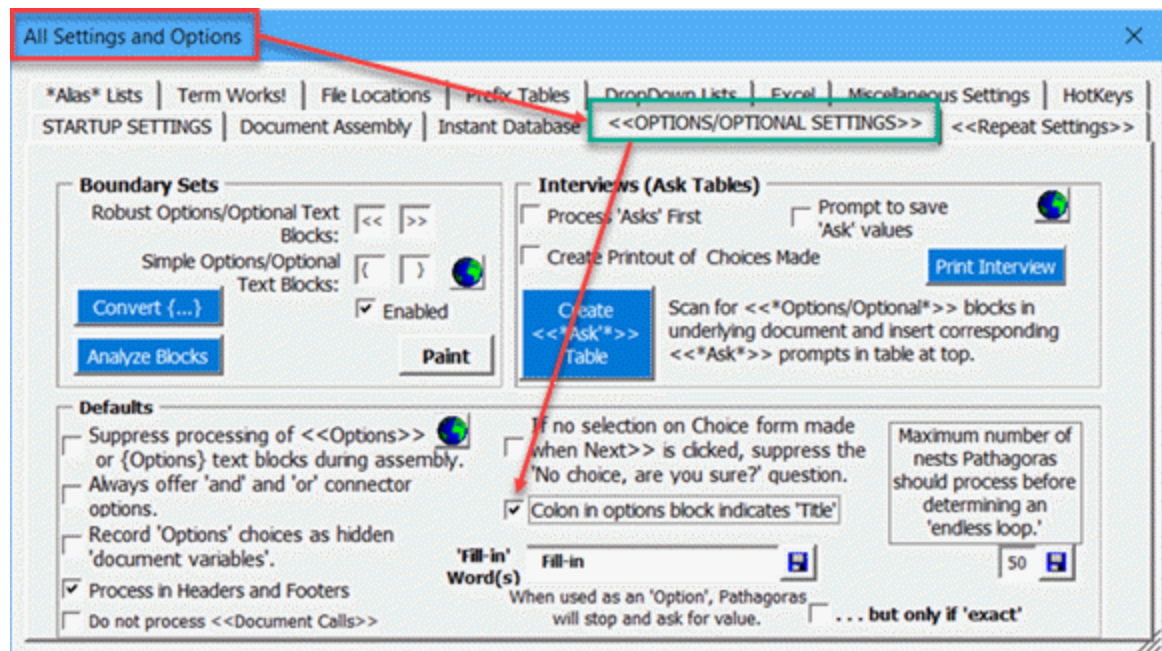
Notes: The automatic selection of subsequent choices by Pathagoras when using !GroupNames! depends upon the *position* of the first choice in the list, not its answer. So, in the first above example, if 'yellow' (the second option) is selected when the first block is encountered, 'purple' (the second option in the second block) will be automatically selected.

Note : The same examples above appear in the section discussing [!Groupnames! with robust <<Optional/Options>> blocks](#)^[214]. The differences are (1) the absence of the ability to add a short question or prompt for the end user, and (2) the requirement that '/OR' (instead of just a 'slash') be used to separate choices.

Titles: Sometimes you need to communicate more information about the simple options block that the display of the actual text can accomplish. You can add a short **title** to the options block. To do so, simply type a word or phrase, up to 30 characters, followed by a colon, to the beginning of the simple options block. The title will be removed before the optional text is highlighted. It will be added to the question asking whether the text should be kept or deleted.

Example: {Free Shipping:There will be no additional charge for this service.}

Notes: You must enable this feature. To do so, select "Colon represents Title" in the Options/Optional settings screen.




Click to enlarge

A title can be up to 30 characters in length. If the colon appears more than 30 characters from the left, it is presumed part of the original text.


If you need to retain what Pathagoras would otherwise interpret as 'title' text, either proceed the text with an asterisk or use Robust Optional.


E.g., { *Attention: [Contact Name] }. OR <<*Optional* Attention: [Contact Name]>>


You can combine !Groups! with Titles, but the !Groupname! must come first. (Use a Title only with the first appearance of the !Group!. You simply won't need it for the remainder.)


 **Note:** The {Simple Optional Text} module will identify and act upon any curly brace sets in your document. If a document already contains text within {curly braces} that you do not intend to be optional, you have 3 choices:

1. **Disable** Pathagoras' ability to process text it finds within curly braces. Click [here](#) to learn how to [enable and disable](#)^[208] this function.
2. Change the boundary markers that denote the 'native' text to something other than '{' and '}'. (Plain parenthesis will typically serve the same 'highlighting' function.)
3. Change the boundary markers that denote the 'simple' options text to something other than '{' and '}'. [Read more here](#)^[202].
4. Just forego simple options and use '[robust options](#)'^[172].

 A {Simple Optional} text block cannot span more than a single cell of a table. {Simple Optional} text block can encompass an entire table, but both block markers must be outside the table.

 You are limited to no more than 9 choices if using multiple choice (/OR or /ANDOR) features of simple options. If you need more, use the Robust Options described in previous sections.

 The appearance of /OR in a simple options block will always be interpreted as 'belonging' to the simple options block, not to a nest 'other' component (most often a nested multiple choice variable). So "{I really like fruit. My favorite is [apples/ORbananas/ORMangos]}" will result in the optional block behaving as an 'options' block. If you want to preserve the multiple choice variable, just use plain slashes, e.g., "{I really like fruit. My favorite is [apples/bananas/mangos]}"

 **NESTING RULES:** You cannot nest a simple options block within a simple options block. You cannot nest a <<*Repeat* . . .>> block within a simple options block. You **can** nest a robust <<*Optional* . . .>> block within a simple options block. You **can** nest a robust <<*Options* . . .>> block within a simple options block. You **can** nest plain variables within a simple options block, but not multiple choice variables. Despite these allowances, a 'best practice' would be to use simple options for their original 'simple' purpose, and don't try to nest anything (other than simple variables) within them.

12.1 Differences

While similar in function, the 'simplified approach' is more limited in function as compared to its 'big brother':

- the more powerful <<*Options/Optional*>> text blocks allows you to provide meaningful questions to the end user. See [Optional Text \(advanced\)](#)^[175].
- the more powerful <<*Options/Optional*>> text blocks allows you to provide a series of (typically shorter) 'questions' to reflect the options, instead of reproducing the actual text of the options; and
- the more powerful <<*Options/Optional*>> allow you to create sophisticated [Ask Tables](#)^[252] which present a top of document listing of all <<*Options/Optional*>>questions found in the body of the document.

- the more powerful <<*Options/Optional*>> allow you nest one option within another, allowing you to cascade questions. Once the 'outside' block is processed, any remaining options blocks will be processed. (You cannot nest 'simple options/optional' blocks.)

Because of the way that Pathagoras 'reads' the text in order to process it, Word 'fields' such as cross-references (which frequently contain "{" and "}" characters) can cause Pathagoras to act unpredictably. If you get such result, 'disable' the simplified {Optional} text block feature. (Do so via the main Utilities/Settings screen. The check box is at 'screen center.') Test again and see if the results improve. If they do not, contact Pathagoras Customer Service.

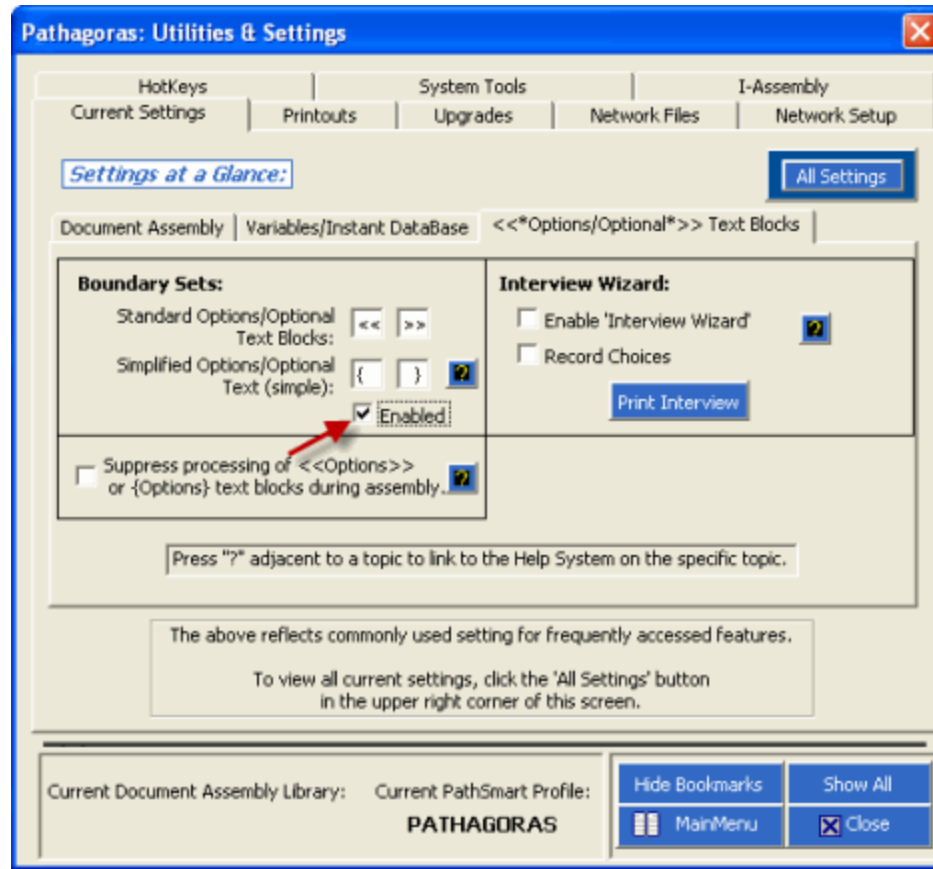
For many users, however, the simplicity offered by the {optional text} technique will surpass the benefits of the <<*Optional/Options*>> blocks. Please note that this feature may be used *in addition to*, and concurrently with, the <<*Options*/*Optional*>> Text blocks discussed in the previous screen.

12.2 Enabling/Disabling

Many users already have documents containing curly braces surrounding blocks of text. More likely than not these preexisting braces do not represent optional text. (Perhaps they have been used to mark out instructional or informational material in legacy documents.) If Pathagoras 'processed' {text} without warning, it would generate quite a bit of user confusion. {Simple Optional} ships in the 'enabled' state', but you can (1) disable the function or (2) you can change the character set that identifies a 'Simple Options' block to something else.

1. To enable/disable the module, call up the Utilities/Settings screen. Click the <<*Options/Optional* Text Blocks>> tab under 'Settings at a Glance' section. Next to the 'Simplified Options' label, change the characters or uncheck the "Enabled" box .

2. To change the boundary characters to another character set, type one or two 'mirroring' character sets in the appropriate text box. '{{' and '}}' will work, as will '<*' and '*>'. (When you type an opening boundary set, Pathagoras will automatically mirror the closing set for you.)



Of course, if you have a existing text already marked with {curly braces}, you might consider changing the boundaries of that text. Plain (parentheses) typically can serve the same 'highlighting' function, and they won't interfere with Pathagoras' operations.

12.3 Options (Mixing Simple with Complex)

If you use [!groups!](#)¹³⁶ to link selections, you will be pleased know that you can pair {simple options} with the more complex sister using the [!groupname!](#) feature. That way, you can ask a meaningful 'question' at the top of the document and carry the answer forward to the remainder of the document using the less complicated {simple options} construct.

Example: The groupname !sex! is typically not descriptive enough to stand alone. An alternative would be to name the group "!sex of our client!" but its length might be cumbersome depending upon how many members of the group there are further in the document. So by using the robust AskOptions construct at the top of the document, we can ask a meaningful question, but use the simple grouname !sex! in the remainder of the document body.

```
<<*AskOptions*!Sex!Our client is a male/Our client is a female*>>

DEED OF CONVEYANCE

This Deed, by and between [Client Name], an unmarried
<<*Options*!sex!*male/female*>>, Party of the First Part, and . . .
```

```

    The Party of the First Part conveys all rights and title that
    {!sex!he/ORshe}has in the property, to the Party of the Second Part and further
    conveys any inchoate or residual interests held by {!sex!him/ORher}in said
    property. . .

```

12.4 Negative Simple Optional Text

The default action of an 'No' or 'False' choice for a {simple optional} text block is to delete it completely.

However, when you tie several {simple optional blocks} together using !groupnames!, there occasionally is a need to delete the 'yes' choice but allow other text in the document that reflects the 'no' or 'false' choice to remain.

Pathagoras lets you provide a 'false' value to simple optional text. To provide this 'false' part, simple type "/NEGOPT" (no quotes) at the end of the 'True' part of the {simple optional text block}, followed by the language you want to remain in the document if "No" or "False" is chosen. (If the user answers 'True' or 'Yes' or 'Keep', the part before the slash remains and the false part is deleted, If the user answers 'False' or 'No', the part after the slash remains and the 'true' part is deleted. (Don't add a space after the /NEGOPT divider. Treat it like a slash.)

Here is an example. Let's assume you are drafting a Will and want to provide for Guardianship of minor children (if any). You might first ask if there are any children, and if there are, name a guardian as their custodian if any are minors. If there are no children, you want to state that fact. Here is how that setup might look.

```
{!children!There were [number of children] born of the marriage. }
```

```
{!children!If any children who are still under the age of 18 years survive me, I appoint [Name
of Guardian] to serve as their guardian during their minority./NEGOPTThere were no children
born of the marriage. }
```

OTHER USAGE NOTES:

Keeping 'number' consistent.

The letter 's' is typically added to nouns to make them plural. That same 's' is deleted from the corresponding verb. Conversely, when the noun is singular, the 's' is attached to the verb.

"The dog barks." "The dogs bark." (Exceptions abound: Children, not 'childs'; 'is/are' and 'go/goes'; but the solutions are similar).

This 's' shift defies easy use of *Optional* block, but to set up everything as '*Options*' takes up a lot of real estate on your screen. The 'negative option' can be adapted to help keep your noun and verb 'number' consistent with minimal extraneous text. Examples:

- The child{!c!ren} {!c!are/NEGOPTis}.
- The dog{!d!s} bark{!d!/NEGOPTs}.

Note in each of the above examples the first choice is pure 'optional' text. You could have written the above as all two part Options, and you would achieve the same result. The choice is yours as an author. Choose whichever seems the most economical and logical.

So the 'negative optional' has the effect of behaving like pure optional text when there is no /NEGOPT (the typical situation) but when the /NEGOPT exists in the optional block, it behaves like an Options (or multiple choice) block with two elements.

"I'll do this . . . You do that" lists

The /NEGOPT tool is perfect for what we call the 'I'll do this. . . You do that' assignments.

Study the following. The group names 'wash' etc. are identical for both this, but the /NEGOPT switch controls when 'I' am not performing a certain chore, and you are

You can copy and paste it into a document and process it by pressing Alt-P.

I'll do this:

- {!wash!Wash the dishes }
- {!mow!Mow the lawn }
- {!garbage!Put out the garbage }
- {!cat!Put out the cat }
- {!children!Feed the children }

You do that:

- {!wash!/NEGOPTWash the dishes }
- {!mow!/NEGOPTMow the lawn }
- {!garbage!/NEGOPTPut out the garbage }
- {!cat!/NEGOPTPut out the cat }
- {!children!/NEGOPTFeed the children }

See Also: [/NEGOPT with robust Optional text](#) ¹⁹⁶

12.5 Converting

{Simple Optional/Options} blocks are, well, simple. They are simple to create and simple to understand.

Here's a quick review on how to create {Simple Optional/Options} blocks:

- To create a {simple optional} block of text, surround that text with curly brackets. The text can be of any length.
- To create a {simple options} block of text, surround that text with curly brackets. If the choices are mutually exclusive, insert '/OR' (caps required) between each choice. If you want to be able to select one or more choices, insert /ANDOR (caps required) between each choice.
- To 'group' simple blocks (so the first selection of a simple options/optional block will automatically determine the selection for the remaining ones), insert a !groupname! (any text, any length up to 20 characters) immediately inside the opening curly bracket.

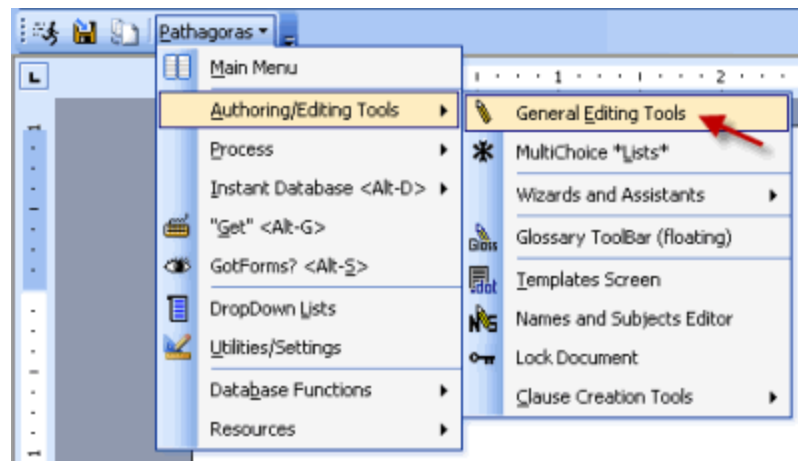
Remember: "Optional" means the entire block of text is keep or delete.
 "Options" will list choices, one or more of which you intend to keep.

But with this simplicity come some limitations. They just don't have the robustness and flexibility that the more 'elaborate' <<*Options/Optional*>> blocks possess. Examples: You cannot attach questions to {Simple} text blocks. You cannot nest {Simple} blocks with each other to obtain a cascading answer pattern.

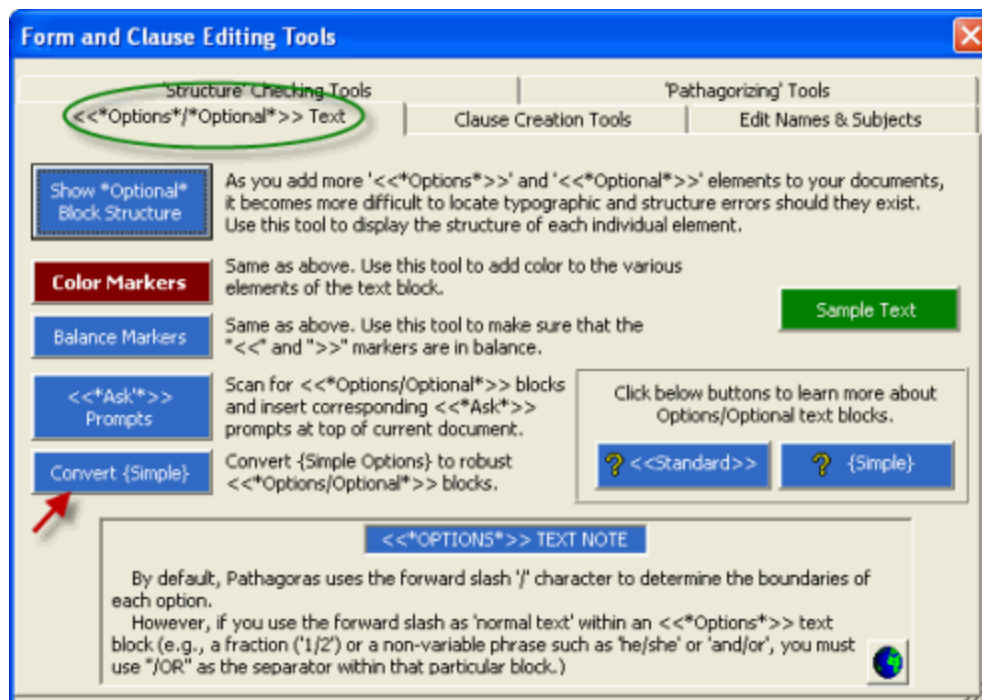
There likely will come a time that you wish to convert at least some your {Simple Optional/Options} text blocks to <<*Options/Optional*>> blocks. This will be especially so if you want to take advantage of Pathagoras' <<*'AskOptions*>> table and more immediately, the <<*AskOptions*>> wizard that Pathagoras provides to automatically create those tables.

It is easy to convert {Simple Options} to robust <<*Options/Optional*>> blocks:

1. From Pathagoras' dropdown features menu, click 'Editing/Clause Creation Tools' and then 'General Editing Tools'.



2. Select the <<*Options/Optional*>> tab from the screen that next appears.



3. Click the **Convert {Simple}** button.
4. Pathagoras will quickly identify and then convert any {simple options} blocks into the appropriate <<*Options*. . .>> or <<*Optional*. . .>> block. If no !groupname! has been assigned to the {simple optional} block, Pathagoras will ask for one as it converts each block. (If the {Simple} block already contains a !groupname!, it will be carried forward.)

See Also:

[!GroupNames!](#)²¹⁴

The Pathagoras System

Groups and !GroupNames!

Part



XIII

13 Groups and !GroupNames!

Sometimes, the answer to a particular question will lead to an answer to a subsequent question. In such cases, rather than responding to each <<*Options/Optional*>> text block, you can create text block ‘groupings.’ When you answer the question, or select a certain option (or set of options) from the first member of the group, Pathagoras will hunt down other members of the group and select the corresponding choices. E.g.:

Structure:

```
<<*Options*!groupname!(prompt1/prompt2)(etc.)* . . .>>
```

```
<<*Optional*!groupname!(prompt)* . . .>>
```

Note: The asterisks surrounding 'Options' or 'Optional' are part of the. That's (so it will be distinguished from other appearances of these words in your text). If you have added a !group name! or prompts/questions, or both, a final '*' must be inserted to close that 'administrative' section of the block.

Note: At the risk of repetition *ad nauseum*, but recognizing that this may be the first time you have seen this, the colors used in the examples are not required. they are for emphasis only. If used in your document, the colors will have no effect on the final product.

Examples:

We suggest that the color for the living room be
 <<*Options*!colors!*red/yellow/burnt orange>> and the contrasting trim color be
 <<*Options*!colors!*yellow/purple/royal blue>>.

<<*Optional*!Disclaimer!Include disclaimer language?*We disclaim all liability if
 you try to shave your head with our chainsaw.>> (blah blah blah . . .) and
 furthermore <<*Optional*!Disclaimer!we also disclaim responsibility if you try to
 use our hedge trimmer to file down your fingernails.>>

Notes:

- The groupname (the text between the two “!” marks) can be no longer than 30 characters. We recommend 'the shorter the better' (including a single character). However, keep the end-user (who may not be yourself) in mind and make it meaningful to others who may be processing the document.
- When used with an *Options* block of text, the automatic selection of subsequent choices by Pathagoras almost always depends upon the *position* of the first choice in the list, not its answer. So, in the first above example, if ‘yellow’ (the second option) is selected when the first block is encountered, ‘purple’ (the second option in the subsequent blocks of the same !group!) will be selected automatically when it is encountered.


There are two 'exceptions' to this rule:


(1) When you !group! a <<*Repeat* . . .> block with an <<*Options* . . .>> block, the selection of the <<*Options* . . .>> text depends upon whether '0', '1' or '2+' is the response to the *Repeat* question. [This is discussed here](#)^[237].

(2) When the value being selected/tested is numeric (e.g, grades, scores, temperatures), and you are using the `<<*AskValue*>>` command. The `<<*AskValue*>>` command directs Pathagoras to select a response that falls within various ranges of values. the `<<*AskValue*>>` command is discussed here.)²⁵⁷

- A `!groupname!` is **mandatory** for `<<*Optional/Options* . . .>>` blocks placed in headers, footer and text boxes.
- `<<*Optional/Options* . . .>>` blocks in headers and footers cannot be processed 'stand-alone'. There must be a 'referral' to the header/footer via a `!groupname!` set in the document body. Therefore, the header/footer `<<*Options/Optional* . . .>>` block must contain a `!groupname!`, and that `!groupname!` must have its origin within the document body.

The document body origin s can be done via a 'regular' `<<*Options*!groupname!* . . .>>` setup, via an Interview (e.g., `<<*AskOptions*!groupname!* . . .>>` or via a `<<*Set*!groupname!= . . .>>`

 When creating a second and subsequent members of a `!group!`, the prompts (the shorthand terms used to identify a particular option), if any, need not (actually should not) be repeated. Once the first member of the `!group!` is processed, Pathagoras has all the information it needs to handle the remainder. The extraneous repeat of the prompt language just takes up space, will be ignored by Pathagoras, and may be confusing to the end user.

 `!GroupNames!` are only needed when you intend to effect the 'auto-select' functions described about. However, the `!GroupName!` is displayed at the top of the selection screen (above the choices). If you think the end-users could benefit from additional guidance as to the selection(s) they are about to make, a meaningful `!groupname!` would be appropriate.

See also:

Structure Checker

[!Groups! and *Aliases*](#)¹²²

The Pathagoras System

Anatomy of Variable and Optional Text Characters

Part



XIV

14 Anatomy of Variable and Optional Text Characters

Here is a collection of the various character sets for marking variables and optional text. They are listed from the simplest to the most complex:

i The boundary and other colorings that you will see in the text markers below is for emphasis only. No marker coloring is required.

The anatomical discussion of each example is set out in blue:

VARIABLES:

Simple variable: [Client Name];[quantity] **Anatomy:** Two inward facing square brackets (the *boundary* characters) surrounding a core term. The core term can one word or a short phrase or itself can have structural components

Multiple choice variable: [chocolate/vanilla/strawberry] **Anatomy:** A simple list of optional terms within variable boundary characters. Each choice separated by a forward slash.

MultiChoice variable (alias): [*States*]; [1st stop *States*]; [second stop *States*]. **Anatomy:** Regular variable structure. The two asterisks indicate the call to an '*Alias* List' An *Alias* List is a series of optional values (in this case the 50 United States) stored in a separate Excel file. Note that the same alias term can be used within multiple variables. The additional text is what distinguishes each from the other.

Titled variable: [Special Order:Yes/No/Not Applicable] . . . [Priority Mail:Yes/No/Not Applicable] **Anatomy:** Regular variable structure. The 'title' is typed immediately after opening boundary character and 'closed' by the colon. Titles can be used for simple or multiple choice variables to assist user as to what the specific variable (which otherwise is identical to another variable) references.

Grouped variables: [!client!he/her/they] . . . [!client!his/hers/theirs] **Anatomy:** Regular variable structure. The 'groupname' is typed immediately after the opening bracket between '!' (exclamation marks). A groupname is used to tie two or more variables together. (The positional selection of the first variable automatically causes the same positional selection of subsequent variables. Groups used only for multiple choice variables)

OPTIONAL/OPTIONS TEXT BLOCKS (simple)

{Shipping included in cost} **Anatomy:** Two facing curly braces (the *boundary* characters) surrounding a core term. The core term can range from a single character to multiple paragraphs. When encountered, Pathagoras will highlight the text and ask if you want to keep or delete.

{!shipinfo!Shipping included in cost.} . . . {!shipinfo!Remember shipping charges have already been included in the list cost of the product.} **Anatomy:** Two facing curly braces (the *boundary* characters) surrounding a core term. The 'groupname' is typed

immediately after the opening brace, between '!' (exclamation marks). The core term can range from a single character to multiple paragraphs. The !group! used to tie two separate blocks together. The selection or rejection of the first encountered member of the group results in the same action for the remaining members of the group.

{Shipping included in cost/ORShipping Costs Extra} **Anatomy:** A simple list of optional terms surrounded by the boundary characters. Each option separated by the characters "/OR" (no quotes, but the capital 'OR' is mandatory). To allow multiple selections, use /ANDOR as separator.

{*States*} **Anatomy:** Regular options structure, except the two asterisks indicate the call to an '*Alias*' List' which provides the various choices. (An 'alias' list is a series of optional values (in this case the 50 United States) stored in a separate file and referenced by a simple 'alias.' The core term is the alias and it is surrounded by the boundary characters.

Introduction: OPTIONAL/OPTIONS TEXT BLOCKS ('robust')

The setup for Pathagoras' more robust <<*Options*/*Optional*>> text blocks shown below is not without its complexities. Compared with the setup required by competitive programs, however, it is undeniably simpler. Because it is plain text, you don't have to fish around for the right field coding and switches. Seen 'on screen' when the source text is recalled, it actually becomes quite 'readable' once you become familiar with the 'anatomy.'

The precise placement of the various characters that make up the blocks is mandatory. Not to worry too much, however. Pathagoras provides tools (1) to automate the creation of <<*Options*/*Optional*>> blocks and (2) to check the structure of each block after they have been created.

The '**call to action**' is the key term in the text block that tells Pathagoras what to 'expect' from the remaining sections of the text block. It will be either Optional, Options or Options(radio).

The '**administrative section**' is that part of the text block that contains the 'call to action' and the group and display information. It does not remain in the final assembled document.

OPTIONAL TEXT BLOCKS (robust)

Basic:

<<*Optional*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>> **Anatomy:** Two sets of 'facing' angle brackets (the **boundary** characters) surround the core block. The red is only for emphasis. The word "Optional" surrounded by asterisks constitutes the 'call to action.' The actual optional text can range from a single character to multiple paragraphs. Don't forget the closing brackets at the end of the block.

With question:

<<*Optional*Is pickup available?*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>> **Anatomy:** Boundary characters surround the core block. The word "Optional" surrounded by asterisks constitutes the 'call to action' followed immediately by the prompt or question that you want to be shown to the end user during document assembly. A third asterisk closes out the '**administrative section**' of the block. The actual optional text can range from a single character to multiple paragraphs.

With **!Group!** (A **!group!** name is used to tie two blocks together. The selection or rejection of one member of the group results in the same action for the remaining members of the group.):

<<***Optional*!pickup!***You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>> (body text . . .body text)

<<***Optional*!pickup!***Be sure bring a crane with you.>>

Anatomy: *Boundary* characters surround the core block. The word "Optional" surrounded by asterisks constitutes the 'call to action.' The '**!groupname**' is typed immediately after the call to action between '!' (exclamation marks). A third asterisk closes out the '*administrative section*' of the block. The actual optional text can range from a single character to multiple paragraphs.

With !Groups! and Question/Prompt:

<<***Optional*!pickup!**Is pickup available?*You may also pick up your block of cement from our help desk between the hours of 9 a.m. and 5 p.m.>> (body text . . .body text)

<<***Optional*!pickup!***Be sure bring a crane with you.>>

Anatomy: *Boundary* characters) surround the core block. The word "Optional" surrounded by asterisks constitutes the 'call to action.' The '**!group!** name' is typed immediately after the call to action between '!' (exclamation marks) followed immediately by the prompt or question that you want to be shown to the end user during document assembly. A third asterisk closes out the '*administrative section*' of the block. The actual optional text can range from a single character to multiple paragraphs. (The prompt would appear only in the first instance of the group. It does not need to be repeated in subsequent members of the group.)

OPTIONS TEXT BLOCKS (robust)

Basic:

<<***Options***I have no children./I have one minor child./I have [num cdn] minor children.>>

Anatomy: Two sets of 'facing' angle brackets (the 'boundary' characters) surround the core block. The word "Options" surrounded by asterisks constitutes the 'call to action.' The various choices are separated by forward slashes. The actual text of the options will appear as the 'prompts' on the selection form during document assembly process.

With reference to *Aliases*

<<***Options****States*.>> **Anatomy:** The 'boundary' characters followed by the call to a 'MultiChoice List' alias.

With Prompts/Questions (up to 6 choices and 6 responses)

<<*Options*No children/One child/Two or more children?* I have no children./I have one child to whom I give, devise and bequeath the remainder of my estate./I have [number of children] minor children to whom I give, devise and bequeath the remainder of my estate, in equal shares.>> **Anatomy:** *Boundary* characters surround the core block. The word “Options” surrounded by asterisks constitutes the 'call to action.' The optional term '(radio)' is included in this call to action to indicate that the choices are mutually exclusive. The call to action is followed immediately by a series of prompts or question that you want to be shown to the end-user during document assembly. A third asterisk closes out the 'administrative section' of the block. The options text follows, each option separated from the other by forward slashes. Each option can range from a single character to multiple paragraphs.

With !Group! (A !group! name is used to tie two blocks together. (The positional selection of the first variable automatically causes the same positional selection of subsequent variables.):

<<*Options*!Children!*I have no children./I have one minor child./I have [num cdn] children.>>

<<*Options*!Children!*I give the remainder of my estate to the following persons./I give the remainder of my estate to my child./I give the remainder of my estate in equal shares to my children, per stirpes.>>

Anatomy: *Boundary* characters surround the core block. The word “Options” surrounded by asterisks constitutes the 'call to action.' The '!group!' name is typed immediately after the call to action between '!' (exclamation marks). A third asterisk closes out the 'administrative section' of the block. The actual options text, each separated from the other with a forward slash, can range from a single character to multiple paragraphs.

With !Group! and Questions/Prompts:

<<*Options*!Children!No children/One child/Two or more children*I have no children./I have one minor child./I have [num cdn] minor children.>>

<<*Options*!Children!*I give the remainder of my estate to the following persons./I give the remainder of my estate to my child./I give the remainder of my estate in equal shares to my children, per stirpes.>>

Anatomy: *Boundary* characters surround the core block. The word “Options” surrounded by asterisks constitutes the 'call to action.' The '!group!' name is typed immediately after the call to action between '!' (exclamation marks) followed immediately by the prompts or questions that you want to be shown to the end user during document assembly. A third asterisk closes out the 'administrative section' of the block. The actual options text, each separated from the other with a forward slash, can range from a single character to multiple paragraphs. (There must be as many prompts as there are actual choices. The prompts would appear only in the first instance of the group. They do not need to be repeated in subsequent members of the group.)

The Pathagoras System

<<*Repeat*. . .>> Blocks

Part



XV

15 <<*Repeat*. .>> Blocks

The 'Repeat' function allows you to duplicate a specific block of text a designated number of times.

When a <<*Repeat*(text)>> prompt is encountered, Pathagoras will ask you to designate the number of times the text within the block should be duplicated. When 'go' is signaled, Pathagoras will then 'repeat' the text the requested number of times.

Pathagoras can provide this 'repeat' function in a wide variety of ways. They are summarized here, and elaborated upon, with examples, in the subsequent sections:

Simple: In its simplest iteration, the structure of a 'repeat' block is simply "<<*Repeat *(text)>>" (with (text) being anything you wish: text, a signature line, a variable, anything).

Simple with incrementing variables. If you include bracketed variables within the scope of the <<*Repeat*>> prompt, Pathagoras will add a suffix to each repeated of the variable and increment its 'name' by one. "[Name@1] [Name@2] [Name@3], etc.

!Groups! You can add an optional !groupname! to the key text to link the 'repeats' value assigned when the first member of the repeat !group! is encountered to the repeats blocks in other locations in the document

Series Connectors: Pathagoras can add commas and an appropriate conjunction so that the repeated text can be made into a grammatically correct list. ("A1, A2 and A3" instead of simply "A1 A2 A3")

=====

But wait! There is still more!



After you have run any of the above routines and brought in, let's say, 5 'repeats' of a particular clause, and you find that you need a sixth, type the word "repeat" at the location where you need the next clause, followed by the key press <Alt-G>.

See Also:

['Repeat' Settings](#) ^[235]

['Repeat' in Clause Selection Screen and DropDown Lists](#) ^[240]

['Repeat' Alternatives](#) ^[241]

15.1 Simple Repeats

The structure of a simple <<*Repeat*>> block is simply the introductory word "<<*Repeat *" followed by whatever text you want to repeat, followed by a closing ">>"

Example (The colors are added for illustration purposes. They are *not* required):

```
<<*Repeat *Row, row, row your boat, gently down the stream.>>
```

When this above is encountered during a document assembly session, Pathagoras will pause and ask "How many times do you want to repeat this text?". If, let's say, your answer to the question is 3, the 'return' of the function will be

```
Row, row, row your boat, gently down the stream. Row, row, row your
boat, gently down the stream. Row, row, row your boat, gently down the
stream.
```

If you add an 'Enter' just before the closing '>>', i.e.,

```
<<*Repeat *Row, row, row your boat, gently down the stream.
>>
```

the processed text will appear like this:

```
Row, row, row your boat, gently down the stream.
Row, row, row your boat, gently down the stream.
Row, row, row your boat, gently down the stream.
```

15.2 Incrementing variables

The above causes a repeat of the identical text X number of times. You will more likely want to use the <<*Repeat*>> function to increment a variable that you place within the repeating text. The structure is identical to the 'Simple' example above, with the introductory word "<<*Repeat *" followed by whatever text you want to repeat, followed by a closing ">>"

Examples (The colors are added for illustration purposes. They are *not* required):

```
<<*Repeat *[ChildName]>>
```

When the repeat block is encountered during a document assembly session, Pathagoras will ask "How many times do you want to repeat this text?" The 'return' of the function would be, in this case, the named variable, incremented by the number of repeats called for. E.g, [ChildName@1] [ChildName@2] etc

You can include as much or as little additional text within the repeat block as you wish. All plain text will be repeated 'intact'; all bracketed text will be incremented as per the examples.

```
<<*Repeat *[Child@Name], Date of birth [Child@DOB]>>.
```

You can increment a number outside of a variable using the character set '[#]' (no quotes).

For example, if you wanted to make a numbered list using the above, you might type:

```
<<*Repeat *[#]. [ChildName].
>>
```

Suppress incrementing:

By default, all bracketed variables are incremented by each repeat called for. However, if you place a variable that you do not wish to increment (perhaps a date variable that will be consistent for all blocks, simply append an underline to the variable name (inside the bracket) and Pathagoras will repeat the variable but not increment it.

Example:

```
Signatures:
<<*Repeat* _____ [Date_]
                [Buyer]
>>
```

You can modify the settings of the 'non-increment' function, including the non-increment ending character and whether it is removed (let's say so that it will match other variable) or left in. "_" and 'remove' are the default settings.

The settings are found under the "Repeats" tab of the Utilities/Settings | All Settings screen.

15.3 Repeat with !Groups!

Frequently there is a connection between a <<*Repeat*>> prompt at the 'top' of a document with elements further down. For example if there are 'three' shareholders accounted for at the top of the document, three additional text blocks may be needed further down in the document to account for the number of shares each holds. The various <<*Repeat*>> blocks can be tied together with a common !groupname!

Structure: <<*Repeat *!groupname!(text1)>> . . . <<*Repeat *!groupname!(text2)>>.

When encountered during a document assembly session, Pathagoras will ask "How many times do you want to repeat (groupname)?"

Example (A 'repeat ask' is placed at the top of the document. It's sole purpose is to ask 'How many shareholders?' If the repeat ask is not included, the 'How many shareholders?' question will be asked when Pathagoras encounters the first substantive 'repeat block'):

```
<<*Repeat*!shareholders!*>>
```

The following individuals own stock in the corporation:

<u>Name</u>	<u>Shares Owned</u>
-------------	---------------------

```
<<*Repeat*!shareholders!* [ShareholderName], [SharesOwned]
```

```
>>
```

(. . .)

The shareholders waive notification of the upcoming annual meeting as evidenced by their signatures below:

```
<<*Repeat*!shareholders!* _____
```

[ShareholderName]

```
>>
```

When encountered during a document assembly session, the very first line (the repeat gatherer) causes Pathagoras to ask for the number of shareholders. It will use the answer further down the document to duplicate the text between the remaining << and >> markers the designated number of times. Of course, all [variables] inside any repeat block will be properly incremented.

Here the result of the answer to "How many shareholders?" is 3.:

The following individuals own stock in the corporation:


<u>Name</u>	<u>Shares Owned</u>
[ShareholderName@ 1],	[SharesOwned@ 1]
[ShareholderName@ 2],	[SharesOwned@ 2]
[ShareholderName@ 3],	[SharesOwned@ 3]
(. . .)	

The shareholders waive notification of the upcoming annual meeting as evidenced by their signatures below:

[ShareholderName@ 1]

[ShareholderName@ 2]

[ShareholderName@ 3]

 The !GroupName! that you assign to a Repeat block can be shared with subsequent <<*Options*>> blocks carrying the same name. Some restrictions apply. If you want the Repeat value to control a subsequent Option,

1. You must set the Repeat Value first. It must be set via an <<*AskRepeat*> question at the top of the document
2. The number of choices you can set for the <<*Options*>> block can be 3 and only 3. While the actual text can be anything, the nature of the text must be as follows:
 - The first choice must reflect the text you want in the document if the Repeat value is 0.
 - The second choice must reflect the text you want in the document if the Repeat value is 1.
 - The third choice must reflect the text you want in the document if the Repeat value is 2 or more.
 - The 'reason' for this rule is not necessarily obvious, so here is our stab at an explanation:

The primary interplay of a Repeat value to a subsequent Options block is to set noun to verb consistency. In English, 'None *are*', but 'One *is*' and 'Two (and three, four, etc.) *are*.' Plus, words like "in equal shares" would apply to an estate distribution to two or more children, but not to 0 children or to one child. Other examples abound that suggest that 3 choices can address the full range of possibilities.

Further, it simply would not be reasonable (or practical) for a document author, faced with a possible repeat of 100 shareholders, would have to compose 100 separate choices in an Options block.


- Our surveys suggested that 3 is the right number.


Example:

```
<<*AskRepeat*!Children!*>>
```

```
<<*Options*!Children!*There were no children born of our marriage./One child, namely
[ChildName], was born of the marriage/There were <<*Repeat*!children!Return*one>>
children were born of the marriage and their names are <<*Repeat(and)*!children![Child]
>>>>
```

3. Not so much a rule, but strong advice: Don't accidentally reset the Repeat value by creating an *AskOptions* or *AskOptional* prompt with the same groupname. This will cause the Repeat value to be overwritten.
4. You can avoid the above restrictions simply by having an Options !groupname! that is different from the Repeats !groupname!.

 The use of !Groups! is found in other Pathagoras routines. See [Variables\(!Groups!\)](#)^[136] and [Options\(!Groups!\)](#)

 If you want to return to the editing screen the spelled out value of a number that has previously been assigned to a !groupname! for a Repeat, you can do so by enclosing the !groupname! within double angle brackets. E.g., from the above examples, <<!shareholders!>> will return 'three' to the screen if '3' was the answer to the 'How many shareholders are there' prompt higher up in the document.

15.4 Anatomy of a Repeat Block

```
<<*Repeat*!children!Return*one>> children were born of the marriage and their names are <<*Repeat(and)
*!children*! [Child]>>
```

Anatomy:

Two sets of 'facing' angle brackets (the boundary characters) surround the core block.

A !groupname! is used to tie two or more Repeat blocks together.

The 'Return' call tells Pathagoras to 'return' the response when the repeat block is processed. If your response to the "How many . . .?" question is '3', then 'three' will be returned in place of the repeat block. (The 'style' of the number to the right of the asterisk -- in this case 'one' -- is what controls what is returned. If, instead of 'one' the numeral '1' was present, and you selected '3' in response to the 'How many. . .?' question, a '3' would have been returned.) More on this below.

The administrative section is closed by a third asterisk.

The 'one' in the first Repeat block indicates the 'style' of the return. In this example, the number will be typed out instead of it being the actual digit. If you use a digit, a digit will be returned.

In the second Repeat block, the "(and)" tells Pathagoras to use commas as separators between a series of two or more repeated variables, the group name !children! ties back to the first Repeat block to obtain the value.

[Child] is the variable that will be repeated 'X' number of times, each time incremented.

15.5 Arguments ('and', 'or'; default #)

In the previous examples, the text was repeated 'intact ' with no embellishment except the incremented variables.

However, you may wish your 'repeated' text to appear in the final version of your document in a 'series' or 'prose' style display, with commas and a conjunction in the appropriate location , e.g., "[child@1], [child@2], [child@3] and [child@4]".

To accomplish this, you will want to add a 'series' argument to the <<*Repeat*>> prompt.

There are only two possible series arguments: "(and)" and "(or)". Just type the desired argument "(and)" or "(or)" immediately after the '<<*Repeat' keyword and before the closing '*'. (Note: no space between 'Repeat' and the argument.)

Examples:

No series argument:

```
<<*Repeat*[ChildName]>>
```

Result, if "3" repeats chosen:

[ChildName@1] [ChildName@2] [ChildName@3]

With a series argument:

```
<<*Repeat(and)*[ChildName]>>
```

Result, if "3" repeats chosen (note that adding commas is automatic with series argument):

[ChildName@1], [ChildName@2] and [ChildName@3]

With a series argument and a groupname (third '*' is needed to close the administrative text):

```
<<*Repeat(and)*!NumChildren!*[ChildName], [ChildDOB]>>
```

and the same variable(s) elsewhere in document:

```
<<*Repeat(and)*!NumChildren!*[ChildName]>>
```

Result, if "3" repeats chosen:

[ChildName@1], [ChildDOB@1], [ChildName@2], [ChildDOB@2], and
[ChildName@3], [ChildDOB@3],

and elsewhere in document:

[ChildName@1], [ChildName@2] and [ChildName@3]

Note: With series arguments, not only is the conjunction 'and' or 'or' added, but the appropriate commas separating the initial elements in the listing are added.)

Connector punctuation:

If you want to specify the punctuation between elements, you can specify it as part of the 'and' or 'or' argument. E.g.,

```
<<*Repeat(and;)*!NumChildren!*[ChildName], [ChildDOB]>>
```

Result, if "4" repeats chosen:

```
[ChildName@1], [ChildDOB@1]; [ChildName@2], [ChildDOB@2];
[ChildName@3], [ChildDOB@3] and [ChildName@4], [ChildDOB@4]
```

Note: You could also add a ';' (or other punctuation) after the repeating block of text, but that would cause the ';' to appear after the final repetition, something you may not desire.

15.6 Repeating Tables & Rows

Tables:

- If you wish to repeat the contents of an entire table, surround the table with a <<*Repeat* block>>. (In the below example, the groupname !Shareholders! is optional, an needed only if the number of shareholders is also used further down in the document.) E.g.:

<<*Repeat*!Shareholders!How many shareholders?*

[Shareholder]	[Shareholder Owned]	[Shareholder Address]
---------------	---------------------	-----------------------

>>

- If the table contains a header row or rows, you can tell Pathagoras to automatically repeat just the non-headers rows. Here's an example (similar to the above):

<<*Repeat(header)*

In the following section, provide the requested information regarding each child:		
Name of child	Address	Date of Birth
[Child]	[Childaddress]	[Date of Birth]

>>

When you "Process Page" on the above example, Pathagoras will ask for the # of repeats. If Answer=3, the table becomes:

In the following section, provide the requested information regarding each child:		
Name of child	Address	Date of Birth
[Child@ 1]	[Childaddress@ 1]	[Date of Birth@ 1]

[Child@2]	[Childaddress@2]	[Date of Birth@2]
[Child@3]	[Childaddress@3]	[Date of Birth@3]

And after the application of the Instant Database:

In the following section, provide the requested information regarding each child:		
Name of child	Address	Date of Birth
John Doe	123 Main Street	January 28, 1953
Paula Doe Smith	334 Oak Landing	July 7, 1952
Francis Doe	117 Cherry Orchard Land	August 18, 1962

- Repeating a row. A single row (whether within an existing table, just a single row that you created just for this purpose) and be easily repeated without the presence of <<*Repeat*>> instructions. Just follow these steps:
 1. Place your cursor in the row you want to repeat. (In the below example, place cursor anywhere in the last row.)
 2. Click the Pathagoras Features menu.
 3. Click "Process Tools" and then 'Repeat Rows'.
 4. When prompted, type in the number of times you want the row to repeat. All rows at and below the cursor level will be duplicated and all [bracketed variables] will be incremented so they can be identified and answered separately when you run the the Instant Database.


In the following section, provide the requested information regarding each child:		
Name of child	Address	Date of Birth
[Child]	[Childaddress]	[Date of Birth]

Here is the result if “3” chosen:

In the following section, provide the requested information regarding each child:		
Name of child	Address	Date of Birth
[Child@1]	[Childaddress@1]	[Date of Birth@1]

[Child@2]	[Childaddress@2]	[Date of Birth@2]
[Child@3]	[Childaddress@3]	[Date of Birth@3]

Note: Inside rows cannot be duplicated as 'stand alone' units. All rows from the cursor position down will be duplicated. The limitation can easily be overcome by simply not having information you do not want repeated beneath the section of the table you want duplicated. If necessary, split the table.

 If you like this latter (repeat a single row) routine, remember that you can copy the command to the Quick Access Toolbar (QAT) or to the Quick Picks (Alt-Q) table.

15.7 Repeating Within Tables

New 2018.

Pathagoras can repeat the content of a single cell throughout a table. You now can stack repeating content in a vertical table (left side of below sample--the old default result), or you can choose a more horizontal presentation (right side). Further, if variables reside in the repeating text, they are [automatically incremented](#) ²²⁵.

[Client Name@1] [Client Address@1] [Client City, ST ZIP@1]	[Client Name@1] [Client Address@1] [Client City, ST ZIP@1]	[Client Name@2] [Client Address@2] [Client City, ST ZIP@2]	[Client Name@3] [Client Address@3] [Client City, ST ZIP@3]
[Client Name@2] [Client Address@2] [Client City, ST ZIP@2]	[Client Name@4] [Client Address@4] [Client City, ST ZIP@4]	[Client Name@5] [Client Address@5] [Client City, ST ZIP@5]	
[Client Name@3] [Client Address@3] [Client City, ST ZIP@3]	OR		
[Client Name@4] [Client Address@4] [Client City, ST ZIP@4]			
[Client Name@5] [Client Address@5] [Client City, ST ZIP@5]			

Setup:

For a 'stacked' presentation, follow the steps outlined [on this page](#) ²²⁵:

For a horizontal presentation, create a table of a single row. The number of cells in that row should correspond to the number of columns you want the table to contain..

In the first cell (column 1, row 1), type the content you want to be repeated. If you are including variables, type just the 'raw' variable names. Don't increment anything. Pathagoras will take care of

that. (If you want the incrementing number to appear in a certain location, type an ampersand at that spot. (See [Client@ Address] in the sample below.) Otherwise the incrementing number will appear at the end of the variable.

Surround the text you want repeated with the <<*Repeat*. . .>> command. Leave the remaining cells in the table blank. (The empty cells are Pathagoras' cue that you want them filled with the content of Row1/Col1. Don't worry about how many rows you'll need. Pathagoras will calculate that for you and add all necessary rows when the table is processed.

Example:

<<*Repeat*[Client Name] [Client@ Address] [Client@ City, ST ZIP] >>			
---------------------------------------------------------------------------------	--	--	--

If you prefer, you can type the <<*Repeat*. . .>> boundaries outside of the table. Pathagoras will still assume you want to duplicate the content of Row1/Col1.

<<*Repeat*

[Client Name] [Client@ Address] [Client@ City, ST ZIP]			
--------------------------------------------------------------	--	--	--

>>

Usage:

When the <<*Repeat*. . .>> command is encountered (either via the automatic processing during document assembly, or by pressing <Alt-P> to 'force process' the text), Pathagoras will 'see' the repeat block and the blank cells to the right of Row1/Col1. It then asks "How many repeats?" Provide a number. Pathagoras then asks you to confirm that you want to copy the text into the appropriate cells 'left to right.'. With a 'Yes' answer, Pathagoras will quickly duplicate the content into the appropriate number of cells, automatically incrementing the variables in the process. (If you say 'No', Pathagoras will duplicate the text within Row1/Col1 and leave the other cells blank.)

15.8 'Return' the Repeat value

Returning the 'repeat' value. The above examples capture the number of requested repeats, but do not 'return' that value (i.e. '1', '2', 'three', etc.) to the editing screen. If you want to 'return' a value to your editing screen at the same time you make your selection, add the word 'return' in the administrative section of the 'Return' prompt (just after the Repeat command along with the 'style' of the returned number (numeral or spelled out.)

Examples (combining !groups! and 'returns'):

<<*Repeat(return,1)*!shareholders!*>>

<<*Repeat(return,one)*!shareholders!*>>

```
<<*Repeat(return,one(1))*!shareholders!*>>
```

- The "1" (or any other 1 or 2 digit number) after the second "*" tells Pathagoras to return the Arabic numeral;
- The "one" (or any other number) tells Pathagoras to return a "spelled out" number.
- The "one(1)" tells Pathagoras to return the number in this style: one (1)
- Whatever Arabic number you provide becomes the 'default' value when the 'how many' question is asked.
- The command is closed with a final '*' to indicate the close of the [administrative text](#) ¹⁹⁸.

More examples:

```
<<*Repeat(return,one)*!shareholders!*>> individuals own stock in the
corporation:
```

```
<<*Repeat*!shareholders!*Name: [ShareholderName] Shares Owned:
[SharesOwned]
```

```
>>
```

The shareholders waive notification of the upcoming annual meeting as evidenced by their signatures below:

```
<<*Repeat*!shareholders!*_____
[ShareholderName]
```

```
>>
```

Here the result if the answer to "How many shareholders? is 3:

Three individuals own stock in the corporation:

Name: [ShareholderName@ 1] SharesOwned: [SharesOwned@ 1]

Name: [ShareholderName@ 2] SharesOwned: [SharesOwned@ 2]

Name: [ShareholderName@ 3] SharesOwned: [SharesOwned@ 3]

The shareholders waive notification of the upcoming annual meeting as evidenced by their signatures below:

[ShareholderName@ 1]

[ShareholderName@ 2]

[ShareholderName@ 3]

Another Example:

Children (the repeat's 'groupname' is '!cdn!')

Minor Children of the Parties

```
<<*Options*No Children/One Child/2+ Children*
```

```
Husband and Wife have no minor children from the marriage./
```

```
There is one minor child of the parties: [Child1 Name], born [Child1 Birth
Date]./
```

```
There are <<*Repeat(return,two*!cdn!*>> minor children of the parties:
```


```
<<*Repeat*!cdn![ChildName], born [ChildBirthDate]>>
```

```
>>
```

Here is the result of the answer to "How many children? is '4':

Minor Children of the Parties

```
There are four minor children of the parties: [ChildName_1], born [ChildBirth
Date_1] [ChildName_2], born [ChildBirth Date_2] [ChildName_3], born
[ChildBirth Date_3] [ChildName_4], born [ChildBirth Date_4]
```

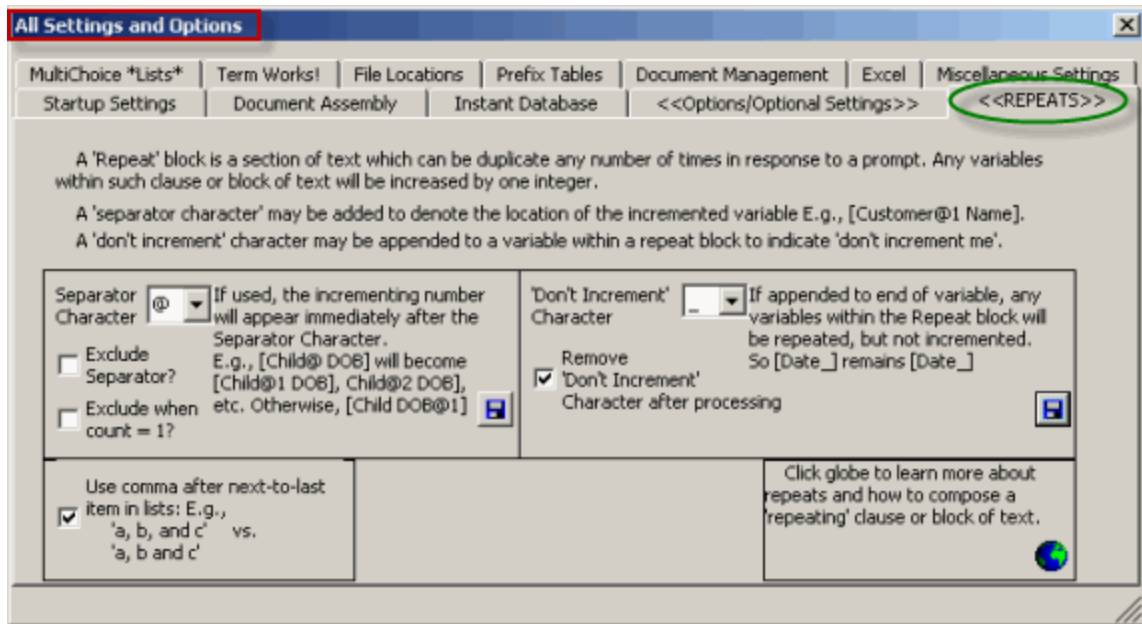
 If a value has previously been assigned to a !groupname! for a Repeat, you can return the spelled out value by simply enclosing the !groupname! within double angle brackets. E.g., from the above examples, <<!shareholders!>> and <<!cdn!>>.

15.9 Repeat Settings

The default 'increment marker' (the character that separates the base variable name from the incremented value) is "@".

Examples: The variable [Child] becomes [Child@1], [Child@2], etc.

You can choose a different marker, or choose not to have a marker at all. Make your choices at the 'Repeats' tab of the 'All Settings and Options' screen.



If you want to exclude the separator altogether, check the Exclude Separator box at the left.

Sample result: [Child1], [Child2], etc.

If you want to change the separator itself, select the desired Separator Character from the dropdown list at the left.

Sample result: [Child_1], [Child_2], etc.

If you don't like the default placement of the @ at the end of the variable, 'tell' Pathagoras where to put the incrementing number by pre-placing the '@' sign in the 'repeat' text block itself. So, if a variable were written [Shareholder@ Address], it would increment [Shareholder@1 Address], [Shareholder@2 Address], etc.

Some variables within a Repeat block (such as [Date]) don't need to be incremented as the text block is being repeated. It will be the same value in all cases. You can tell Pathagoras not to increment a variable by inserting a 'Don't Increment' character as the last character of the variable name. The default 'Don't Increment' character is the underscore. The variable might look like this: [Date_]. (When the document is processed, the underscore will be removed, unless you uncheck the box that tell Pathagoras to do so.)

See Also:

['Repeat' Function \(general\)](#) ^[224]

['Repeat' in Clause Selection Screen and DropDown Lists](#) ^[240]

['Repeat' Alternatives](#) ^[241]

15.10 'AskRepeat'

At or near the top of the document, you can insert a request for the user to provide a 'repeat' value for a designated groupname in this fashion.

<<*AskRepeat*!groupname!*>>. It is simply the 'repeat' keyword and the !groupname!. Nothing else. The entire prompt is 'administrative' which is why it closes with a '*'. Its sole purpose is to ask how many repeats of each !groupname! there are to be. The value is recorded, but nothing is returned to the screen.

You can augment the block with a question or prompt. E.g.:

```
<<*AskRepeat*!numchildren!How many children were born of this marriage?*>>"
```

Since the entire prompt is 'administrative', the entire prompt must close with a '*'.

See also:

[AskRepeat with Options](#) ²³⁷

[GroupNames](#) ²¹⁴

[<<*AskIf*>> prompts](#) ²⁶⁰

15.11 Using 'Repeat' value for Options

EXCEPTION TO THE "POSITION" RULE Normally, !grouped! elements are processed by 'position'. Whether you select the 1st member of the group or the 50th, the same 'position' of other members of the group is returned as the desired value.

However, when the groupname or a <<*Repeat*. . .>>block is paired with an <<*Options*. . .>> block, the selected option is processed based on an analysis of the actual number provided.

Zero: If the 'repeat' number is 0, the option occupying position 1 of the <<*Options*>> block with the same !groupname! is processed.

One: If the 'repeat' number is 1, the option in position 2 of the <<*Options*>> block with the same !groupname! is processed.

Two or more: If the 'repeat' number is 2 or higher, the option text in position 3 of the <<*Options*>> block with the same !groupname! is processed.

Let's illustrate with some sample text. (The logic behind this madness is explained below the block.):

```
<<*AskRepeat*!Children!How many children?*>>
```

```
<<*AskRepeat*!Bene!How many beneficiaries?*>>
```

(The above would appear at the top of the document. Pathagoras always processes 'Ask' blocks first.)

Last Will and Testament of
[Testator Name]

. . .

A. Family: <<*Options*!Children!*I have no children/I have one child whose name is [Name of Child]/I have <<*Repeat(return,two)*!children!*>> children and their names are <<*Repeat(and)*!children!*[Name of Child]>>>>.

. . .

```
<<*Options*!Children!*/
```

G. All trust assets shall be distributed to my child named in paragraph A if that child is alive at the time of my death. If my child is not alive at the time of my death, but has surviving descendants, the share that would have gone to such child shall be distributed to such child's descendants, per stirpes./

G. All trust assets shall be distributed to each child named in paragraph A if that child is alive at the time of my death. If a child of mine is not alive at the time of my death, but has surviving descendants, the share that would have gone to such child shall be distributed to such child's descendants, *per stirpes*.>>

<<*Options*!Children!*

H. All assets shall be distributed to <<*Repeat(and)*!Bene![AltBenef Name] residing at [AltBenef Address]>>, if that beneficiary is alive at the time of my death<<*Options*!Bene!*//,in equal shares>>. If a named beneficiary should not survive me, such shares shall be distributed to such beneficiary's descendants, *per stirpes*./

H. If my child named in paragraph A, and no descendant of such child, shall survive me, all assets shall be distributed to <<*Repeat(and)*!Bene![AltBenef Name] residing at [AltBenef Address]>>, if that beneficiary is alive at the time of my death<<*Options*!Bene!*//,in equal shares>>. If a named beneficiary should not survive me, such shares shall be distributed to such beneficiary's descendants, *per stirpes*./

H. If no child named in paragraph A, and no descendant of any such child, shall survive me, all assets shall be distributed to <<*Repeat(and)*!Bene![AltBenef Name] residing at [AltBenef Address]>>, if that beneficiary is alive at the time of my death<<*Options*!Bene!*//,in equal shares>>. If a named beneficiary should not survive me, such shares shall be distributed to such beneficiary's descendants, *per stirpes*.>>

There is a real logic behind this operation, and it track the noun/verb pairings used in the English language:

If there are 'none' of something, the *plural* of the verb is used to describe this 'nothingness.'

If there is 'one' of something, the singular is used.


If there are '2 or more', the plural is used.

For example:


"There *are* no children. There *is* one child. There *are* 3 children."

Further, the actual text (if any text is provided at all) may vary greatly based on the number of children, shareholders, quantity ordered, etc.

So, when using the same !groupname! with a <<*Repeat*>> block and an <<*Options*>> block, simply set out the appropriate choices for the '0', '1' and '2+', possibilities, separating each choice with slashes.

 If there is no appropriate text to display for one of the choices, you will still need the slash. It then just serves as a holding spot. See just above Paragraph G ('0' children requires no text), and see in Paragraph H near the !bene! group ((there is no 'equal shares' possibility for '0' or '1' beneficiary)).

"But what if I need separate paragraphs based on all of possibilities reflected in the AskRepeat answer?" Sorry, this tool does not address that situation. We are working on a '<<*Options(choose)* . . .>>' tool that will allow you to process a specific result for a specific number. Stay tuned. (And if you have the need for this, write to us. You can help beta test it.)

 **NOTE:** By design, Options blocks will always be processed before Repeat Blocks. This is so even when the Option block is nested within the Repeat block. (In any program, there must always

be a processing order or precedence. Here, with the exception described immediately below, the order is 'Options and Optional' blocks first, and then Repeats.) If you want the document's Repeat blocks to process first, simply present it as an 'AskRepeat' prompt at the top of the document. This is shown in the example.

If you see your Repeat blocks automatically processing when it bears the same groupname as an Options block, that is because of the above rule. Here's what's happening. When the values for the Options block are presented, and you select one, the value for the !groupname! is immediately set. Then, when the Repeat block is encountered, its value is 'known' to Pathagoras and the Repeat block is repeated the 'appropriate' number of times, based on the Options position selected.

The 'cumulative' exception: There is an important exception to the above. If the [\(cumulative\) argument](#)¹⁸², as in <<*Options(cumulative)*!groupname!* . . . >>, the <<*Repeat* value will be first applied to the cumulative, at its original value. So if 5 is the answer, the first 5 choices will be accumulated in the normal 'cumulative' fashion. Any 'non-cumulative' <<*Options* . . . >> will be processed as described above.

Note: if the first or second position in an <<*Options* . . . >> block needs to be blank, simply type a '/' (and no text) to indicate the blank nature of the 'answer'. This is illustrated just below the second ellipse (no first choice) and in the **'in equal shares'** sentence (no first or second choices) of the above example.

15.12 Repeat(merge)

If you need to include data from a data source in a 'string-like' or stacked fashion, the <<*Repeat(merge)*[variables]>> tool may be just the ticket. (We call this 'intra-document' merging. Classic mail merge produces multiple documents from the merge source. The Repeat(merge) simply repeats the text within the scope of the Repeat block and merges data in that more confined area.)

E.g., **Stacked list of names and addresses:**

```
<<*Repeat(merge)*[Client Name]
[Client Address]
[Client City, ST ZIP]
>>
```

can result in:

```
John Doe
123 Main Street
New York, New York 22334
```

```
Jane Roe
334 Oak Court
Raleigh, NC 26678
etc.
```

String list of attendees:

```
<<*Repeat(merge)*[Attendee Name], >>
```

can result in: John Doe, Robert Roberts, Mary Roe, Jane Johnson, Harry Houdini, Bob Marley

When Pathagoras encounters the Repeat(merge) command, it will prompt you for a data source. Those listed will be your preassigned external data sources. (Free-style navigation to an unassigned data source is also possible from the prompt screen.) Select the records you want in your list and press the next button. (You will not be asked for the 'number of repeats' with the (merge) function. The number of repeats is calculated by the program from the number of items you select to merge into your document.) Everything after that is automatic. There will be a perceptible delay as connections to the Excel file are made and variables are replaced, but it won't be overly long.

Notes:

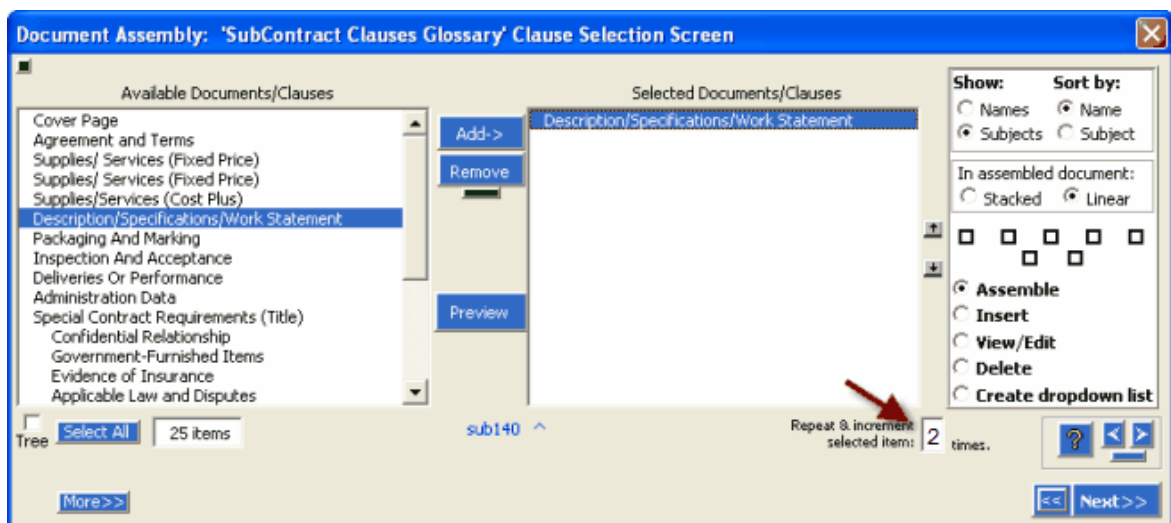
- The data source must be an Excel spreadsheet. (Other data sources will be added later)
- The spreadsheet must contain values in its first row that pair to the variable name(s) in the document.
- The spreadsheet can have more columns than just the variables in the spreadsheet. But each variable must have a corresponding term in the spreadsheet.
- Brackets in the spreadsheet are optional. [Client Name] in the document will be paired to either Client Name or [Client Name] in the spreadsheet.)

More conventional, but still plain text, merge functions are discussed at [this link](#).

15.13 Repeats Elsewhere

 There are two additional 'repeat' functions which Pathagoras makes available:

1. The **Clause Selection Screen** provides a 'Repeats' box that displays when a single non-set element has been selected and moved to the right panel. When completed, Pathagoras will insert the selected clause the designated number of times.



The Repeat Function is available in the Clause Selection Screen when a single element has been selected.

See Also:

['Repeat' Function](#) ²²⁴

['Repeat' Settings](#) ²³⁵

['Repeat' Alternatives](#) ²⁴¹

15.14 Repeat Alternatives

There are other methods available in Pathagoras which mimic the 'Repeats.' They offer more controlled results in terms of 'variable naming.'

- The '**cumulative**' modifier to the <<*Options*>> options is our 'top pick' as an alternative to 'Repeat'. With 'cumulative', you list the choices reflecting an increasingly higher count of 'items'. if you select item 4 in the list, all selections up to and including 4, are inserted, and the remainder are deleted. Pathagoras can even add the proper punctuation (commas and 'and').. See '[Cumulative Options](#)' ¹⁸² for more information and examples.

=====

- You can save what would be the results of a Repeat as separate choices in an <<*Options* . . . text block, e.g.:

```
<<*Options*No children/One child/Two children/Three children/Four children*There were no
children born of this marriage/There was one child born of this marriage, namely [Child1Name],
born [Child1DOB]/There were two children born of this marriage, namely [Child1Name], born
[Child1DOB] and [Child2Name], born [Child2DOB]/There were three children born of this
marriage, namely [Child1Name], born [Child1DOB], [Child2Name], born [Child2DOB], and
[Child3Name], born [Child3DOB]/There were four children born of this marriage, namely
[Child1Name] born [Child1DOB], [Child2Name] born [Child2DOB], [Child3Name], born
[Child3DOB] and [Child4Name], born [Child4DOB]>>.
```

=====

- Another easy and straightforward alternative is to prepare, and save as separate documents, content that reflects the text that would result if you chose 2, 3 or 4, etc., of the element sought to be repeated. .

For example, in a Will, you might have the following alternative clauses reflecting from zero to four children:

[will20_0:](#)

There were no children born of this marriage.

[will20_1:](#)

There was one child born of our marriage, namely [Child1Name] born [Child1DOB].

[will20_2:](#)

There were two children born of our marriage, namely [Child1Name] born [Child1DOB] and [Child2Name] born [Child2DOB].

[will20_3:](#)

There were three children born of our marriage, namely [Child1Name] born [Child1DOB], [Child2Name] born [Child2DOB], and [Child3Name] born [Child3DOB].

[will20_4:](#)

There were four children born of our marriage, namely [Child1Name] born [Child1DOB], [Child2Name] born [Child2DOB], [Child3Name] born [Child3DOB] and [Child4Name] born [Child4DOB].

When you are assembling a clause, simply select the appropriate 'family size' clause from the list of clause options. This technique is straightforward and easy to implement.

15.15 Repeat Restrictions

Nesting:

- A <<*Repeat*>> block can be nested within itself but only to one level.

Structure:

- A 'groupname' for a Repeat block must be enclosed within "!"'s.
 - This structure is legal: <<*Repeat*!Shareholders!* . . . >>
 - This structure is not legal: <<*Repeat*Shareholders* . . . >>
- All text following the closing of the 'administrative text' will be included. So if you wanted to have a group name called '!Shareholders!' but also want the word "Shareholder:" to appear in the repeated text block, just add it after the groupname. Here is an example:

<<*Repeat*!Shareholders!*Shareholder:

[Shareholder]	[SharesOwned
]
[ShareholderAddress]	

>>

See [Administrative text](#)¹⁹⁸.

15.16 Testing Repeat Blocks

You can test the structure, integrity and action of any <<*Repeat*>> block by either 'assembling' a document or by running 'Process' against the text. [Click here to read more about Testing your documents.](#)¹⁹⁷

15.17 Repeat Examples

Lawyers typically have special requirements for the preparation of 'pleadings' and other court-filed documents. Here are some examples of how Repeat blocks can be used to prepare a 'style' of the case.

More samples and examples of captions and signature blocks can be found here as well.

To keep things aligned, you may wish to use a 3 column table, the left side of which contains the parties' names, the center (very narrow) column of which contains the 'squiggles' and the right column of which contains the case info, etc.

Examples are provided below. You should be able to highlight, copy and paste any of these examples into your document. However, some margin and table settings could be lost in the translation. If the copy/paste is not accurate, you can download the identical examples from this URL. [Repeat \(Case Captions\).doc](#)

(Remember, the colors are for illustrative purposes. 'Painted' text is not required.)

```
<<*AskRepeat*!Plaintiffs!*>>
<<*AskRepeat*!Defendants!*>>
```

```
<<*Repeat(and)
*!Plaintiffs!*[PLAINTIFF NAME]
>>
, Plaintiff{!Plaintiffs!/OR/ORs}
vs.
Case Number: [Case No.]

<<*Repeat(and)
*!Defendants!*[DEFENDANT NAME]
>>
, Defendant{!Defendants!/OR/ORs}
```

MOTION FOR RELIEF

Now come your <<*Options*!plaintiffs!*Plaintiff/Plaintiff/Plaintiffs>>, <<*Repeat*!plaintiffs!*[Plaintiff]>> and state the following:

1. . . .

```
=====
<<*AskRepeat*!P!*>>
<<*AskRepeat*!D!*>>
```

The following is the identical setup as the above except we have shortened the groupnames to a single character. (The shorter the groupname, the better the appearance, but the groupname must be 'clear' enough to the actual operator.

```
<<*Repeat(and)*!P!*[PLAINTIFF      )
NAME]                                )
>>                                )

      , Plaintiff{!P!/OR/ORs}

      vs.                            ) Case Number: [Case No.]

<<*Repeat(and)*!D!*[DEFENDANT      )
NAME]                                )
>>                                )

      , Defendant{!D!/OR/ORs}
```

MOTION FOR RELIEF

Now come your <<*Options*!p!*Plaintiff/Plaintiff/Plaintiffs>>, <<*Repeat*!p!*[Plaintiff]>> and state the following:

1. . . .

=====

Here is an example with the split caption frequently used in Bankruptcy courts

<<*AskRepeat*!Debtors!*>>

**IN THE UNITED STATES BANKRUPTCY COURT
FOR THE [EASTERN/WESTERN/NORTHERN/SOUTHERN] DISTRICT OF [STATE]
[DIVISION NAME] DIVISION**

```
<<*Repeat(and)                        )
*!Debtors!*[DEBTOR NAME]              )
>>                                    )
                                    )
                                    )
      , Debtors{!Debtors!/OR/ORs}
```

```
<<*Repeat(and)                        ) CHAPTER 13
*!Debtors!*[DEBTOR NAME]              )
>>                                    ) CASE NO. [Case Number]

      , Movant{!Debtors!/OR/ORs}.
```


vs.)
 [SECOND MORTGAGE HOLDER]) CONTESTED
) PROCEEDING
)
 , Defendant)
)

MOTION TO AVOID LIEN

Now come your Debtor{!Debtors!/OR/ORs}, pursuant to 11 U.S.C. Section 506(a) and (d) and Fed.R.Bankr.P. 3012,

=====

Create a list of shareholders using Pathagoras 'Repeat' function. Tell Pathagoras not to repeat the first row by including the 'header' argument. Include an optional question inside the final 'administrative text' asterisk. When processed, Pathagoras will ask "How many shareholders?". When answered, Pathagoras will repeat the appropriate number of rows, incrementing each variable as appropriate.

<<*Repeat(header)*How many shareholders?*

[Shareholder@]	[Shareholder@Shares]	[Shareholder@Class]

>>

15.18 Repeat (a 'one page' lesson)

Frequently you will have a form document where the number of actors (plaintiffs, defendants, shareholders, beneficiaries, buyers and sellers, etc.) varies from client to client.

Solution 1: It is possible to 'hard code' all the possible variations into the form using Options (e.g., "no children/one child/two children/three children" and so on). If the number of options is fixed and small, that frequently is the best solution.

Solution 2: Where the number of possible options is large or infinite, Solution 1 is not terribly efficient. You should use Pathagoras "Repeat" function in those cases. The "Repeat" function is simple to add, but very powerful in application.

'Repeat' tells Pathagoras to ask "how many actors" to account for. When answered during a document assembly session, it causes Pathagoras to insert a separate, distinct variable for each actor. ('Distinct'

means not only is the variable repeated the proper number of times, but an incrementing number is inserted into the variable name so that it is slightly different from the others in the same set.)

For example, if your basic form includes a variable called [Shareholder Name] and you are drafting a document with 3 shareholders, you might insert the Repeat command into the source document like this:

```
<<*Repeat*[Shareholder Name]>>
```

(Note: the colors are for illustration and emphasis only.
They are not required for the repeat function to properly function.)

At document assembly time, Pathagoras will locate the repeat block, pause and ask for the number of Shareholders. When the end user supplies an answer (let's say it is '3'), Pathagoras will delete the original variable and replace it with designated number of distinct variables:

```
[Shareholder Name1] [Shareholder Name2] [Shareholder Name3].
```

When you are ready to personalize the document using Instant Database, Pathagoras will see the new variables, display them onto the Instant Database screen along with all other document variables for completion.

Can more than one piece of information appear within a Repeat block? Yes. Continuing with the above example, let's say you also want to include addresses and the number of shares for each shareholder. The setup is identical. Just add the more variables as desired:

```
<<*Repeat*[Shareholder Name] [Shareholder address] [number of shares]
```

```
>>
```

(note that the closing '>>' is on a separate line.
That signals Pathagoras to put each variable set on a separate line.)

If the end user tells Pathagoras '3' during document assembly time, Pathagoras will return these lines of text:

```
[Shareholder name1] [Shareholder address1] [number of shares1]
```

```
[Shareholder name2] [Shareholder address2] [number of shares2]
```

```
[Shareholder name3] [Shareholder address3] [number of shares3]
```

Can I repeat an entire document?

Absolutely. Suppose you want to send your shareholder agreement to each shareholder. You can draft a cover letter which will also have incrementing variables so that one letter will be drafted for each shareholder. Your letter might look like this:

```
<<*Repeat*
```

```
[Shareholder Name],
```

```
[Shareholder Address]
```

Dear [Shareholder Salutation],

Enclosed is a draft of the shareholder agreement. Please call me with your comments and questions.

Sincerely,

Arnie Attorney

>>

If the end user tells Pathogoras to generate '3' letters, the result will be a single document contained three distinct letters:

[Shareholder Name1],
[Shareholder Address1]

Dear [Shareholder Salutation1],

Enclosed is a draft of the shareholder agreement. Please call me with your comments and questions.

Sincerely,

Arnie Attorney.

[Shareholder Name2],
[Shareholder Address2]

Dear [Shareholder Salutation2],

Enclosed is a draft of the shareholder agreement. Please call me with your comments and questions.

Sincerely,

Arnie Attorney

.
[Shareholder Name3],
[Shareholder Address3]

Dear [Shareholder Salutation3],

Enclosed is a draft of the shareholder agreement. Please call me with your comments and questions.

Sincerely,

Arnie Attorney

(If you had an End Page or Section break within the Repeat block, they would be duplicated along with the text.)

But wait, there's more!!!

What if I want to generate multiple documents, but there are certain variables that I do not want to be incremented. Can I do that?

Of course you can. You simply need to tell Pathagoras which variables do not increment. Your letter might look like this:

```
<<*Repeat*
[Date of Letter_]

[Shareholder Name],
[Shareholder Address]

Dear [Shareholder Salutation],

Enclosed is a draft of the shareholder agreement. You can see that the
corporation will issue [total number of shares issued_]. You will receive
[number of shares] of those shares. Please call me with your comments and
questions.

Sincerely,

Arnie Attorney.
>>
```

Did you notice the extra _ at the end of [Date of Letter_] and [total number of shares issued_]? That underscore tells Pathagoras not to increment the variable. (The date of the letter and the total number of share issued will always be the same from document to document. By default, Pathagoras will remove the ending '_' during processing.)

In the above examples the incrementing number always appeared at the end. Can I tell Pathagoras to place the incrementing number somewhere else?

Yes. Just put the symbol "@" at the spot in the variable where you want the number to be placed. Eg., [Child@ Name] will become [Child@1 Name], [Child@2Name], [Child@3 Name], etc.

The above examples simply repeated the variable, without any connecting commas and the word 'and' before the final variable.. Can I add list 'punctuation' to the repeat block?

Yes. Just add the appropriate 'list signal' to the Repeat command. Examples:

```
<<*Repeat (and) * [Child]>>
```

becomes (with the answer 3)

```
[Child1], [Child2] and [Child3]
```

```
<<*Repeat(and;)*[Child@ Name], born [Child@ DOB]>>
```

becomes (with the answer 3)

```
[Child@1 Name], born [Child@1 DOB]; [Child@2 Name], born [Child@2 DOB] and  
[Child@3 Name], born [Child@3 DOB]
```

Can I tell Pathagoras to use the same number of repeats at disparate locations in the document?

Yes. Just add a group name to the Repeat block, just after the repeat command. (A groupname is any text between two exclamation marks.) Close the extended 'administrative text' section with a third asterisk. The groupname E.g.,

```
<<*Repeat*!shareholders!*>>
```

```
The following individuals own stock in the corporation:
```

```
    Name
```

```
    Shares Owned
```

```
<<*Repeat*!shareholders!* [ShareholderName], [SharesOwned]
```

```
>>
```

```
(. . .)
```

The shareholders waive notification of the upcoming annual meeting as evidenced by their signatures below:

```
<<*Repeat*!shareholders!*_____
```

```
[ShareholderName]
```

```
>>
```


The Pathagoras System

Interviews / *Ask Tables*

Part



XVI

16 Interviews / *Ask Tables*


A typical 'well Pathagorized' document will likely have many 'Options', 'Optional' and 'Repeat' blocks throughout its content.

In a typical document assembly session, during the 'process' phase Pathagoras will identify the various <<*Options/Optional/Repeat*>> blocks throughout the document. It starts at the top and works its way to the bottom. If a !Group! is encountered, all members of the !Group! further down in the document are processed immediately after the first member of the !Group! is answered.

The drawback to this approach is that each <<*Option/Optional/Repeat*>> block is answered one at a time, as encountered. 'One-at-a-time' is not a 'bad' thing. Depending upon the document, however, it may become tedious.

If you find yourself asking 'Why can't I just answer all the questions at the outset, let the program take over, and then I'll go get a cup of coffee?' then it's time for 'Ask Tables.' 'Ask Tables' are Pathagoras form of interviews. They allow document assembly questions to be asked, answers and processed in one seating.

Beginning with the next section of this Manual, we will introduce the 'Ask' prompts and the parent 'AskTable'. You will learn the elements of the equations, and how to create them yourself. Once you understand the structure, syntax and 'grammar' of these elements, you will find yourself having fun with the programming. But we will also show you how Pathagoras can automatically create these Tables for you.

 Ask Tables, as used by Pathagoras, are not the same as 'Intake Forms' discussed elsewhere.

Intake Forms are designed to gather names, addresses and similar personal information. Intake Forms can be completed by your client/customer/patient to speed up the intake process.

'AskTable' answers, on the other hand, are provided by you or your in-house staff, and often require the technical knowledge you have been retained for to answer. ("What type of Deed? What type of Trust?" or "Is this a springing Power of Attorney?")

Interviews (AskTables) lead to the initial rough drafts of documents. Client data is then applied to personalize the document. Read more about Intake Forms [here](#).

16.1 The <<*Ask. . .*>> Commands

<<*Ask. . .*>> Commands

Sometimes you want to <<*Ask*>> a question of the end user: Something like: "What type of document do you want?" and "Is this a springing Power of Attorney?" and "Are there children?" or "Has condition A, B and C been met?" Based on the answer provided, you want the program to (1) select text to be inserted or (2) set another value (3) ask another question, cascade fashion.

The <<*Ask. . .*>> series of commands handle this logic.

Types of <<*Ask. . .*>> Commands:

Four <<*Ask. . .*>> **Commands** are available in Pathagoras. Collectively they cover the range of possible 'Asks' that an author would need to implement a full Interview (Read more about the structure of Ask commands in the below article called <<*Ask*>> Elements):

```
<<*AskOptions*!Children!No children/One child/2+children*>>
<<*AskOptional*!Minors!Are any children <18 years of age?*>>
<<*AskRepeat*!NumChildren!How many children?*>>
<<*AskValue*!Score!What score did the student receive on the test?*>>
```

<<*Ask. . .*>> commands are in one sense 'parents' to <<*Options*>> and <<*Optional*>> text blocks within the document body that bear the same !GroupName!. (Think here of a parent bird gathering information and feeding it to the chicks. The <<*Ask. . .*>> routines ('parent') gather information 'interview style' and the various responses are assigned to !groupnames!. All items in the document of the same !groupname! (the 'chicks') are fed the answer and processed accordingly.

There are four other important affiliated commands that bring <<*Ask. . .*>> commands to their full potential. They are briefly described below, with links provided to fuller explanations and examples:

1. <<*If²⁶⁰*>>. This adds powerful logic to an <<*Ask. . .*>> command. It can directly set a value to another !Group! and further process the document, or it can be used to pose another <<*Ask. . .*>>. Here is an example:

```
<<*If*!Children!="Yes",<<*AskOptional*!Minors!Are there minor
children*>>,!minors!="False">>
```

2. <<*Set²⁶⁸*>>. The Set command can be used to preset default values or to hard code a value in the Ask table (or any other) area of the document..

```
<<*If*!Children!="No",<<*Set*!Children!=0(#)*>>,>>
```

3. <<*Case²⁸⁴*>>. Used to analyze complex Ask structures involving comparison of multiple elements. This is described more fully [at this page.](#)²⁸⁴


4. <<*Remarks²⁷⁶*>>. Some AskTables can get long. The 'logic' behind some logic steps may not be apparent to the end user (or to yourself after a lengthy period of time, when you decide you want to further develop the AskTable). Use <<*Remarks* . . . >> add comments and explanations to various sections of your AskTables.

5. <<*Break²⁷⁴*>> Normally Pathagoras will display up to 10 'questions' at a time. <<*Break*>> allow you to say "Stop this group of interview questions here," thereby giving you to better control of those questions.

Here are a few additional things you should know about <<*Ask. . .*>> commands at the outset.

- They are totally optional. If the first appearance of an <<*Options/Optional*>> block in the document provides a prompt that is clear enough for the typical user to make the proper selection, you may choose to not use an <<*Ask*>> command.
- <<*Ask. . .*>> commands are typically placed at the top of the document. That way, they can be grouped together to form an 'interview' at the start of the assembly process. (It is also easier to edit questions you will be asking your end users when they are grouped together.)

- `<<*Ask. . .*>` commands are all 'plain text.' As such, you can easily copy a good set of `<<*Ask*>` commands from one document to another. Doing so will help you to make the questions uniform from document to document.
- If you use `<<*Ask. . .*>` commands, you can get rid of the prompts that you may have typed within your `<<*Options/Optional*>` blocks. This may result in a significant savings of document 'real estate'.

 `<<*Ask* . . .>`, `<<*If* . . .>` and `<<*Set* . . .>` commands are typically located at the top of the document. That way, it is the first thing that the end-user will encounter during a document assembly session.

However, `<<*Ask* . . .>` commands are usually created *last* in the **Pathagorizing** process. The reason is simple: unless you know what the `<<*Options . . . Optional . . . Repeats*>` text is in the document's body, you will not know what to even "Ask".

So as you study and try to mimic the examples on the following pages, please don't even try to create the `<<*Ask*>` commands until you have thoroughly composed and tested the routines in the document body. (And because these commands are optional, you certainly don't need them in the initial stages of source document development.)

Benefits of `<<*Ask*>` commands:

When you add prompts to an `<<*Options*>` block, the amount of 'real estate' it consumes can be substantial. When that block appears in the middle of a paragraph, it can be difficult to visually process (both to the editor and the end user who may peruse the document before processing it.)

There is something intangibly better about questions being at the top of, and segregated from the body of, the main text.

Best reason: Once you have your 'best' questions composed and residing nicely within one of your documents, you can copy and paste an entire collection of `<<*Ask*>` commands into any other document containing similar `<<*Options/Optional*>` blocks. This assures a consistency in the questions that are posed to the end user.

For documents-by-building-block aficionados, you can also save the AskTable as a separate clause, calling it in along with other clauses to build, and then process, the document.

Try that with other programs!

16.2 `<<*Ask*>` Elements

An `<<*Ask. . .*>` command requires the following elements:

- Opening '`<<`' and closing '`>>`' to designate the scope of the command. (Remember: the red, blue, cyan and green colors used in these examples are not important. The color is only for emphasis and clarity.)
- The term `"*AskOptions*"`, `"*AskOptional*"`, `"*AskRepeat*"`, `"*AskValue*"` (depending upon the nature of the command) followed by the `!groupname!`. Of course, this `!groupname!` will tie the answer to the `*Ask*` to the related Options/Optional/Repeat blocks in the body of the document.

- **<<*AskOptions*>>** blocks: list the choices you want to present to the end user. Separate each choice with a '/'. Close the list with a '*'.

E.g.,

```
<<*AskOptions*!shipmethod!FedEx/UPS/USPS*>>
```

and later in the document:

Your order will be sent by <<*Options*!shipmethod!Federal Express (2 day delivery)/United Parcel Service (ground; allow 4 days for delivery)/US Postal Service (1st class mail).>>

- **<<*AskOptional*>>** blocks: pose the question that you want the end user to answer. Close the question with a '*'.

E.g.,

```
<<*AskOptional*!freeshipping!Will shipping be free?*>>
```

and later in the document

```
<<*Optional*!freeshipping!*Plus, if you place your order in the next 30 days, your shipping is free!>>
```

- For **<<*AskRepeat*>>** blocks, pose the question that you want the end user to answer. Close the question with a '*'.

E.g.,

```
<<*AskRepeat*!NumCdn!*How many children?*>>
```

 See discussion on how Pathagoras handles the repeats 'count' with Options blocks beginning [this page](#)²³⁷ of the Manual.

- For **<<*AskValue*>>** blocks, you can ask a question that you want the end user to answer. Close the question with a '"'.

E.g.,

```
<<*AskValue*!Score!*>> or
```

```
<<*AskValue*!Score!Test Score?*>>
```

and later in the document

```
<<*Options*!Score!0-59/60-69/70-79/80-89/90-100*F/D/C/B/A>>
```

i Okay, why the final '*'? Just like in a regular **<<*Options/Optional*>>** block, the asterisk closes the administrative section. The administrative section includes any prompts/questions that

are posed. In an `<<*Ask*>>` block, it is all 'administrative text' since none of it remains in the document. Hence the need for the final '*'.)

i Each Ask command must reside on its own line. It cannot reside on the same line as another Ask command, nor can any other text be combined with it.

i The `<<*Ask` commands, collectively, constitute the Interview phase of document assembly. As such, the commands should appear at the top of the document. The Ask table can also reside in a separate document that you call in as a 'building block.' You should avoid interspersing Ask commands within the document body. Pathagoras will likely not crash if you do, but it is not considered a best practice.

i `<<*Ask` commands ask and record values for !groups!. They serve no other purpose. Therefore, every `<<*Ask*` command must contain a !groupname!. Absent a !groupname!, an Ask is powerless to do anything in the document, and will fail. (You will also get an error message.)

16.3 (Arguments)

There are several 'options' (termed 'arguments' in programming jargon) that provide you flexibility in how you present the desired interview question to the end user. Arguments appear immediately after the base command, inside of parentheses, and before the '*' that closes the command.

`<<*AskOptions(radio)* . . .`

The standard `<<*AskOption* . . .>>` will present to the end user a series of check boxes from which the user can select one or more of the options.

`<<*AskOptions(radio)*. . .>>` presents a series of radio buttons from which the user can select only one of the options.

`<<*AskOptions(connector)*. . .>>`

The (connector) argument adds to the bottom of the question box a group of connectors: "And", "Or" and "Enter". The items chosen from the listing are presented in a linear fashion with commas and the selected connectors interposed between the selections.

Note: Use "(connector)" only when you want all `<<*Options*` blocks in the main document (of the same !group!, of course) to follow the designated pattern. If one or more of the `<<*Options*` blocks will not be following the pattern, use the '(and)' or '(or)' arguments in the body text. See ['\(and\)' and '\(or\)' arguments](#).¹⁸⁰

`<<*AskRepeat(3)* . . .`

To set a default number in the AskRepeat question, insert any number from 0 to 9 as an argument. (If no argument is provided, the default will be '1'.

16.3.1 Requiring an Answer

Compare `<<*AskOptional*>>` and `<<*AskOptions(radio)*>>`

Let's say you are drafting a Will. How might you "`<<*Ask. . .>>`" if there are minor children?

This works great: `<<*AskOptional*!Minors!Are any children under 18 years of age?*>>`

When processed, the user will be presented with the question and a checkbox to fill or not fill. (Checking the box means 'Yes' or 'True.' Leaving it unchecked means 'No' or 'False'.)

But what if the answer should have been 'Yes' but the user forgot to check it (or thought he/she did, but didn't). Then !Minors! would be set to 'False' and some results might be messed up.

To avoid this possibility, you might prefer to rewrite the question in such a way as to force an affirmative response. This can be done via the `<<*AskOptions(radio)>>` command.

Here is a recast version of the above which forces the user to choose an answer before processing will continue:

`<<*AskOptions(radio)*!Minors!Minor Children/No Minor Children*>>`

It is not so much the 'AskOptions' that does it, but the 'radio' argument. Without 'radio,' two check boxes will be presented on the screen: "Minor Children" and "No Minor Children". Both could be left blank, defeating the purpose behind this exercise. Only 'radio' requires the user to provide an answer.)

16.4 <<*AskOptions*>> additional results

When an `<<*AskOptions. . .*>>` block is answered during a document assembly session, Pathagoras assigns and records the positional value of the selected item(s) to the !GroupName! of the equation. Pathagoras then creates a new set of !GroupNames! and implicitly assigns a 'True' (or 'False') value to each selected (or not selected) value. (The !GroupNames! created carry the exact names of the `<<*AskOptions . . .>>` choices. This allows you to insert an `<<*Optional*!groupname!* . . . >>` in the document body.

See [this page](#)^[292] for more information and examples.

16.5 <<*AskValueInRange*>> Command

The `<<*AskOptions*>>` command returns a 'fixed' result based on the position of the 'choice' in the list you provided. If you chose 'bananas' from the choices apples/banana/cherries, Pathagoras only knows that you chose the second item.

Sometimes this 'positional' approach will not work. Sometimes the actual value (and not its position in the list) is needed to control the result. If the answer to a question falls within a range of values, as opposed to being a specific value, you need a different approach. That's where the `<<*AskValue*>>` command comes into play.

Example:

Grades: An 'A' is a range from 90 to 100, a 'B' ranges from 80-89, etc. A score that deserves Honor Roll recognition is any score of 85 and above.

Let's assume a test score of 88. You conceivably could have several lists of 101 choices each (all tied together with a !groupname! so they change in tandem), one list representing the actual numeric score (e.g., 100/99/98/. . . /90/89/88/. . . /0), another 100 element list representing the letter grades by position (A/A/A/. . . /A/B/B . . . /F), and 100 choices representing whether a particular grade

made the honor rolls (Honor roll/Honor roll/ . . . /Not honor roll/ . . . etc). But that borders on the absurd.

Enter the `<<*AskValueInRange*>>` command, Pathagoras 'Asks' for a specific value of 'something' (grade, quantity, score, etc.) at the top of the document. It will then analyze the response per a variety of 'ranges' that you provides elsewhere in the document. As with other `<<*Ask*>>` commands, Pathagoras requires you to provide a `!groupname!` to associate it with other elements in the document.

The AskValue method. Example #1

```
<<*AskValueInRange*!Grade!Grade Awarded*>>
```

- a. Your end of semester grades have been released.
- b. Your raw score for the semester was `<<!Grade!>>`. This translates to a letter grade of `<<*Options*!Grade!0-59/60-69/70-79/80-89/90-100*F/D/C/B/A*>>`.
- c. `<<*Options*!Grade!0-84/85-100*/Congratulations! You have made the honor roll.>>`

*(You can copy and paste the above sample into your own document and 'process' it.)
(Note: in line 'c' there are 2 options. The 'no text' before the first slash after the close of the administrative text is a choice. Just a blank one.)*

When the user assigns a value in response to the `<<*AskValueInRange*>>` command, Pathagoras remembers its value and applies what it knows in one of two situations:

1. If a `!groupname!` ontained within double angle brackets (written like this: `<<!groupname!>>`) is located within the document, it will be directly replaced with the raw value. This is shown in line b. above.
2. When you have provided a 'full' `<<*Options* . . .>>` block that contains value 'ranges' within the administrative section of the block, Pathagoras will analyze the answer using those ranges and return the appropriate value. This is shown in lines b. and c. above.

Note *(and this illustrates the very important distinction between `<<*AskValueInRange*>>` and `<<*AskOptions*>>`):* The element count need not be the same within various member of the AskValueInRange group member. -- note that 5 options are provided in line b. above. Only 2 choices are provided in line c. (Note also, in line c, the result of the first choice -- the grade is between 0 and 84 -- is a 'blank'. There is no text between the asterisk that closes the administrative text and the first slash, meaning 'type nothing'. Therefore, no 'honor roll' award is presented.)

The AskValueInRange method. Example #2

```
<<*AskValueInRange*!Total!Total of Order*>>
```

Your order totals \$`<<!Total!>>`. `<<*Options*!Total!0-75/>75*`Your purchase does not qualify you

for free shipping./Congratulations! You are eligible for free shipping of your order. Select your preferred shipping method below.>>

```
<<*Options*!Total!<=74.99/>=75*/<<*Options*!Free Shipping Method!*FedEx/UPS/USPS>>>>
```


(You can copy and paste the above sample into your document and 'process' it.)

The above example provides a different situation where `<<*AskValueInRange*>>` might be used. It also illustrates that non-finite ranges can be used. Note the '>75' in both lines. You can use '>' and '<' and '<=' and '>=' (The order of the signs does not matter. So you could have '>=' as well.)

In the area of direct value replacement, you have to decide whether the value is best provided using the Instant Database or via the AskValue command.

When `<<*AskValueInRange*>>` is better than [Variables]:

- With `<<*AskValueInRange*>>`, the presumption is that the value will be analyzed in multiple locations in the document with varying sets of ranges. (This is not a requirement, but is the reason why the routine was written. You can use `<<*AskValueInRange*>>` to replace `<<*AskOptions*>>`, but the setup for AskOptions is quite a bit easier.
- Use `<<*AskValueInRange*>>` where there is no pressing need to save the value as part of an Instant Database record. You want to save the customers name, address, etc., so that you can use that information in letter after letter, but you may not need to record (at least for document assembly purposes) the specific grade or the amount of the purchase that leads to the generation of the document. (That data presumably is recorded elsewhere.)

 Note the following structural requirements:

The `<<*AskValueInRange*>>` command, like all `<<*Ask . . .>>` commands, closes with a `"*"`.

The comparison must be between a minimum of two sets of values. Those values can be a finite range (e.g., 0-75) or a non-finite range (e.g., >75; >=75 will also work).

Each 'range' option must have a corresponding 'return' option. So, in this example:

```
<<*Options*!Grade!0-89/90-100*Sorry, you did not make the
honor roll/Congratulations! You have made the honor roll.>>
```

there are two ranges: '0-89' and '90-100'. The 'return' text is whatever appears after the `"*"` that closes the administrative section of the block.

Let's provide one more example. There is no sense in 'rubbing in' the fact that the 75 grade student did not make the honor roll. Perhaps it is best just to say nothing.

```
<<*Options*!Grade!0-89/90-100*/Congratulations! You have made
the honor roll.>>
```


In the above example, `"/Congratulations! You have made the honor roll"` is now the return text. The slash, with no text preceding it, means it is a 'blank' choice. It returns 'nothing'.

Here is a psychological example, returning the IQ Classifications for the **Wechsler Intelligence Scale for Children–Fifth Edition (WISC-V)**:

```
<<*Options*<69/70-79/80-89/90-109/110-119/120-129/>130*Extremely Low/Very Low/Low  
Average/Average/High Average/Very High/Extremely High>>
```


When a [Variable] is better:

- When the result is a defined value needed only for a direct replacement and need not be analyzed for use elsewhere in the document and it is imperative that you save the value in the client/customer's Instant Database information.

 **NOTE:** <<*AskValueInRange*>> commands must be manually created in the AskTable. If you have structured an <<*Options*>> block in the body of the document that is being scanned which contains ranges of values, Pathagoras will create an <<*AskOptions*>> command at the top, with the ranges reflected as the alternative choices. Simply change <<*AskOptions*>> to <<*AskValueInRange*>> and either replace the alternative choices with a simple question or just close the block with a "*".

```
<<*Get*!Grade!
```

16.6 Hover-over Text

 **Hover-Over text:** Sometimes even the question you pose in the AskTable doesn't perfectly communicate what you intent to the end user. You can provide up to 256 more characters of 'hover-over' text for each choice that is presented. This hover-over text will appear when the user moves the cursor over each of the various section in the AskTable. It will display only during the time the cursor is 'hovering over' the selection.

To add hover-over text, simple type a '+' sign plus the hover-over text at the end of each option. Make sure the hover-over text is within the administrative section of Options block (i.e., before the third asterisk).

```
<<*AskOptions*!TypePOA!Springing+Use springing when power effective only upon disability of  
grantor/Immediate+Effective when signed and delivered*.>>
```

See also: [Hover-over Options](#)¹⁷⁹

Hover-over text for variables

16.7 <<*If*...>> Command

While <<*Ask*>> commands discussed above may constitute the bulk of your Interviews, it is the <<*If*>> equation that will form the backbone of the document automation process. It is the <<*If*>> equation that lets you declare comparisons and to set up your decision and branching points.

Let's say you are drafting a document that has optional text dealing with children. You want the text to be kept in the document when there are children and discarded when there are none. Let's further say that the document contains other language that you want to keep when there are *minor* children, but which should be discarded where there are no minors to provide for.

Using `<<*AskOptional*>` commands, you might ask the question "Are there children?". If (but only if) that question is answered "Yes", you might then ask "Are there minor children?". (Of course, if there were no children, there couldn't be any minor children; therefore the question should not be asked..)

Here is the way this sequence might appear:

```
<<*AskOptional*!children!Are there children?*>>
<<*If*!children!="Yes",<<*AskOptional*!minors!Are there minor
children*>>,!minors!="False">>
```

i NOTE: The values "Yes", "True" and "1" are functional equivalents. The above could have been written:

```
<<*AskOptional*!children!Are there children?*>>
<<*If*!children!="True",<<*AskOptional*!minors!Are there minor
children*>>,!minors!="False">>
```

Let's study the structure.

1. Note first that a `!groupname!` (in this case `!children!`) is used. As with other `!group!`, the name ties the various elements of the document together.
2. Now note the classic three-part "if . . . then" programming structure in line 2:

"If Condition, (If 'True' Action), (If 'False' Action)"

This three-part logic structure is practically universal. Simple commas are used to separate the three elements.

As applied to the example, the user first encounters the initial (Children) question. The user provides either a "True" or "False" (or 'Yes' or 'No') answer. The "If" command is encountered immediately after. Pathagoras 'measures' the value of the `!Children!` group, and responds accordingly. If `!Children!` was answered "Yes" (by checking the check-box), Pathagoras presents the prompt that appears between the first comma and the second comma. If Pathagoras sees that `!Children!` was not answered "Yes", it processes the information that follows the second comma. This 'logic' follows classic 'Boolean' logic, which 'reads': If (this) is true, then (do this), else (do this).

It is perfectly acceptable (sometimes) to leave the 'False' portion of the 'If' statement blank:

```
<<*If*!Children!="Yes",<<*AskOptional*!Minors!Are there minor children*>>,>>
```

See [Bigger Example](#)²⁹⁵ for more examples of the `<<*If*>` command.

Setting a True/False Value

Setting a 'True/False' value (<<*Optional* . . .>>):

<<*Optional* commands are processed based on simple 'true' and 'false' values of a !group!. And the group can be set via an <<*If* statement, and several examples above demonstrate that. Here is a repeat of one:

```
<<*If*!Children!="Yes",<<*AskOptional*!Minors!Are there minor
children*>>,!minors!="False">>
```

Repeats: If you need to assign a Repeat value of one GroupName to that of another, you can make that assignment with the !Group1!=!Group2! command. E.g.,

```
<<*AskOptions*!OurClient!Plaintiff/Defendant*>>
<<*AskRepeat*!Plaintiffs!>>
<<*AskRepeats*!Defendants!>>
<<*If*!OurClient!="I",!Clients!=!Plaintiffs!,!Clients!=!Defendants!>>
```

Depending upon the various assignments, if your client is the defendant, the number of repetitions of [Client Name] in the document will be equal to the number of Defendants selected at the top.

Debugging 'If's

When your 'If' statements seem not to give the expected results, check these elements:

The sequence of the 'True' and 'False' portions

The sequence of a precedent Ask commands that sets the value of the 'If' comparator.

The proper spelling of the !GroupNames!.

Is the 'False' value of the statement set? [See above](#)²⁶¹ for discussion.

NOTES:

1. The <<*If command, and its various parts (the 'True' part and the 'False' part and its mandatory structural elements) is definitely 'programming language. Therefore, If's will be use only within the Ask table portion of a document where programming is allowed.
2. If an Ask table contains an '<<*If*. . .>>' logic point for which the <<*If*>>'s !groupname! has no value (because, most likely it was asked in an immediately preceding <<*Ask*>> and therefore has not been set), Pathagoras will not proceed further until the precedent value is assigned. You can also break processing until the !Groupname! is set using the <<*Break*>>²⁷⁴ command.
3. When a number represents a position in a list, it is enclosed in quotes. When the number represents an actual value (i.e., 5 means 5 pets in the above examples), it is not enclosed in quotes.
4. You can set multiple values with the <<*If*>> command. Simply separate each item with the pipe (!) character. E.g.

```
<<*If!*Type of Deed!="2",!QuitClaim!="True"!!Warranty!="False",!QuitClaim="False"|
!Warranty!="True">>
```

(The comma (in red) still separates the 'if true' and 'if false' parts of the master equation. The pipe tells Pathagoras to split the true and the false parts into multiple actions.

5. You can set multiple groupnames to the same value within the <<*If*>> command. List them within the same ! and ! boundaries. E.g.,

```
<<*If!*Type of Deed!="3",!Warranty!="True"!!Quit Claim,Special Warranty,No
Warranty!="False",>>
```

5. The '<<*If' command can accept the following arguments:

```
(any)--<<*If*(any)!*Type Of Deed!="1,3,4,6",!Title Search Done!="True",!Title Search
Done!="False">>
```

```
(all)--<<*If*(all)*!Selection!="3,4,6,9,10",!Vegan!="True",!Vegan!="False">>
```

```
(not)--<<If*(not)*!States!="1,7",!Continental Tour Only!="True",!Continental Tour
Only!="False">>
```

See next page for more examples.

<<*If logic can be tough stuff. If you are nesting equations, adding multiple comparators and augmenting the equations with arguments and lists of possibilities, it will take you some time to get it all right. Don't get frustrated. Keep on trying. When testing, don't test on the entire document. Break out the relevant sections and put them in a new document for testing. When its working as expected, then move it back into your main document for 'full' testing.

16.7.1 Cascading Logic

The value of a precedent operation can be used to determine in a highly focused manner a subsequent values. The process by which the answer to Question 1 leads to a relevant Question 2, the answer to which leads to an answer or relevant Question 3, etc., is called 'cascading'.

By way of illustration, let's assume that a 'Last Will' is the objective. The initial questions that might need answering a 'What is the sex of Client? (Male or Female)', 'Is the Client Married? (Yes or No)' and 'Does Client have children?' With that information, other decisions can be automatically set or other relevant questions posed.

```
<<*AskOptions(radio)*!T-orSex!Testator = Male/Testator = Female*>>
<<*AskOptions(radio)*!Client Married!Client is Married/Client is Single*>>
<<*AskOptions(radio)*!HasChildren!Client has children/Client has no children*>>
<<*AskOptional*!Minor Children!Are any children minors*>>
<<*AskOptional*!Guardian!Appoint Guardian for minors?*>>
<<*AskOptions*!UGMA trust!UGMA Trust/No UGMA Trust*>>
```

While the above setup makes sense, the last 3 questions are only relevant if there are children. Cascading logic avoids the waste of time reading and answering irrelevant question. Here are 'cascading' questions

```
<<*AskOptions(radio)*!T-orSex!Testator = Male/Testator = Female*>>
<<*AskOptions(radio)*!Client Married!Client is Married/Client is Single*>>
<<*AskOptions(radio)*!HasChildren!Client has children/Client has no children*>>
<<*If*!Client has children!="True",<<*AskOptional*!Minor Children!Are any children minors*>>,!Minor
Children!="False">>
<<*If*!Minor Children!="True",<<*AskOptional*!Guardian!Appoint Guardian for minors?*
>>, !Guardian!="False">>
<<*If*!Minor Children!="True",<<*AskOptions(radio)*!UGMA!UGMA trust/No UGMA
trust*>>,!UGMA!="False">>
```

In the above example, the 'minors' questions are asked only if there are children. The 'guardian' and 'UGMA' questions are asked only (1) there are children and (2) one or more of those children are minors. Otherwise, !Guardian! and !UGMA! are set to "False." (!Minor Children! was set to 'False' near the beginning if there were no children to begin with). (Note also that for !Guardian! we asked an 'AskOptional' to determine its value and for !UGMA! we presented a two part 'AskOptions'. This was to demonstrate that it does not matter which you use. AskOptional takes up less space, but it could accidentally be left unchecked when 'checked' was the proper response. Some folks prefer the mandatory response of 'AskOptions(radio)').

So, the next question that naturally arises is "Why the need to affirmatively assign 'False' to the values for !Minor Children!, !Guardian! and !UGMA!." [Read answer here.](#)^[276]

16.7.2 Math within <<*If*...>>

Pathagoras can do a bit of math for you if you desire. When a !GroupValue! has been set to a numerical amount, either through an <<*AskRepeat*>> or a <<*Set*>> function, you can perform a math function on the values. If the value matches the comparison set up, then the 'True' part of the <<*If*>> is processed, otherwise, the 'False' part is.

```
<<*If*!NumCdn!+!NumPets!>5,!CrazyHouse!="True",!CrazyHouse!="False">>
```

16.7.3 Multiple Comparitors

Math (discussed previously) lets you add values assigned to !groups!. But what if you just want to ask "If This AND This are true (or If This OR This are true), then do this.

That is possible. Just use the 'AND' or the 'OR' connector:

```
<<*If*!NumCdn!>2 AND !NumPets!>2,!CrazyHouse!="True",!CrazyHouse!="False">>
<<*If*!NumCdn!>4 OR !NumPets!>4,!CrazyHouse!="True",!CrazyHouse!="False">>
```

More examples:

```
<<*AskOptions*!state!*States*>>
<<*AskOptions(radio)*!Size!Small/Medium/Large/Custom*>>
<<*AskOptions(radio)*!Color!Red/Blue/Green/Custom*>>
<<*If*!Size!="Custom"AND !Color! = "Custom",<<*AskOptions(radio)
*!ExtraCharge!$100/$150/$200*>>,>>
```

```

<<*If*!Size!="Custom"AND !Color! ~ "Custom",<<*AskOptions(radio)
*!ExtraCharge!$50/$60/$70*>>,>>
<<*If*!Size!="Custom" OR !Color! = "Custom",<<*AskOptions(radio)
*!Shipping!$20/$30/$40*>>,>>
<<*If*!State!#"California",<<*AskOptions(radio)*!Shipping!$20/$30/$40*>>,>>

```

In the above example, we used 'hard' (as opposed to positional) values for the comparisons.

'Legal' comparators

'='

'~' ('not equal')

'#' ('includes'; used when multiple items can be selected, and you are checking to see if the named item was one of them.)

'Legal' connectors:

'AND' or 'and'

'OR' or 'or'

'+' (math only)

'-' (math only)

This example addresses what a manufacturing company might face in processing special orders. There are extra manufacturing charges and extra shipping charges associated with custom orders. Here the ~ (not equal) is illustrated.

```

<<*AskOptions(radio)*!Size!Small/Medium/Large/Custom*>>
<<*AskOptions(radio)*!Color!Red/Blue/Green/Custom*>>
<<*If*!Size!="Custom"AND !Color! = "Custom",<<*AskOptions*!ExtraCharge!$50/$60/$70*>>,>>
<<*If*!Size!="Custom"AND !Color! ~ "Custom",<<*AskOptions*!ExtraCharge!$100/$150/$200*>>,>>
<<*If*!Size!="Custom"OR !Color! = "Custom",<<*AskOptions(radio)*!Shipping!$20/$30/$40*>>,>>

```

Limitations

While, as shown above, you can have multiple comparitors (as many as you wish), you cannot mix ANDs and ORs in the same equation. You cannot use paraentheses to group items for an initial 'small' comparison before the 'large' comparison. So you can have:

```

<<*If*!Size!="Custom"AND !Color! =
"Custom"AND!Fabric!="Silk",<<*AskOptions*!ExtraCharge!$70/$80/$90*>>,>>

```

But, this is not allowed:

```

<<*If*!Size!="Custom"AND !Color! =
"Custom"OR!Fabric!="Silk",<<*AskOptions*!ExtraCharge!$50/$60/$70*>>,>>

```

You can write two separate <<*IF* . . . >> equations, however, to accomplish the same result.

16.7.4 Multiple 'True' or 'False' Results

New in v. 2020:

A True or False result can lead to multiple events. (An 'event' would be to 'set a value' or to 'ask another question'.) You can either stack results (if that visually results in a better interview), or you can combine the results. To combine them, just insert a pipe character: '|' between the results.

```
<<*AskOptions*!children!No children/Yes, but all adults/Yes, some still minors*>>
<<*If*!children!="3",!Guardian Clause!="True"|<<*AskOptional*!UGMA!Include UGMA
Clause?*>>,>>
```

Decoded:

1. Ask whether there are children. As you can see, 3 choices are presented.
2. If the answer to the Ask is "3" (i.e., the third choice), then (a) set !Guardian Clause! group to "True" (the 'Set' command is understood) and also ask whether the UGMA clause should be included.
3. If the answer to the Ask is not "3", do nothing more.

16.7.5 <<*If(arguments)* . . .>>

New in v. 2020:

The <<*AskOptions* . . .>> can present many choices. Imagine the fifty United States as a list of choices. Next imagine that you want Pathagoras to perform certain actions if Florida, Georgia and/or South Carolina are selected, and another set of actions if Washington, Oregon and/or California are selected.

Pathagoras' <<*If(any)* . . .>> and <<*If(all)* . . .>> commands will make this possible. (Before introduction of the 'any' and 'all' arguments, you could 'stack' multiple <<If* commands to accomplish the same ends. 'Any' and 'all' simplify the process tremendously.

Here is perhaps a more practical (law office based) example.

```
<<*AskOptions*!TypeDeed!Standard Deed/Deed of Gift/Marital Settlement Deed/Deed of
Trust*>>
```

Some deeds are taxable, others are not. Let's say you want to set a groupname called '!Taxable!' and assign a 'True' or 'False' value (so that the 'tax' language will appear, or will be omitted, depending upon the selection made).

You could do this:

```
<<*If*!TypeDeed!="1" OR !TypeDeed!="1",!Taxable!="True",!Taxable!="False">>
```

but the equation can be shortened with the '(any)' argument, thusly:

```
<<*If(any)*!TypeDeed!="1,4",!Taxable!="True",!Taxable!="False">>
```

(Note that the reference of the choices is to their respective positions in the list, not the actual text.)

As there is an unlimited number of AskOptions equations of unlimited length (again, picture the 50 United States), you can see how an '(any)' or '(all)' or '(not)' argument can save much space and avoid many typos.

Anatomy:

- `<<*If(any)*!Groupname!(choices, separated by commas),!Taxable!="True",!Taxable!="False">>`
- The command `<<*IF` followed by the appropriate argument 'all', 'any' or 'not' (in parenthesis), closed by the '*'
- The `!Groupname!` surrounded by '!'s
- The list of choices to be compared, enclosed in quotes, separated by commas.
- A comma that separates the 'If' part of the equation from the 'then' parts.
- The If true part.
- A comma that separates the 'If True' part from the 'If False' part.
- The If false part
- The closing boundary mark `'>>'`

Another AskTable Example:

```
<<*AskOptions(radio)*!Type of Deed!Standard Deed/Deed of Gift/Marital Settlement Deed/Deed of Trust*>>
```

```
<<*If(any)*!Type of Deed!"1,5",!Taxable!="True",!Taxable!="False">>
```

Body text:

```
{!Taxable!The tax due on the recording of this deed is $[Tax Due]/NEGOPTNo tax is due upon the recording of this deed.}
```

(You can copy and paste the above into a Word document and press Alt-P to process it. You can modify it as desired. Remember, Pathagoras plain text means no fields to type, no fields to break.)

Notes:

You can have a string of If(. .) comparisons but they must be of the same type. You can have

```
<<*If(and)!Color!="1,3,5" and !Size!="3,4",. . .>>
```

but you cannot (at present) change in midstream to

```
<<*If(and)*!Color!="1,3,5" and *If(all)!Size!="3,4",. . .>>
```

The opening `<<*If(and)*` modifies all comparisons)

(The rule of not mixing 'and' and 'or' in the same equation also applies.)

16.7.6 <<*If. . .>> formatting**New in v. 2020:**

Most of the previous examples formatted the `<<*If` equations in a strictly linear array. But (beginning with v. 2020.40), you can (and perhaps should for your more complex equations) format such equations in such a way as to make them more readable to a casual scan by a user. You can add an Enter between the major sections and you can also indent the True and the False parts. The comma that separates the initial `<<*If. . .>>` equation from the True part, and the comma that separates the True part from the False part must still be included.

Example:

```
<<*AskOptions*!TypeDeed!Standard Deed/Deed of Gift/Marital Settlement Deed/Deed of Trust*>>
```

Each of the below presentations of <<*If. . .>> equations based on the above are 'legal:'

```
<<*If(any)*!TypeDeed!="1,4",!Taxable!="True",!Taxable!="False">>
```

OR

```
<<*If(any)*!TypeDeed!="1,4",
!Taxable!="True",
!Taxable!="False">>
```

OR

```
<<*If(any)*!TypeDeed!="1,4",
!Taxable!="True",
!Taxable!="False">>
```

OR

```
<<*If(any)*!TypeDeed!="1,4",
!Taxable!="True",
!Taxable!="False"
```

```
>>
```

16.8 <<*Set*>> Command

Sometimes you need to **set** a value. To do so, use the <<*Set*>> command.

When <<*Set*>>-ing values, you are assigning a value associated with a !groupname! without using other Pathagoras equations. (<<*Options*, *Optional*, *Repeat*, *AskOptions*, *AskOptional* or *AskRepeat* commands all 'set' values, but there may be circumstances where you want to <<*Set*>> a value directly.)

The formula is "<<*Set*!groupname!=" followed by a value. The value can either be:

- "True", or "Yes" (for a 'true' type value)
- "False, "No" (for a 'false' type value)
- A number > 0 when setting a 'repeat value' for the !groupname!
- A positional value when you are setting a selection for a [multiple choice](#)²⁶⁹ item. (See special requirements in below examples.)
- A single word value.

NOTE: The **quotes** around the value to which the group name is being <<*Set*>> are **mandatory**.

Consider this:

```
<<*AskOptional*!Married!Is our client married?*>>.
```

When answered, the value assigned to !Married! is either "True" or "False" (or "Yes" or No).

You can directly set this value in a document where, avoiding the 'Is our client married?' question altogether (if the situation so calls) in the following manner:


```
<<*Set*!Married!="True">>
<<*Set*!Married!="Yes">> This is the functional equivalent of the above line.
```

You can also use a value set by an <<*AskOptions*>> or <<*AskOptions(radio)*>> command to set another value. Here's the setup, based on two MultiChoice lists. The alias of the first MultiChoice list is *States*. It contains 50 elements (the 50 United States of America). The second alias is called *SecurityType*. It, too, contains 50 elements. The values of these elements are the type of security instrument ('mortgage' or 'deed of trust') associated with each of the 50 States. The goal is to automate the selection of 'Deed of Trust' or 'Mortgage', depending upon the state in which the transaction is occurring.

The AskOptions below (first line) sets the value of the groupname !st! to the value the user selects from the list of the 50 States when presented in the Interview. Once the State is selected, Pathagoras can then the value of the groupname called !Security! to the parallel position as the selected state. (So, if Indiana, the 14th position in the list of *States*, is selected, the value of !st! would be '14'. Then Pathagoras processes the !security! groupname by looking at the !st! position in the *SecType* alias list. "Mortgage" occupies that 14th position, and therefore Mortgage is assigned to !security!

```
<<*AskOptions(radio)*!st!*States**>>
<<*Set*!Security! = !st!*SecurityType**>>
```

Multiple Choices:

You can set the value of a multiple choice value by stating the positional value(s) you are selecting followed immediately by the total number of possible choices in parentheses). Examples

Options (single choice):

```
<<*Set*!Spouse!="2(2)">> (The first number reflects the selected position of the choice, and
the number in parentheses represents the total number of choices
```

Options (multiple choices):

```
<<*Set*!Colors!="1,3,5(9)">> (where the first numbers, separated by commas, reflect the
selected positions of the choices, and the number in parentheses represents the total number
of choices.
```

Numeric/Repeats:

```
<<*Set*!NumCdn!="5(#)">> (where the first number reflects the number or repeat value, and
the pound sign/hash mark tells Pathagoras that an actual number, as opposed to position, is
intended.
```

Optional (True or False)

```
<<*Set*!Married!="True">>
```

Set Math:

[See next page](#) ²⁷⁰

Notes:

- The <<*Set*>> command can be a 'stand alone' command, and it can be used anywhere in a document (in addition to within the AskTable).
- 'Set by <<*If* . . >>. <<*If* . . >> equations (samples below and elsewhere) allow you to set groupname values directly. E.g.:

```
<<*If*!NumCdn!+!NumPets!>5,!CrazyHouse!="True",!CrazyHouse!="False">>
```

```
<<*If*!Fund Strategy!="Hybrid",!invest strategy!="1(2)",>>
<<*If*!Fund Strategy!~"Multi-Manager",!invest strategy!="2(2)",",!invest strategy!="1(2)",>>
```

- Beginning with Pathagoras 2020, you can combine an unlimited number of !groupnames! in the same set equation so long as they are being set to the same value. Just list them within a single set of exclamation marks, separated by commas. So, instead of this setup:

```
<<*Set*!AAA!="True">>
<<*Set*!BBB!="False">>
<<*Set*!CCC!="False">>
<<*Set*!DDD!="True">>
<<*Set*!EEE!="True">>
```

you can use:

```
<<*Set*!AAA,DDD,EEE!="True">>
<<*Set*!BBB,CCC!="False">>
```

(We think this will be much more readable and intuitive, and should result in fewer potential typos. Simple commas separate the individual groups.)

Caveat: While 'Set by <<*If*' (discussed above) works implicitly for setting a value for a single group, you cannot use it to set values for multiple groups as discussed in this bullet. You must include the <<*Set* command. E.g.,

```
<<*If*!Children!="False",<<*Set*!minors,guardian,UGMA!="False">>,>>
```

16.8.1 Set: Math

A potentially useful application for the <<*Set*>> commands is to compare or add (or subtract) the values of one or more existing !groupname! values.

E.g.,

```
<<*AskRepeat*!NumCats!*How many cats?>>
<<*AskRepeat*!NumDogs!*How many dogs?>>
<<*Set*!NumCritters!=!NumDogs!+!NumCats!>>
```

The resulting value of !NumCritters! can be used anywhere in the document. E.g.,

We have <<!numcats!>> cats and <<!numdogs!>> dogs. <<!NumCritters!>> pets may seem like a lot, but it's just right for me.

16.8.2 Set: Equal

You can <<*Set the value of one !groupname! to equal the selected positional value of another groupname.

```
<<*AskOptions*!petitioner!Plaintiff/Petitioner/Appellant/Complainant*>>
<<*Set*!respondent!=!petitioner!>>
```

The same value selected for !petitioner! group will be assigned to the !respondent! group. So if you selected 'Appellant' from the !Petitioner! group (position 3 of 4), the !Respondent! group will likewise reflect 3 (of 4) (irrespective of the fact that the textual values of each group may be quite different, even opposites).

For further explanation, let's assume is that there are a series of potential titles/values for group 'A' that have a parallel series of correspondent titles/values for group 'B'.

E.g.,

Let's assume the above entries in the AskTables, and the following in the body of the document:

```
<<*Options*!petitioner!*Plaintiff/Petitioner/Appellant/Complainant>>
<<*Options*!respondent!*Defendant/Respondent/Appellee/Respondent>>
```

The selection of Appellant in the Interview results in 'Appellant' and 'Appellee' remaining in the document body. (Actually, choosing Appellant for the !petitioner! group results in the assignment of "3 of 4" for the !petitioner! and the !respondent! group, but the ultimate result is 'Appellant' and 'Appellee'.)

Similarly,

```
<<*AskOptions*!owner!Grantor/Seller/Lessor*>>
<<*Set*!buyer!=owner!>>
```

In document body:

```
<<*Options*!buyer!*Grantee/Buyer/Lessee>> hereby agrees to acquire and
<<*Options*!owner!*Grantor/Seller/Lessor>> hereby agrees to convey . . .
```

Now, as stated above, and elsewhere, that the value of a group is simply positional. It is possible simply to assign '!petitioner!' (or '!buyer!') as the name of the 'second' group (instead of creating a new group) and the same result would obtain.

```
!petitioner!Plaintiff/Petitioner/Appellant/Complainant
```

```
!petitioner!Defendant/Respondent/Appellee/Respondent
```

If 'Appellant' (third position) is selected for the first instance of !petitioner!, the same position in any following same-named group will be automatically selected. The result is identical.

But that doesn't mean the 'Set: Equal' function has no benefit. While which !groupname! is used does not matter to Pathagoras (given that he has no innate intelligent), it probably does matter to you and your staff as you create and review (and teach) the document's setup. Reusing !groupnames! for opposites, while perfectly 'legal,' nevertheless can be quite confusing. Titling/grouping with more accurate and descriptive names will enhance understanding and teach-ability.

16.9 <<*Get*>> Command

You can get the value of a particular named cell (or several named cells) in an Excel spreadsheet, and assign its value to a !groupname!(s) using the following command:

E.g., <<*Get*!DecedentName!jones probate.xlsx>>

Anatomy: <<*Get*!groupname!spreadsheet.xlsx>> (colors are for emphasis and reference only)

- <<*Get* = the command

- **!groupname!** = either (1) the named cell in the spreadsheet or (2) the column title in a two row spreadsheet in the target spreadsheet. The named cell can be anywhere in the spreadsheet. The column title must be in row 1, and the value that Pathagoras returns must be in the cell immediately beneath that title.

Groupname represents the groupname constant to which you wish to assign a value used in a document. Its value within the Excel spreadsheet can be:

(1) an 'X(Y)' value representing selected values X (or X1, X2, etc.) of Y options. The X part represents the 'position' of a corresponding `<<*Options*!groupname!* . . .>>` block in the document. The Y part represents the total number of choices presented in the document's `<<*Options*!groupname!* . . .>>` block.

A typical value might look like this: '2(4)' (no quotes). '2' is the selected choice and '4' is the number of possible choices in the document.

Multiple selections are possible. Just separate with commas. 1,4,7(9). The selected choices 1, 4 and 7 of 9 possible choices.

(If you just include the position(s), but not the number of possible choices, Pathagoras can still process the item. It may take a split second longer.)

(2) a 'boolean' ('True' or 'False') value representing a 'keep' or 'delete' value for a corresponding `<<*Optional*!groupname! . . .>>` command. The value must be 'True' or 'Yes' or 'False' or 'No.'

(3) a numeric value representing the number of 'repeats' to be applied to a corresponding `<<*Repeat*!groupname!* . . .>>` command.

(4) a `<<document call>>`. The spreadsheet term must include the boundary markers '<<' and '>>' to denote the call..

- **spreadsheet.xlsx** = the target spreadsheet
 - If the spreadsheet is in the folder of the source document OR in the Excel folder set in the 'Settings ! All Settings ! Excel' screenage, you need not 'qualify' -- i.e., add drive and folder information -- the spreadsheet. Otherwise, you must qualify the spreadsheet with full drive and path information.
 - Actually, referencing the target spreadsheet is optional. If no spreadsheet is provided e.g., `<<*Get*!DecedentName!>>`, Pathagoras will look in the 'default' spreadsheet that is set in the 'Settings | All settings | Excel' screenage.
 - If "*" is stated as the spreadsheet name, e.g., `<<*Get*!DecedentName!*.*>>`, Pathagoras will present a navigation screen to let you select a spreadsheet.

Usage:

Use it as the 'position' in an `<<*Options* . . .>>` block.

- `<<*Get*!discount!>>` resides in the interview section.
- The value of !discount! in the target spreadsheet is '2(3)'.
- This text is in the document body:

Your purchase entitles you to a <<*Options*!discount!*ten/twenty/thirty>> percent discount."

- the resulting text (after processing) will read:

Your purchase entitles you to a twenty percent discount.

- Note: The positional value can be used with the '(cumulative)' argument as well.

Use it as the true/false value in an <<*Optional* . . . >>

- <<*Get*!discount!>> resides in the interview section.
- The value of !discount! is 'True'.
- The text in the document body is:

<<*Optional*!discount!Your purchase entitles you to a 10% discount.>>

- the optional text will remain in the document. If the value is 'False' (or 'No'), the clause will be deleted.

Use it as the value for a <<*Repeat* . . . >>block

- <<*Get*!Shareholders!>> resides in the interview section.
- The value of !shareholder! is 4.
- The text in the document body is:

<<*Repeat(and,)*!Shareholders!*[Shareholder Name]>>

- The resulting text will be:

[Shareholder Name@ 1], [Shareholder Name@ 1], [Shareholder Name@ 1] and
[Shareholder Name@ 1]

Return a document:

- <<*!groupname!>> will place the value of the groupname in the document.
- If the groupname is 'color' and its value in the spreadsheet is 'orange.docx', the word 'orange.docx' will replace the groupname, and the surrounding << and>> tell Pathagoras to call the target document in place of the <<document call>>.
- If the groupname is 'sigblock'" and its value in the spreadsheet is, let's say '<<sigblockJPQ>>', the text '<<sigblockJPQ>>' will temporarily replace the call. However, it being a document call, Pathagoras will immediately search for the document 'sigblockJPQ' and insert it into the document at this location.

Use it in an AskTable

- Compare the 'gotten' value with another value using the <<*If*>> [command](#)^[260].
- Analyze it using [Pathagoras 'range' function](#)^[257]. (In this case, Pathagoras will process the result and return an appropriate block of text associated with the 'gotten' value. Perfect for psychologists and others who need to provide information associated with a particular score on a test. Lots of other applications possible.)

=====

Getting multiple values: to save a bit of time, you can request multiple values from the same spreadsheet. Just list the target names as part of a 'long' groupname, separating the individual names with commas. E.g.

```
<<*Get*!DecedentName,DecedentCity,PRName!jones probate.xlsx>>
```

16.10 <<*Break*>> Command

The processing of commands in an AskTable is top to bottom. Pathagoras will try to display as many Asks as possible, but no more than 10 at a time (this is more a function of screen size than anything else).

Two exceptions:

1. If the AskTable contains an '<<*If*...>>' logic point for which the comparison '!groupname!' has no value, Pathagoras will stop while you provide that value.
2. The <<*Break*>>command. Insert "<<*Break*>>" on its own line anywhere in the AskTable and Pathagoras will 'cut-off' adding additional questions until the one's before the 'break' are answered. <<*Break*>> allow you better presentation and control over the questions that are presented.

16.11 <<*Process*>> Command

Many documents prepared for clients are practically mirror images of each other. A classic 'I Love You Will' set (where all property is given to the surviving spouse, for the second to die, all property evenly among the children) can, with a proper setup, be produced in a single run. One example of a 'reversal' is provided at this link. (There, the reversal is accomplished by manually 'reversing' the order of the variables in the second document to be created.

The <<*Process*>> command, in conjunction with the <<*Set*>> command. The positional values of the first will are initially set, and the document is processed. When the <<*Process*>> command is hit, every thing below the command is ignored until the top portion is fully processed. The the portion below is processed, with any <<*Set*>> command honored for that bottom portion. (Since there is nothing left to process at the 'top, the only the 'bottom' will be controlled by the new settings.'

```
<<*Set!Testator!="1(2)">>
```

Last Will and Testament of

!Testator![HUSBAND NAME]/OR[WIFE NAME]

I, {Testator![Husband Name]/OR[Wife Name]}, being of sound mind, make this document my Last Will and Testament.

1. I give all of my property and estate to my {!Testator!wife/ORhusband},
{!Testator![Wife Name]/OR[Husband Name]}

2. I appoint my {!Testator!wife/ORhusband}, {!Testator![Wife
Name]/OR[Husband Name]} to be my Personal Representative.

{!Testator![Husband Name]/OR[Wife Name]}

<<*Process*>>

<<*Set*!Testator!="2(2)">>

Last Will and Testament of

{!Testator![HUSBAND NAME]/OR[WIFE NAME]}

I, {!Testator![Husband Name]/OR[Wife Name]}, being of sound mind, make this document my Last Will and Testament.

1. I give all of my property and estate to my {!Testator!wife/ORhusband},
{!Testator![Wife Name]/OR[Husband Name]}

2. I appoint my {!Testator!wife/ORhusband}, {!Testator![Wife
Name]/OR[Husband Name]} to be my Personal Representative.

{!Testator![Husband Name]/OR[Wife Name]}

The main difference between this reversal and the one illustrated at the link above is that the identical language can be used for both documents. Indeed, the text itself can be called in via a document call, so that one a single document need be edited. Here is an example

```
<<*Set*!Testator!="1(2)">>
<<I Love You Will Base>>
<<*Process*>>
<<*Set*!Testator!="2(2)">>
<<I Love You Will Base>>
```

The above can be copied and pasted into a Word document and saved as "I Love You Will.docx" The 'Will' text (just one copy, minus the commands) can be saved in the same folder as "I Love You Will Base.docx". (Names just suggestions. Call them anything you want. When processes, Pathagoras will run the first 'base' with the group !Testator! set to the first position. Once it has fully processed, the group !Testator! is reset to select the second position. A second copy of 'base' is inserted and processed as requested. Note: A section break is inserted by the command at the '<<*Process*>>' line so that headers and footers that may be in the 'base' document are kept separate.

Note: This was designed to enable reversals of simple documents such as indicated above. Long and complex documents, with multiple sections and orientation changes will not 'reverse' well.

16.12 <<*Remarks*. . .>>

As you take advantage of the sophisticated logic possibilities of Pathagoras, you will likely want to record comments and explanations to remind you or others as to the objective of a certain section of the interview (AskTable). Do so simply by inserting a <<*Remarks*. . .>> command. The command can be of any length.

E.g., <<*Set*!married,children,trust,QuitClaim,disinheritance!="False">>

<<*Remarks*Above line sets certain values to 'False'; specific values will later be set to 'True' as appropriate.>>

<<*If*!TypeDeed!="3", !QuitClaim!="True">>

16.13 Logic Lesson

 **The opposite of 'True' is not 'False'.** The opposite of True is simply Not True. Let discuss this in the context of the below example:

```
<<*If*!Children!="True", <<*AskOptional*!Minors!Are there minor
children*>>, >>
```

The above <<*If* . . .>> statement is incomplete if you need to set !Minors! to "No" (or "False") when of !Children! is 'No' and you also intend to use a statement such as <<*If*!Minors!="False", (action)>>. The 'False' value of !Minors! cannot be assumed. It must be set by an equation, or you must affirmatively set it elsewhere if you ever need to use it.

The below 'If' statement handles the issue:

```
<<*AskOptional*!Children!Are there any children of the marriage?*>>
<<*If*!Children!="True", <<*AskOptional*!Minors!Are there minor
children*>>, !minors!="False">>
```

The above 'reads' (in 'English'):

Ask if there are children of the marriage.

If the value of !Children! is 'True', then ask 'Are there are any minor children?'

But if there are no children (i.e., the value of Children! is 'False') then simply set 'minors' to 'False'.



If you have many !groups!, you can preset some (or all) of them to a determined "True" or "False" value to avoid having to set an 'affirmative opposite' in each individual equation. [See this link](#) ²⁶⁸.

16.14 Setting Processing Order

The normal processing of <<*Options/Optional/If*>> commands is from top to bottom. On receipt of the 'Process' command (automatically triggered when item is called during Document Assembly or from a DropDown List, or from the <Alt-P> triggered), Pathagoras scans the document starting from character 1.

<<*Ask*. . .>> commands, if present, are always processed first (even if they are not at the top of the document). They are processed top to bottom.

Once all elements in the Interview (AskTable), and their respective !group! members, have been processed, the scanning for the remaining 'process' controls begins, starting from character 1, and running top-to-bottom.

When Pathagoras finds an <<*Options*>> or <<*Optional*>> block, it performs the requested action.

Do note these rules and exceptions to the 'top-to-bottom' rule for non-interview process blocks:


- a. <<*Repeat* commands are processed next to last (but still top to bottom). That way, if a <<*Repeat* is within a Optional or Options block that is going to 'disappear', no time is wasted processing it. (If you want a <<*Repeat*. . .>> to be processed 'first' or 'earlier', simply include it as an <<*AskRepeat*. . .>> in the interview.)
- b. {Simple Optionals} that are not part of a group are processed dead last (but still top to bottom).
- c. Any block that is member of a just processed !group! is processed (top to bottom) immediately after the first member of the group is processed.

16.15 Creation of Interviews

16.15.1 Automatic Creation of <<*Ask*>> prompts

Automatic creation of <<*Ask*>> commands:

At your request, Pathagoras can scan your source document for any <<*Options*>>, <<*Optional*>> and <<*Repeat*>> blocks that are present. It can pull the essential elements from each such block and then create a basic <<*Ask*>> command for each such block, placing it the top of your document. You can further edit and refine the prompt so that it best meets your needs.

 **Note:** Only <<*Options*>>, <<*Optional*>> and <<*Repeat*>> blocks that contain a !group! reference will be processed. So, before running the routine discussed below, you may wish to add a !group! reference to your <<*Options*>> and <<*Optional*>> text blocks to take advantage of this feature. Even if there is only a single member of the group, you may find that the benefits of the <<*Ask*>> command are worth the effort to do so.


To create the <<*Ask*>> commands in this automated fashion:

1. Make sure that the document to which you want to add <<*Ask*>> commands is on screen.
2. Click the Pathagoras Features | Wizaards and Assistants | Create Interview (AskTable) element.

When clicked, Pathagoras will scan the document for any <<*Options*>> and <<*Optional*>> text blocks that contain a !groupname! reference. Pathagoras parses out

the administrative content of each one and placed it into an <<*AskOptions*>> or <<*AskOptional*>> block, as appropriate, at the top of the document. An additional 'instruction' block is added containing usage information.

3. Review the prompts that were created. Modify the questions as appropriate to make the appropriate response as unambiguous as possible.
4. Save the document.

 **Clause Sets:** If you assemble documents using Clause Sets, the actual <<*Options*>>, <<*Optional*>> and <<*Repeat*>> blocks may not be present in the 'raw' document, and therefore are not going to be picked up by the tool discussed above. Here are the steps to follow in that instance:

- create a 'complete' document which contains all possible options and optional text blocks.

It is not important at this stage that the resulting document contains more text than would ever be used. The idea here is to have a document that contains all of your <<*Options*>>, <<*Optional*>> and <<*Repeat*>> elements.

Before you create this 'huge' document, you should turn 'off' processing so that the <<*Options/Optional/Repeat*>> blocks are not automatically processed.)

- Run the automatic <<*Ask*>> command creator described above.
- Edit the <<*Ask*>> commands to your satisfaction.
- Copy and paste the entire block of <<*Ask*>> commands into the first clause that you use when assembling via Clause Sets. (Typically, the 'first clause' is the same throughout all documents in the topic, but not necessarily.

Limitations:

1. Only <<*Options/Optional*>> blocks with !groupnames! will be elevated to the <<*AskTable*>>.
2. The <<*Options/Optional*>> must either contain 'prompt' statements/questions or, if no such statements, actual text options whose combined length is less than 150 characters. (If longer than 150, an <<*Ask*>> block will still be created, but you will be prompted to create manually the actual prompts that will be asked of the end user.)

16.15.2 Logic Assistant

Pathagoras provides a Logic Assistant to help you to construct the equations necessary to perform logical operations within your document.

Activate the Logic Assistant from the Pathagoras Features | Wizards and Assistants. Click the Logic Assistant element. The following screen will appear:

Scan the screen for a second. Note the logic: "If" this value exists, then "perform or set" this value. Here are the steps to complete the assistant.

1. Click one of the radio buttons at the top of the screen. These options reflect the various types of Ask elements that can exist in a document (Options/Optional/Repeats). In our example, we choose Optional because "Are there children?" is presented as an "AskOptional" in the Ask Table.

2. As soon as you click on the Ask element (1), the left and right panels are populated with choices that you can make. At the left, select the element against which you want to apply your logic equation. Here, we want to apply logic against the element "Children." That is, "IF there are Children . . . do something."
3. Continuing with the "IF" part of the equation, select the appropriate value of the element that is being compared.
4. While this '4' is not a step, we point out that Pathagoras is beginning to construct the logic formula for you. Use this as a reference point if you become confused.

5. Look at the screen below. In the middle panel, make a selection as to which logic step is to be first presented. The choices are simple: to display the 'Ask' associated with the target value, or to set a specific value (True or False, or one of the choices) for the target. The check box reflect which appears first.

6. From the right panel, select the element that will be acted upon (here we have selected 'Minors') based on the value of the selection at the left. Once we select the interrelated elements from the left and right panels,
7. Choose the value of the element from the left panel that is to be measured. In the case of Optional blocks, the only values are "True" or "False". (For "Options" blocks, the values can be any combination of the actual choices.) All two possible actions are shown.

IMPORTANT: Check "where you are" frequently. Always keep an eye to the formula bar (8) that changes every time you make a new selection. None of the instructions above will make sense in the vacuum of this page, but when you see the formula change as you press buttons, we assure you that the Logic Assistant will communicate what it needs. Plus, read the elements of the form. Note the left to right layout and 'descending' displays that make it easier to visualize the formula you are creating.

8. Check the formula bar to make sure that the "If (THIS), then (THIS), else (THIS)" is what you desire.
9. Make changes as needed. When done, click the Transfer button.

Document Logic Assistant

Select the type of prompt that will provide the 'test' value when answered:

- ☐ Options (multiple choice)
- ☒ Optional (keep or discard)
- ☐ Repeats

This form is designed to assist you in creating logical IF/THEN type equations with your <<*Options/Optional/Repeats*>> blocks.

Click 'type' of block (options, optional or repeats) from the choices at the

If this Optional block term is selected:

(Choose from below !Groups!)

- Children
- Minors
- Withdrawal Rights
- Special Use

is:

- ☒ 'True' or 'Yes'
- ☐ 'False' or 'No'

Perform/Set (check = Show First)

- ☒ ... the Ask associated with selection at the right, and
- ☐ ... the value(s) selected below, right

... on this value.

(Choose from below !Groups!)

- Married
- Sex of Client
- Children
- Minors
- Withdrawal Rights

is:

- ☐ 'True' or 'Yes'
- ☒ 'False' or 'No'

Sample:

```
<<*If*!Children!= "Yes", <<*AskOptional*!Minors!Minor children?*>>, !Minors!= "No">>
```

```
<<*If*!Children!= "True", <<*AskOptional*!Minors!Are there minor children?*>>, !Minors!= "False">>
```

Transfer to Asks Table

When you click the Transfer button, Pathagoras hunts down the element in the Ask table that is associated with the target (in this case "Minors") and replaces it with the formula. So the Ask table shown at the top of this page will now look like this (check out the 4th row):

<<*AskOptions(radio)*!Married!Married/Not Married*>>
<<*AskOptions(radio)*!Sex of Client!Male Client/Female Client*>>
<<*AskOptional*!Children!Are there children?*>>
<<*If*!Children!= "True", <<*AskOptional*!Minors!Are there minor children?*>>, !Minors!= "False">>
<<*AskOptional*!Withdrawal Rights!Exercise Annual Withdrawal Rights.*>>
<<*AskOptional*!Special Use!Special Use Valuations*>>

Here is the logic equation found in the 4th row, in 'English':

"If there are children, then ask the question 'Are there minor children?'
Otherwise, set the value of the !Minors! group to 'False' ."

16.16 Saving Interview Answers

Starting with version 2019, you can tell Pathagoras to save your selections. After you have made choices on the 'Wizard' screen, but before clicking the Next button, check the Save box near the Next button.

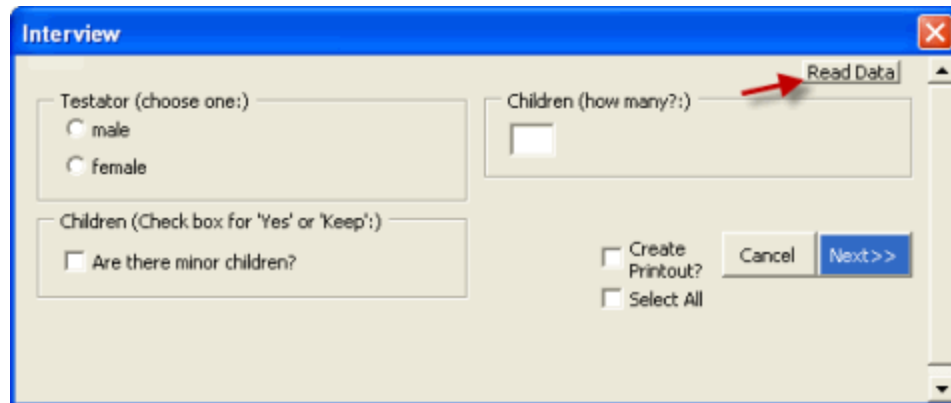
Provide a record name when prompted. (This is similar to the Instant Database used to record personal values that will replace variables. Pathagoras will record your choices for later use and then proceed to implement your selections.)

If a second or subsequent screen of the Wizard appears, Pathagoras will automatically record the choices in the same file named in the initial screen. Pathagoras will make all changes to your document (making Options choices, deleting optional text you elected not to keep, repeating text blocks and incrementing variables as appropriate, etc.) as reflected in your selections.

The flip-side to 'Saving' is 'Recalling,' and that is discussed on the [next page](#) ²⁸².

16.17 Recalling Interview Answers

When you have assemble a document which contains an <<*Ask* . . . >> table, the Wizards screen will display. If there are <<*Ask* . . . records that you have previously saved (see previous page) a button will appear in the extreme upper right corner labeled "Read Data". If you wish to recall a record instead of manually making each selection, click that button.



From the screen that next displays, select the desired record. Then, press the 'Go' button on the overlay screen to continue.



Pathagoras will automatically select the radio buttons and checkboxes and fill in the 'Repeat' values in accord with the saved data. (The example given, with only 3 choices on screen, hardly exemplifies the value of this features. But when there are multiple screens and dozens of choices, this can be a tremendous time saver.)

Make any changes as may be appropriate. When you click the 'Next' button, Pathagoras will ask if you want to update the existing record, create a new record or keep the original settings. This latter (keep original settings) feature allows you to record a 'standard' or 'universal' set of answers perhaps named 'Standard' --or anything you wish-- so that Pathagoras will automatically select the 'typical' values for you. You then can modify the selections as appropriate and save the new, more personalized, record for the specific client or customer. (This feature is similar to the 'mask' function of Instant Database records.)

16.18 External source of Interview Answers

Values for answers are typically derived from completion of Interviews discussed in this section of the Manual. But those Interviews are not the only potential source of values.

If you take a look at the answer file that Pathagoras creates, you can see that it's just a simple csv file. This kind of file can be created from any source, including Excel. So if a user skilled in Excel wanted to create a spreadsheet that could generate a csv file in the style of

```
"!groupname1!: /value/^upperlimit"
```

```
"!groupname2!: /value/^upperlimit"
```

```
"!groupname3!: /value/^upperlimit"
```

etc.

'Value' represents the positional value of the choice made by the user. If the choices are Red, Blue and/or Green, and the user chose 'Blue', the 'value' would be '2'. If the user chose Red and Green, the value would be 1*3.

The 'upper limit' refers to the number of 'Options' choices that are presented in the document for possible selection. If the user could choose among Red, Blue or Green, the upper limit would be '3'. For the 50 United States, the upper limit would be '50'.

If the answer is binary (i.e., an <<*Optional* choice, Pathagoras uses a slightly different nomenclature. The value would be 0 (for false or delete) and 1 (for true or keep). The upper limit off an Optional choice is always '1'.

If the value is a number (*Repeat*), the value is presented between hashtags, with no upper limit.

e.g.

!number of Children!: #3#

16.19 Print Interview

To generate a printout of the various <<*Options*>> and <<*Optional*>> "questions", look for and check the "Create Printout" box when the Interview is first displayed.

Note: The printout is actually an editable Word document. All questions, including nested ones, will be shown, in the order in which they appear in the source document. The printout cannot itself be 'read' by Pathagoras as a template for another document, or otherwise processed, but it may nevertheless serve as a good record for printing out and inserting into a client's file.

16.20 'Case' Logic

'Case' logic is a more complex extension of the If/Then/Else logic discussed earlier. The term 'Case' derives from the sentence "In CASE a certain value(s) is (are) THIS, then do this"

Case logic is especially useful for comparing multiple values, multiple combinations of values, and for determining values in a cascading fashion.

- If/Then/Else statements contain basically two result sections: the True part and the False part. It is 'linear' in structure, reading from left to right, "If this value is X, then do this, else do this.
- 'Case' logic allows many result sections. It is 'stacked' in structure, and reads from top to bottom. Example (read the commas as "then"):

Case

Combination 1, A action
Combination 2, B action
Combination 3, C action
etc.

- 'Combination' above refers to the choices made in a preceding <<*AskOptions*>> display. The combinations refer to the positions of the selections made. (Caveats: If the display was from an <<*AskOptions(radio)*...>> call, only one position, not combinations of positions, will be 'analyzed'. Ditto If the display was from an <<*AskOptional*...>> call.)
- When multiple values are involved, multiple considerations must be handled. Here's how we do it.
 - The answer combination is always converted to a positional number. So, if five items are in the list of choices, and the user selects the second and fifth item, Pathagoras will 'see' the answer numerically as 2 and 5.
 - If all the choices must be present in order to trigger the case event (but others can be as well), just type the qualifying choices, separating multiple choices with commas.
 - If only the exact choices listed must be present to trigger the case event (and none others), enclose the choices within curly braces. (The choices themselves must still be separated by commas.)

➤ If any one of the listed choices will trigger the case event, separate the choices with '/OR'.

- When the first true value is reached, the line is processed and the routine ends.
- The Case list can be endless.

Below is an example that demonstrates how the initial selection of items (in this example, salad fixings) can lead to a cascading of subsequent selections including the salad name and the utensils needed to consume the salad.

You can copy and paste this example into a document and process it. Pathagoras will ask you the first question dealing with the salad's "ingredients". If the user selects a predesignated combination of ingredients, Pathagoras will automatically return the common name of the salad and the utensil commonly used to consume it. Otherwise, the AskOptions questions (the final entry in the Case list) are presented.

<pre><<*AskOptions*!Ingredients!Celery sticks/Julienne Carrot Strips/Raisins/Apples/Bananas/Cream Cheese/Peanut Butter/Crackers*>></pre>
<pre><<*Case* !Ingredients!= "1,6",!Salad!="1" !Ingredients!= "2,3",!Salad!="2" !Ingredients!= "1,2",!Salad!="3" !Ingredients!= "3,4,5",!Salad!="4" <<*AskOptions*!Salad!Stuffed Celery/Carrot and Raisin Salad/Sliced Veggies/Fruit Salad/Unappetizing mishmash*>> >></pre>
<pre><<*Case* !Salad!= "1",!Utensils!="4" !Salad!= "2",!Utensils!="2" !Salad!= "3",!Utensils!="4" !Salad!= "4",!Utensils!="4" >></pre>
<p>(The top part of this document is the 'Ask Table'. The below is the 'actual' document. Below are six 'primary ingredients' for making a side dish of some sorts.</p> <p style="text-align: center;">INGREDIENTS:</p>
<pre><<*Options*!Ingredients!*Celery sticks, / Julienne Carrot Strips, /Raisins, /Apples, /Bananas, /Cream Cheese/Peanut Butter/Crackers>></pre>
<p>Objective: We want Pathagoras to select the specific salad (below) if a known combination is selected from among the ingredients above. So, if a pre-determined combination of ingredients is selected, the proper return will be made. Otherwise, Pathagoras will present a question allowing you to manually select the answer. Then, once the kind of appetizer is determined,</p>

Pathagoras will determine (or provide a list for you to select) the utensil to be used to consume the appetizer.

The type of SALAD made from the above ingredients is:

<<*Options*!Salad!*Stuffed Celery/Carrot and Raisin Salad/Sliced Veggies/Fruit Salad/Unappetizing mismash>>

For the above side dish, we will need the following UTENSILS:

<<*Options*!Utensils!*Knife and Fingers/Fork/Spoon/Fingers>>

A reminder: the answers that Pathagoras is analyzing in the Case list are positional. "1,6" refer to the first and sixth positions in the ingredients list (celery and cream cheese). For

<<*AskOptional*>> commands, a 'yes' answer is assigned a '1' value, a 'no' answer a '0' value.

Creating a Case Logic List.

1. Display the Logic Wizard (previous section).
2. Make all selections as before, but check the box in the middle of the screen that reads "Create Case Logic Block"
3. After you have created your first Case Logic item, click the Transfer button. The initial formula will be copied into the clipboard and the screen will close. Navigate to the position in the Ask Table where you want the Case Block to appear and click Paste (Ctrl-V).
4. Note: You can only create the *structure* for the case block, which will consist of the Case title, one (and only one) Case entry PLUS the Ask associated with the primary element selected at the right (this Ask must be the last entry, as it is the 'default item' if no prior Cases are true when document processing occurs).
5. You can (and should) add as many additional 'tests' as you wish, following the pattern shown in the transferred Case Logic Block.
6. If all selections must be met, enclose the selections within {curly braces}. Otherwise, the default 'one or more of the choices must match' will apply.'

Here is another, perhaps more 'practical' example. The clauses <<will100>>, etc are clauses that shipped with the program. Therefore, you can copy and paste this example into a document and actually process it.

Objective:

To automatically select specific clauses based on choices made by the user from an Options or AskOptions block.

In the below example, a variety of family structure elements are presented. These choices are presented to the user once document processing starts. Once the user makes selection, the choices are evaluated by subsequent 'Case' blocks. (That is if any are provided. Case is optional. Pathagoras can evaluate the

AskOptions answer just fine for other Options blocks further in the document Case is used for its ability to set a second or third value that will be used elsewhere.. This is called 'conditional branching'.)

If a pre-determined combination of choices has been made (all selections based simple on position—1,2,3 etc in the Options list), the designated return of a clause will be made. If a pre-determined combination was not selected, the 'default' value provided in the 'Case' block will be chosen. (The default value is always the last choice provided. That is because once a line in the case block evaluates 'true', the Case block is exited. If the last line is reached, it is the only one left. In the example below, the last choice is the same as if "Married, w/Children" had been selected for !Family Structure! group and "No Minor Children" for the !Guardian!.group.)

More explanations provided below.

```

<<*AskOptions(radio)*!Sex of Client!Male/Female*>>
<<*AskOptions(case)*!Family Structure!(1)Married/(2)No Children/(3)Child/(4)Children/(5)
Minor children*>>
<<*Case*
!Family Structure!= "1,2",!Clauses!="1"
!Family Structure!= "1,3",!Clauses!="2"
!Family Structure!= "1,4",!Clauses!="3"
!Family Structure!= "2",!Clauses!="4"
!Family Structure!= "3",!Clauses!="5"
!Family Structure!= "4",!Clauses!="6"
!Clauses!= "3"
>>
<<*Case*
!Family Structure!= "5",!Guardian!="True"
!Guardian!="False"
>>

```

The above text is the 'administrative section' of the document. It comprises the 'Ask Table' and the Case processing section. The Ask Table asks the questions when the document is called up via the Document Assembly screen or a DropDown List (or if you 'fake' processing via Alt-P. The below is the 'actual' document.

If you want to suppress the "and" "or" "space" "none" connector options that typically appear below any AskOptions control, just append '(case)' after "AskOptions".

Here we add another element to the example, and that is the ability of Pathagoras to call in clauses set out in double angle brackets. All clauses to be inserted in the example below are actual clauses. They are part of the DemoData clauses (Wills clauses) that shipped as part of the Pathagoras installation package. Of course, you can substitute real text in lieu of 'wil100'. etc.

Here, the 'Preamble' (wil100) will be inserted regardless of family structure (no 'if' or 'Options' or 'Optional' assigned to it.

```

<<wil100>>

```

Here, depending upon the selections made regarding family structure, one of the following clauses will be inserted. The 'Family Structure' choice made at the outset set a 'Clauses' value. That value is used as a 'position' in the below list of Clauses.

```
<<*Options*!Clauses!*<<will 10(m,c0)>>/<<will 10(m,c1)>>/<<will 10(m,c2+)>>/<<will 10(s,c0)>>/<<will 10(s,c1)>>/<<will 10(s,c2+)>>>>
```

Now, throw in a few more standard clauses:

```
<<will 20>>
```

```
<<will 30>>
```

If 'minor child' was selected in the Family Structure ask, the Guardian clause will be inserted. Otherwise it will be omitted. (This is controlled by the 5th element of the family structure questions as analyzed by the second "Case" block above.)

```
<<*Optional*!Guardian!*I hereby name [Guardian] to be the guardian of my minor children.>>
```

Now, finish up with a few standard closing elements and the signature block based on the answer to the opening Ask prompt 'Sex of Client'.

```
<<will 80>>
```

```
<<will 90>>
```

```
<<*Options*!Sex of client!*<<wil300m>>/<<wil300f>>>>
```

16.21 Hard Values

A 'hard value' as used by Pathagoras is simply the actual text of the selection that was made in response to an <<*Options* . . .>> or <<*AskOptions* . . .>> prompt (as opposed to the positional value of the answer.) So if the prompt is <<*AskOptions(radio)*!Size!Small/Medium/Large/Custom*>>, and you choose medium, two 'answers' are recorded by Pathagoras: "2" for the positional value of the choice (which processes slightly faster) and "Medium" (the 'hard' value).

16.22 Requirements

The Interview form can be created dynamically. There is no design required.

However, the Interview is generated based on the existence of <<*Options*>> and <<*Optional*>> text blocks in the document which control the selection of text by the user. Therefore,

- You must have <<*Options*>> and <<*Optional*>> text blocks in your document. See [this section of the Manual](#)^[172] for information on how to construct (and simple it is to construct) these powerful text blocks.
- Each <<*Options*>> and <<*Optional*>> text block must contain a !GroupName!. It is this !GroupName! that is checked to determine if an <<*Options*>> and <<*Optional*>>

text block will be moved into the Interview form. (If there is no !GroupName!, the text block will be processed, but after the Interview is created and processed.)

- Regular <<*Options*>> and <<*Optional*>> construction rules apply. If you are not sure if your document complies, run the "Structure Checker." Pathagoras will fix things for you automatically. Use the [Options Block Creation Assistant](#)¹⁹⁴ to help you correctly form the text blocks.

16.23 Interview Examples

Enter topic text here.

16.23.1 Simple Interview Examples

Here are a few examples of <<*Ask*>> prompts. (The first line is the Ask. The other lines are the <<*Options*>> or <<*Optional*>> text blocks against which the !group! value is applied.) You can copy any (or all) of the examples to your editing screen and see how the elements work together.

<<*AskOptions*!Customer!male/female/more than one*>>

Memo to JRT:

Send this package to <<*Options*!Customer!him/her/them*>>.

<<*Options*!Customer!He is/She is/They are*>> going to sign it and send it on to

<<*Options*!Customer!his/her/their*>> attorney.

<<*AskOptional*!Children!Will there be young children attending?*>>

We are having a party. Date: [Date of Party]

Location: [Location of Party]

<<*Optional*!Children!*Your children are invited.>>

Please bring a covered dish and your favorite beverages.

<<*Optional*!Children!*Make sure that your children bring their favorite toys.>>

Here is an example that illustrates the point about a 'less than obvious answer' *Options* block. If the *Ask* prompt were not present, an end user who is unfamiliar with the periodic table likely would not be able to process the document with confidence:

<<*AskOptions*!Metal!Gold/Iron/Carbon/Mercury*>>

Here is a great experiment, boys and girls, that you can try at home:

Pour 10 ounces of finely ground <<*Options*!Metal!*Au/Fe/C/Hg*>> into a bowl.

Add a 16 ounces of H₂O and 3 jelly beans. Mix well. Heat in oven at 400 degrees for 5 minutes. Let cool 22 minutes.

Out will come a shiny new <<*Options*!Metal!*Ring/Car/Diamond/Thermometer*>>!

(My lawyer made me do this: Please for goodness sake, do *not* try this at home (or anywhere else)!! It won't work. Besides, 400 degrees is really hot!)

This example illustrates *cascading logic* in an Estate Planning setting. It captures

- whether there are children in the mix and if so,
 - ❖ how many children there are
 - ❖ If just 1, whether the child is a 'son' or 'daughter' ('To my child' in a Will sounds so 'stiff,' whereas 'to my children' always sounds 'good'), and
 - ❖ whether any child is still a minor (so that perhaps a Guardian can be appointed).

(Note: This is not the only approach, so don't hesitate to be creative and design an even more efficient cascade.):

```
<<*AskOptions(radio)*!HasChildren!Client has children/Client has no children*>>
<<*If*!HasChildren!="1",<<*AskRepeat*!NumChildren!Number of Children*>>,!NumChildren!="0(#)">>
<<*If*!NumChildren!="1",<<*AskOptions(radio)*!ChildGender!Son/Daughter*>>,>>
<<*If*!NumChildren!>"0",<<*AskOptions(radio)*!AllAdults!All children adults/Minor child(ren)
*>>,!AllAdults!="1(2)">>
<<*If*!NumChildren!>"0" and !AllAdults!="2",!Guardian!="True",!Guardian!="False">>
```

This example addresses what a manufacturing company might face in processing special orders. There are extra manufacturing charges and extra shipping charges associated with custom orders. Here the ~ (not equal) is illustrated.

```
<<*AskOptions(radio)*!Size!Small/Medium/Large/Custom*>>
<<*AskOptions(radio)*!Color!Red/Blue/Green/Custom*>>
<<*If*!Size!="Custom" AND !Color!="Custom",<<*AskOptions*!ExtraCharge!$50/$60/$70*>>,>>
<<*If*!Size!="Custom" AND !Color!~"Custom",<<*AskOptions*!ExtraCharge!$100/$150/$200*>>,>>
<<*If*!Size!="Custom" OR !Color!="Custom",<<*AskOptions(radio)*!Shipping!$20/$30/$40*>>,>>
```

16.23.2 A Whimsical Example

The previous sample has a useful application. This one does not, but perhaps by its very non-standard nature, it will make more sense to some:

--Copy Below--

The below gets or sets values via the AskTable.

Ask Block Text	Before processing	After processing
<<*AskRepeat* !NumCdn!*How many children?>>	Repeat block created in normal fashion, with 'Repeat' designator and groupname followed by the question. Let's say "2".	The AskRepeat line will disappear, but "2" has been assigned to !NumCdn!
<<*AskRepeat* !NumCats!*How many	Repeat block created in normal fashion, with 'Repeat'	The AskRepeat line will disappear, but "4" has been

<code>cats?>></code>	designator and groupname followed by the question. Let's say "4"	assigned to !NumCats!
<code><<*Set*!NumDogs!=3(#)*>></code>	The values in parentheses relate important information. If a '#', it says the value is a raw number. NumDogs is set to the raw value of 3 (dogs);	The Set line will disappear, but "3" is assigned to !NumDogs!
<code><<*If*!NumCdn! +!NumDogs!>"5",!CrazyHouse! !="1(2)",!CrazyHouse! !="2(2)">></code>	If a number, it conveys that the Group is an 'Options' (vs. Optional) block and the total number of choices. Here, !CrazyHouse! is an Options (not Optional) block and there are 2 choices provided in the document.) If the value set were "True" or "False" instead of a number, the block would be "Optional".	Pathagoras performs the math. Since !NumCdn! plus !NumDogs! is more than 5, !CrazyHouse! is set to "1" (or the first) of 2 choices.
<code><<*Set* !NumCitters!=!NumDogs! +!NumCats!>></code>	The above sets another value based on the input and/or set value.	Pathagoras performs the math and sets a value. Since !NumDogs! plus !NumCats! equals 6, so !NumCitters! is set to "6".

The below is body text. The results of the AskTable are carried into the body text.

The names of my child(ren) is/are <<*Repeat(and)*!NumCdn!*[Child]>>	This illustrates Pathagoras' 'repeat action' using the raw value of the AskRepeat for NumCdn.	The names of my child(ren) is/are [Child@1],[Child@1] and [Child@1]
The names of my dog(s) is/are <<*Repeat*!NumDogs!*[Dog]>>.	This illustrates the 'repeat action' using the raw value of the Set function for NumDogs.)	The names of my dog(s) is/are [Dog@1], [Dog@2], [Dog@3].
I have <<*Options*!NumCdn!*no children/one child/two or more children>>.	This illustrate the 'Options' action which transforms the Repeat value of NumCdn: '0'=first option;'1'=second option;'2+'= third option.)	I have two or more children.
<<*Options*!CrazyHouse!*This is a crazy house with <<!NumCdn!>> { !NumCdn!children/ORchild/ ORchildren} and <<!NumDogs!>> dogs inside./This is a sane house with {!NumCdn!no children/ORonly one	This illustrates (1) how Pathagoras can use a set variable (CrazyHouse); (2) insertion of raw Repeat and Set values via GroupNames; and (3) 'Simple Options' action which transforms the Repeat value of NumCdn: '0'=first option; '1'=second option; '2+'= third option.	This is a crazy house with two children and three dogs inside.

child/ OR only <<! NumCdn! >> children} and <<! NumDogs! >> dogs inside.>>		
This house has <<! NUMCRITTERS! >> critters inside!	This illustrates returning just the 'raw' value. The number is spelled out and the case and emphasis is preserved.	The house has SIX critters inside.

--Copy above--

16.23.3 Explict and Implicit !GroupName! settings

Pathagoras offers much more power and flexibility to choices made in response to <<*AskOptions* . . . >> blocks than at first may meet the eye.

When an <<*AskOptions* equation is processed, the selections (actually, their positions) are 'explicitly' assigned to the !groupname' attached to the equation. Consider the following example:

<<*AskOptions*!**Fruit!**Apples/Bananas/**Cherries!**Dates*>>

If the user selects Apples and Cherries, the groupname !Fruit! is **explicitly** set to "1,3".

Additionally the individual elements of the <<*AskOptions* . . . >> are implicitly assigned a True/False value that can be used for calculations and evaluations anywhere in the document as if it were assigned via individual <<*AskOptional* . . . >> requests for each element.

So, completing the example, !Fruit1 is set to "1,3" and !Apples! is set to "True", !Bananas! to "False", !Cherries! to "True" and !Dates! = "False". (The value of the implicitly created !groupnames! is the exact language of the choice, so long as the text of the choice was 39 or fewer characters.)

An example works best. You can copy and paste the entire block of text to an editing screen. To process it, press <Alt-P>.

--Copy Below--

The below (left column) is a standard 'AskOptions' prompt.

Ask Block Text	Before processing	After processing
<<*AskOptions*! Fruit! Apples/Bananas/ Cherries! Dates*>>	Ask Options block displays the !groupname! and menu choices you want to provide to end user.	This line will disappear once the Ask block has been answered. If Apples and Cherries are checked, the value of the !groupname! 'Fruit' is set to '1,3'

Document body standard behavior:

Body Text	Before processing	After processing
-----------	-------------------	------------------

<<*Options*!Fruit!I eat an apple a day./I like bananas in my cereal./I prefer a cherry on my sundaes.>>	Options block created in normal fashion, with 'Optional' designator and !groupname! followed by the body text.	Since Fruit = "1,3" (per above example), the second option is excised, leaving the first and third options in the document, leaving I eat an apple a day. I prefer a cherry on my sundaes.
---------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Pathagoras automatically new !groups! for each element of AskOptions list assigning to each a 'true' or 'false' value (depending upon whether it was chosen)

These implicitly created !groups! may be used in the document body in the same way that other <<*Optional* . . . >> blocks are created and used. (Recall that 'Optional block' !groups! contains only a 'true' or 'false' values.

Body Text	Before processing	After processing
<<*Optional*!Apples!*An apple a day keeps the doctor away.>>	Optional block created in normal fashion, with 'Optional' designator and !groupname! followed by the body text.	'Apples' was checked in the AskOptions listing per our example above. A new !groupname! called '!Apples!' was established, and its value set to 'True'. The text will remain.
<<*Optional*!Bananas!*My doctors tells me that bananas are the perfect fruit.>>		Bananas was unchecked in the AskOptions listing. A new !groupname! called '!Bananas!' was established. Its value is 'False'. The text is deleted.
<<*Optional*!Cherries!*Please top my [sundae/milkshake] with a cherry.>>		'Cherries' was checked in the AskOptions listing per our example above. A new !groupname! called '!Cherries!' was established. Its value is set to 'True' and so the text will remain.

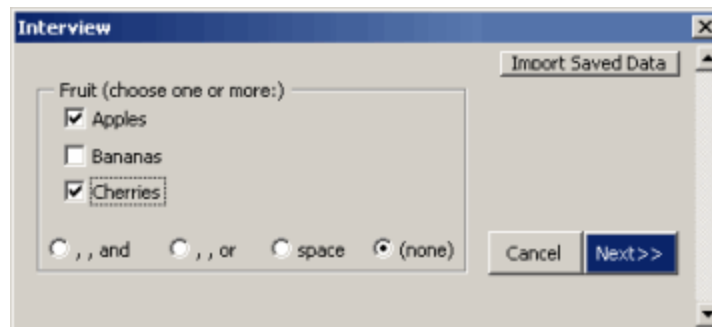
Optional can also accommodate /NEGOPT outcomes.

To enable, just add a '/NEGOPT' (for negative optional) and some 'alternative' text. If the !Group! value of the optional block is True, the text before the /NEGOPT is kept; if false, the text after the /NEGOPT is kept.

Body Text	Before processing	After processing
<<*Optional*!Apples!*An apple a day keeps the doctor away./NEGOPTThose wormy apples didn't keep my doctor away.>>	Optional block created in normal fashion. However, a 'false' value was attached, with '/NEGOPT' separating the 'True' value from the 'False'. (Coloring, italics and underlining for illustration only.)	'Apples' was checked in the AskOptions listing per our example above. A new !groupname! called '!Apples!' was established, and its value was set to 'True', so the 'true' side of the Optional block will remain.

<p><<*Optional*!Bananas!*My doctors tells me that bananas are the perfect fruit./NEGOPTYes, I <u>have no bananas.</u>>></p>	<p>6277</p>	<p>'Bananas' was unchecked in the AskOptions listing per our example above. A new !groupname! called '!Bananas!' was established, and its value was set to 'False', and the 'false' side of the Optional block will remain.</p>
<p><<*Optional*!Cherries!*Please top my [sundae/milkshake] with a cherry./NEGOPTI don't <u>want any cherries. They have pits.</u>>></p>	<p>6277</p>	<p>'Cherries' was checked in the AskOptions listing per our example above. A new !groupname! called '!Cherries!' was established, and its value was set to 'True'. The 'true' side of the Optional block will remain.</p>

--Copy above--



This is what the selection menu would look like when the above text is initially processed.
Not only will Pathagoras assign a value of "1,3" to the GroupName !Fruit!, but it will automatically assign "True" to !Apples! and to !Cherries! and "False" to !Bananas!.

You can then use these values anywhere in the document in the same way any !groupname! is used.

Another Example:

(Can be copied and pasted into document. Press <Alt-P> to process.)

<<*AskOptions(radio)*!Size!Small/Medium/Large/Custom*>>

If 'Small' is selected, Pathagoras records the following values for each !Group! it explicitly and implicitly 'sees':

!Size!="1" (representing it's positional value)

!Small!="True" (reflecting the actual value selected)

`!Medium!="False"; !Large!="False"; !Custom!="False"` (reflecting that they were not selected)

If 'Custom' had been selected, Pathagoras would have created the following values

`!Size!="4"` (representing it's positional value)

`!Small!="False"; !Medium!="False"; !Large!="False"` (reflecting that they were not selected)

`!Custom!="True"` (reflecting the actual value selected)

Optional text blocks placed in the document will be processed accordingly. Example:

`<<*Optional*!Custom!*Please send us the specific measurements (neck, chest, arm length, waist) so that we can custom tailor your jacket.>>`

If 'Small' was selected for `!Size!`, the above optional block would be removed from the document when processed. If 'Custom' was selected, the text would remain.

Simple optional blocks can also be used:

`{!Custom!Please send us the specific measurements (neck, chest, arm length, waist) so that we can custom tailor your jacket. }`

You do need to be a bit careful in naming. If you use an Options `!groupname!` that is identical to on of the 'hard' values, the results may not be as you expect.

By way of example:

`<<*AskOptions(radio)*!Grantors!Grantor/Grantors>>`

Following the 'rule,' if the second option ("Grantors") was selected, `!Grantors!` (initial groupname) would be set to "2", but then Pathagoras would try to set the generated value of `!Grantors!` to "True".

Consider these naming alternatives

`<<*AskOptions(radio)*!Grantors!Single Grantor/Multiple Grantors>>`

or this

`<<*AskOptions(radio)*!Multiple Grantors!Single Grantor/Multiple Grantors>>`

or even something as simple as this (just a '?' added to parent `!groupname!`):

`<<*AskOptions(radio)*!Grantor?!Grantor/Grantors>>`

16.23.4 A 'Legal' Example

Below is a short sample document incorporating a few of the above features. It results in a set of reciprocal Husband and Wife Wills. It is based on the same text presented in the 'Dynamic Creation of Variables' section of the Manual, with more of the document logic features of Pathagoras added. Each of the major features shown in blue links to the section of this Manual that defines and further illustrates them.

Features in use:

[<<*AskOptions*>>](#)^[252] and [!Groups!](#)^[214]

Notes: AskOptions can be created [automatically](#)^[277] from the existing `<<*Options*. . .>>` blocks and placed in an attractive table at the top of the document. Click the above link to see

how. Here, the prompt is used to determine which of the Wills is to be created first. The !GroupName! allows the answer at the top to trigger the Options blocks further down in the document.

<<*If*>>²⁶⁰

Notes: Here, the prompt used to determine if the "Minor Children" question should even be asked. If there are no children, no sense in asking if there are minor children.

<<*AskRepeats*>>²²⁶ with series connectors and !Groups!

Note that once the number of children is determined, it is used in 2 distinct ways. The first is to repeat the [Child. . .] variables the appropriate number of times. When attached to an <<*Options*>> block, the repeat number is repurposed to select a phrase depending upon where there are '0' children, '1' child or '2 or more' children.

You can copy the sample text in the box below by placing your cursor inside the box, pressing Ctrl-A (to select 'all') and then Ctrl-C (for 'copy'). Then paste the text into a document (Ctrl-V) (this one has no mnemonic equivalent) . Process the page (<Alt-P>) to see how it all comes together.

```
<<*AskOptions*!Whose Will!First Is Husband's/First Is Wife's*>>
<<*AskOptional*!Children?!Are there children?*>>
<<*If*!Children?!="Yes",<<*AskRepeat*!NumChildren!How many children*>>,!minors!="No">>
<<*If*!Children?!="Yes",<<*AskOptional*!Minors!Are there minor children*>>,!minors!="No">>
```

Last Will and Testament of

<<*Options*!Whose Will!*[HUSBAND NAME]/[WIFE NAME]>>

I, <<*Options*!Whose Will!*[Husband Name]/[Wife Name]>>, being of sound mind, make this document my Last Will and Testament.

1. I give all of my property and estate to my <<*Options*!Whose Will!*wife/husband>>, <<*Options*!Whose Will!*[Wife Name]/[Husband Name]>>
2. <<*Optional*!Children?!*If my <<*Options*!Whose Will!*wife/husband>>, does not survive me, I give all of my property to my <<*Options*!NumChildren!*/child/children <<*Repeat(and) *!NumChildren!*[Child@Name], born [Child@DOB]>>.>>>>
3. I appoint my <<*Options*!Whose Will!*wife/husband>>, <<*Options*!Whose Will!*[Wife Name]/[Husband Name]>> to be my Personal Representative.
4. <<*Optional*!Minors!*If at the time of my death I have any minor children, I appoint [Names of Guardians] to be the Guardians of such minor children.>>

<<*Options*!Whose Will!*[Husband Name]/[Wife Name]>>

====Page Break=====

Last Will and Testament of

<<*Options*!Whose Will!*[WIFE NAME]/[HUSBAND NAME]>>

I, <<*Options*!Whose Will!*[Wife Name]/[Husband Name]>>, being of sound mind, make this document my Last Will and Testament.

1. I give all of my property and estate to my <<*Options*!Whose Will!*husband/wife>>, <<*Options*!Whose Will!*[Husband Name]/[Wife Name]>>.

2. <<*Optional*!Children?!*If my <<*Options*!Whose Will!*husband/wife>>, does not survive me, I give all of my property to my <<*Options*!NumChildren!* /child/children <<*Repeat(and)*!NumChildren!*[Child@Name], born [Child@DOB]>>.>>>>

3. I appoint my <<*Options*!Whose Will!*husband/wife>>, <<*Options*!Whose Will!*[Husband Name]/[Wife Name]>> to be my Personal Representative.

4. <<*Optional*!Minors!*If at the time of my death I have any minor children, I appoint [Names of Guardians] to be the Guardians of such minor children.>>

<<*Options*!Whose Will!*[Wife Name]/[Husband Name]>>

See these other examples:

[Legal pleadings: Captions using <<*AskRepeat* . . .>>](#) ²⁴³

[Mixing 'Standard' and 'Simplified' Options Blocks](#) ²⁰⁷

['Repeat' Blocks](#) ²²⁴

16.23.5 Anatomy of an AskTable

Below is a sample of an Ask table that contains a wide variable of elements. It is a 'perfect' sample for discussion.

1	<<*AskOptions(radio)*!ClientSex!Client = Male/Client = Female*>>
2	<<*AskOptions(radio)*!Married?!Married/Single*>>
3	<<*If*!Married!="1"and !ClientSex!="1",!SpType!="2(2)",!SpType!="1(2)">>
4	<<*AskRepeat*!NumCdn!Number of Children*>>
5	<<*If*!NumCdn!="0",<<*AskOptional*!AllAdults!All children adults?*>>>>
6	<<*AskOptions(radio)*!ResTrust!Residual Trust/No Residual Trust*>>
7	<<*AskOptions(radio)*!T is Trustee?!Testator is Trustee/Testator not Trustee*>>
8	<<*AskOptions(radio)*!TrSign!Trust signed today/Trust signed earlier*>>
9	<<*If*!ResTrust!="1",!Trustee!="True",!Trustee!="False">>
10	<<*AskOptions(radio)*!POA!Powers of Appointment Not Exercised/Powers of Appointment Exercised*>>
11	<<*AskRepeat*!ExecNum![Executor]>>
12	<<*AskOptions(radio)*!corpExec!Corporate Executor/No Corporate Executor*>>
13	<<*AskOptional*!MultiSuccessorExecs!Multiple successor executors?*>>

14	<code><<*If*!AllAdults!="True",!Minors!="False",!Minors!="True">></code>
15	<code><<*If*!NumCdn!>"0" and !AllAdults!="True",!Guardian!="True",!Guardian!="False">></code>

1. Simply determines the sex of the client. Not the (radio) attribute. That insures that, when the Interview is presented, then user can select only one of the two choices.
2. Determines marital status and assigns it to the groupname !Married?!
3. The (*If*) line compares two previous values to determine a third. The determined value is clear in the underlying document – !Sp Type! will be either 'Husband' or 'Wife'. However, in logic equations such as this, we must speak in terms of position, not absolute value. Here is how the line is evaluated:
 - If the value assigned to !Married! is the position 1 answer (in this case "Married") AND if the !ClientSex! answer is "Male" (also the position 1 answer in its group), then the !SpType! will be the second choice ('Wife').
 - If the !ClientSex! is '2', then !SpType! will become '1' (or 'Husband')
 - The number in parentheses is optional. It is used to indicate how many choices that the !groupname! presents in the main document. In this case 2. While optional, it is much preferred.
4. Line 4 asks for the number of repeating variable given the groupname !NumCdn! (number of children)
5. If the number assigned to the groupname !NumCdn! is 0, then it assigns the 'true' value to 'nothing', but this 'nothing' still needs to be stated. That is done simple with a 'blank' comma. That way, Pathagoras can know where the 'false' value starts. (Note: this could also be re-written with a different comparator:
`<<*If*!NumCdn!>"0",<<*AskOptional*!AllAdults!All children adults?*>>,>>`
6. Line 15 contains a multiple comparisons, connected by 'and'. It is just another way of stating lines 5 and 14.

Notes:

- a. !GroupNames! can be anything. They should help you and the end user (if the end user might see the Ask Table) to determine the purpose of the group. The question mark is not a mandatory element, but may be helpful. (The same groupname could have been !Married! or !Client is Married! or anything else.
- b. Typically, the shorter the !GroupName! the better, but only because it takes up less space. So !ST! arguably would be better than !SpouseType!. The drawback is that a later user may not understand what !ST! stood for and may become frustrated. So, in the example, we settled on !SpType!
- c. The placement of the various lines within the Ask block are not critical, so long as precedent values have been assigned before a dependent value is determined. So line 3 above could actually be anywhere, including line 11, so long as the values that line needs to complete the evaluation (in this case, 1 and 2) precede it.
- d. The evaluation process uses classic Boolean logic structure. If 'this', then 'that', else 'other that'. The astute reader will observe that the logic equation in 3 is imperfect. If the parties are not married and ClientSex = 1 (or 2), the equation will evaluate to the 'false' side, SpType = 1. However, as the document is structured, the value is acceptable because in actuality, it doesn't matter. All 'married' references are processed out of the document when the 'Client not married' selection was made, and it simply does not matter if the client

- is not married to a Husband or not married to a Wife. The same pertains to line 5. If there are no children, it doesn't matter if selection assigned to 'AllChildren' is Adults.
- e. As Pathagoras reads down the Ask table, it will stop at and evaluate all 'If' lines. If the precedent values have already been assigned, it will proceed forward. If not, it will stop, and present another Interview screen for completion of the required variables.
 - f. If the value assigned to an undeclared GroupName is 'True' or 'False', the type of block in the underlying document will always be "<<*Optional*>>", with 2 and only 2 possible values: True or False (or Keep or Delete). Otherwise, it will be assigned as an "<<*Options*>>" block, with the number of elements as stated in the parenthesis, if provided.



An <<*Ask. . .*>> table comprises plain Word text. You can edit it to suit your office's needs. Just highlight, cut and paste until you have the order of questions you wish to present to the end user.

16.24 Displaying a Position

When a value is assigned to a !groupname!, that value is the positional value of the selection. So if you listed all 50 United States in alphabetical order (and for some reason you did not use an *Alias*), and selected 'Florida', the !groupname! assigned to that element would be set to the value "9". But the list of States is long. Other lists can be long. Do you have to count positions each and every time?!? No, of course not.

You can ask Pathagoras to pre-pend as hidden text a number reflecting the position of each item in an Options list. You can turn on and off the display of the hidden numbers by clicking on the pilcro.

Highlight the list and click over to "Pathagoras Features | Authoring/Editing Tools | Miscellaneous". Select the 'Add Hidden #s to Choices'. If you add more choices to the list, you can also 'Delete' the existing numbers and the re-apply.

Don't forget that you can add these tools to your Alt-Q menu. Look for 'Add Hidden #s' and Del Hidden #s.)

The Pathagoras System

Document Calls

Part

XVII

17 Document Calls

Whenever Pathagoras encounters text enclosed within "<<" and ">>" (double angle brackets, or 'DABs'), it will attempt to process what it finds in between those brackets.

- If Pathagoras encounters an [<<*Options*>>](#)^[176], [<<*Optional*>>](#)^[172] or [<<*Repeats*>>](#)^[224] block during a document assembly session, it knows to process the block in accordance with rules associated with each kind of block.
- If Pathagoras sees '!' marks on each end of the DAB text, it presumes you want to return the numeric value of the '!groupname!' associated with a Repeat call. Pathagoras will return the spelled out value of the number assigned to that !groupname! (See [this link](#)^[233] for how to return other formats for a Repeat !groupname!.)
- If there are no special characters (asterisks or exclamation marks) Pathagoras will assume that the text represents a **document name**. Pathagoras will attempt to locate the target document following Pathagoras' [Order of Search](#)^[304] rules. If located, Pathagoras will insert the text of the called document in place of the call. **Document calls** is the primary topic of this and the following sections.


Qualified call: This is a qualified call: [<<c:\office forms\real estate\Listing Agreement.docx>>](#). The full path to the target document is provided, indicating that only that specific document is desired. If the document is present at that location, Pathagoras will insert a copy of it into the document at the location of the **document call**. If not found, a 'not found' error report will display.

Unqualified call: If just [<<Listing Agreement>>](#) was used as the call, it will hunt for the document 'Listing agreement.doc' (or .docx) in a multitude of locations, following ['Hunt Path \(Order of Search\)'](#)^[304] rules. If and when found, Pathagoras will insert the text of the target document in place of the **call**. If multiple documents with the same name resided in the Hunt Path, Pathagoras inserts the first one it finds and stops looking. If not found in any Hunt Path location, a 'not found' error report will display

A 'document call' need not be to just documents. However, you must add at least an extension so Pathagoras will know the type of file you are seeking.

So, with the above firmly in mind, you can use [<<document name>>](#) references to insert:

- Documents
- Images
- PDF files

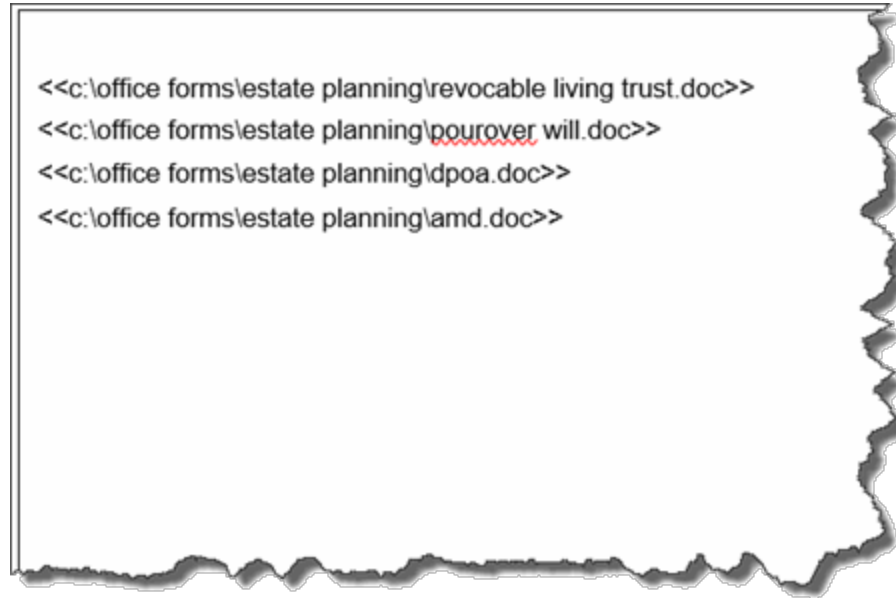
 **NOTE:** Document calls can be nested within [<<*Options/Optional* . . .>>](#) commands. So long as the 'resulting' text (after the Option or Optional is selected) is a [<<document call>>](#), Pathagoras will find it. Here is an example:

```
<<*Options(radio)*Order Amount:$0 - $100/$101-$200/$201-$500/$501+*<<Full
charges>>/<<20pct discount>>/<<50 pct discount>>/<<Free shipping>>>>
```

17.1 Document Packages

'Packages' of <<document calls>> can be easily composed, giving you an entire document with simply references to the desired text. Just list the <<documents calls>> you want the package to return. Save the document with a name that describes its content. (It is not necessary that the package contain only <<document calls>>. You can have as much 'other text' that you want.)

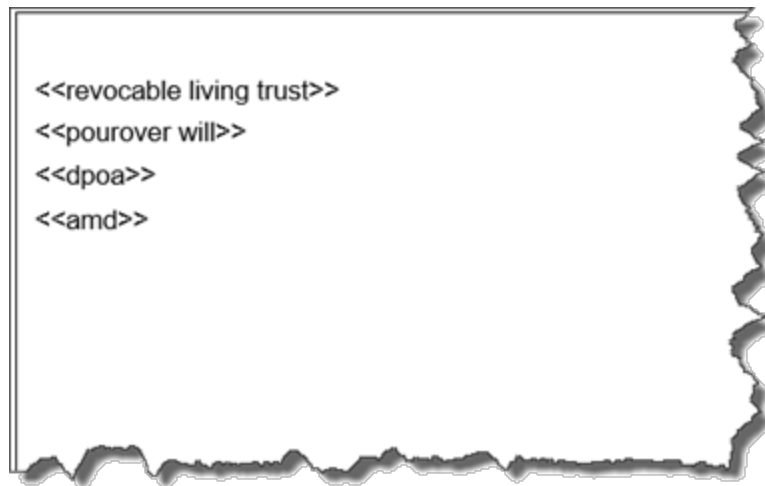
For example, let's say you regularly create estate planning packages comprising a Revocable Living Trust, Pour-over Will, Power of Attorney, and Advanced Medical Directive. You could create a single document that contains a series of document calls. That document might look like this:



A document package with fully qualified document calls.

Save the document as, for example "Estate Planning Package". When the document is recalled and processed, Pathagoras will locate the individual documents and insert them in place of the 'call'. Very quickly, you will have a complete package of documents.

If the documents comprising the package are within the 'hunt path' (such as in a DropDown List; see [Order of Search](#)³⁰⁴ rules), you don't even need to include the drive and path. Pathagoras will find it automatically and insert the text in place of the call. The whole document could look like this:



A document package with just name references.

If the document is in the Pathagoras 'hunt path,' Pathagoras will find in less than a second.

This 'type' of document package is separate from, but complementary to, the document packages that you can create from the Clause Selection Screen. See discussion at this link.

The possible uses of <<Document Name>> references are limitless. Here are a few Tips and ideas to consider.

In the above examples, 'complete' documents were used to create the document package. However, there is no restriction as to what a document package can be. The individual documents could just as easily have been individual clauses that make up a classic "I Love You" Will. E.g.,

```
<<wil100>>
<will10>>
<<wil23>>
<<wil25>>
<<wil45>>
<<wil60>>
<<wil221>>
```



If you have a DropDown List of clauses that you would like to use to insert <<document name>> references, cycle to the <<**Insert Name**>> option in the upper left quadrant of the DropDown Lists section. As you click on an item in the DropDown List, its name, surrounded by DABs will be inserted. There is no quicker way to create a Document Package.

17.2 Order of Search

Pathagoras has a definite 'order of search' pattern of which you should be aware to maximize efficiency in naming and recalling your clauses. When a term is recalled (either from a Clause Set or via <Alt-G>) Pathagoras will look in the following locations in the order listed (this is also called the 'Hunt Path'):

- If the term is '**fully qualified**,' Pathagoras will look in the designated location and no further. ('Fully qualified' means that the document is pointed to by drive, folder, sub-folders(s) and name. E.g., "c:\office forms\estate planning\trusts\living trust clauses\rlt134.doc".)
- If not fully qualified, Pathagoras will first look in the folder from which the base document is located. (Applicable when base document was called via the Document Assembly routine or from a DropDown List.)
- If Pathagoras sees only a 5 to 8 character term name, it will determine if the term name follows the 'prefix/suffix naming' criteria (two to four letters followed by three or four numbers). If it does, it will look in the **book associated with the prefix** in the Prefix Table.
- If the text is a number from 1 to 12, Pathagoras will open the folder paired with the number in your current profile in PathSmart.
- If the text is a name stored in your QuickLinks list, Pathagoras will open the folder paired with the name stored in your QuickLinks list.
- Pathagoras will then look to see if the term name exists in the '**Tag Along**' section of the document. (Here, the desired text must have been previously 'bookmarked' with the term name.)
- If not found in the above locations, or if the term is not a prefixed name, Pathagoras will look in the **Position #1 Book** (The 'Position 1' Book is the one that occupies the first bookshelf in the Current Library.)
- If not yet found, Pathagoras will look in the **SuperFolder** (if designated).
- If still not found, Pathagoras will look in the **SuperGlossary** (if designated).
*(You can tell Pathagoras to look in a **SuperBook** before the **Position #1 Book**. Check the box found in Utilities/Settings / File Locations labeled "Check SuperBooks" first.)*
- If still not found, Pathagoras will look for the clause in each active DropDown List (and if such a sub-folder exists, in the sub-folder called "Hidden")
- If still not found, Pathagoras will look for the term in your Auto-Correct dictionary (Microsoft Word feature).
- If still not found, Pathagoras will look in the first 15 Glossaries that you may have stored in your SuperFolder. (This last search presumes that you have not earlier looked in these glossaries via the prefix search.)



To avoid redundant searches (and to speed up the search process), do not store the following items in your SuperFolder:

- (1) your SuperGlossary or
- (2) any Glossary or any term which you have associated with a prefix.

See also:

Prefix/Suffix Naming Convention

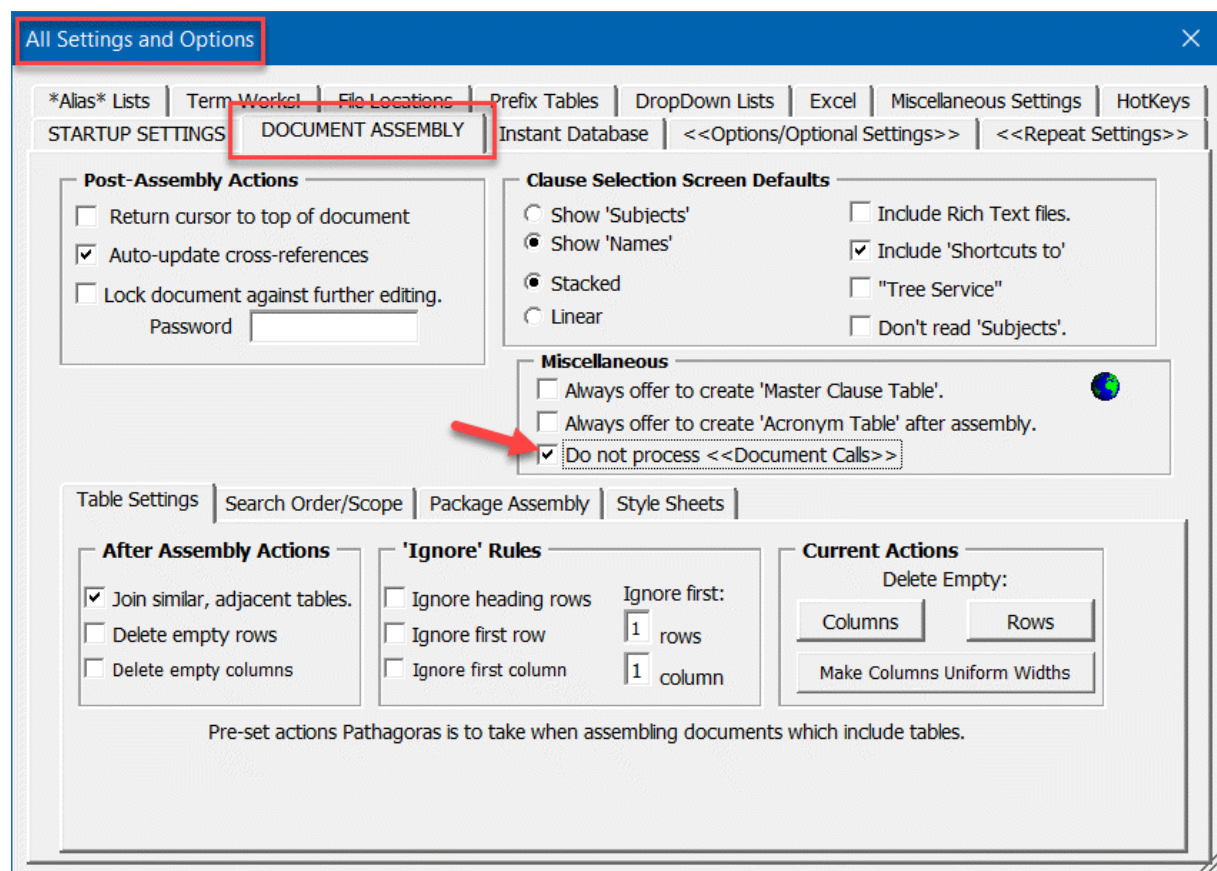
Prefix Registration Table

SuperBooks

Tag Along clauses

17.3 Disabling

In some offices, double angle brackets are used to highlight descriptive or instructional text and modifying the boundaries of that text is not practical or desired. In such case, you will want to disable the ability of Pathagoras to process any DAB text beyond <<*Options* . . .>>, <<*Optional* . . .>> and <<*Repeat* . . .>> blocks. To disable, go to Pathagoras Settings | All Settings. Click the 'Do not process <<Document Calls>>' box.



Click to collapse.

17.4 Interview Documents

This is not really an 'interview' in the sense discussed above. (In the above examples, the interview questions are placed at the top of a source document, and values are assigned and processed.) Here, we are creating essentially a document titled "Interview" with a single question that results in the selection of an appropriate second document. This is presented here as an alternative to the more elaborate Interview process described in above sections. It is yet another way in the Pathagoras arsenal to guide an end-user to an ultimate 'destination.'

Here is an example:

1. Create a document called "Will Interview." This document will contain a single Options text block consisting of one multi-part question, with a series of document references enclosed within "<<" and ">>" brackets being the 'answers'. (It can also contain anything else you want.)

```
<<*Options*!Family Structure!Married with one or more children/Married, no
children/Single (divorced or never married), with one or more children/Single (divorced or
never married), no children*<<Married with Children>>/<<Married no
Children>>/<<Single with Children>>/<<Single No Children>>>>
```

The 'administrative section (the text in blue) provides 'questions' that will be presented to the end user when that user calls up the "Will Interview". The remainder of the text block provides the references to the document that will be called based on the selection made. The "<<" and ">>" marks surrounding the document names indicates that you intend a document (or glossary clause) by that name to be called. (Without those boundary markers, you would get just text.)

Of course, there must actually be documents or glossary terms called "Married with Children.doc", "Married no Children.doc", etc. which respectively provide the text appropriate for "Married with one or more children", "Married, no Children" etc., wills.

2. The answer to the options posed in the 'first' document will result in a call to an appropriate 'second' document. This second document can contain 'terminal' questions (the full Interview, as it were) or can lead to additional documents in the Interview sequence. There is no limit to the nestings and references that can be provided and performed.

Notes:

- Regular <<*Options*>> and <<*Optional*>> [construction rules](#)^[172] apply.
- The references can be to a complete document/template (as suggested in the above example) but they also can be to Clause Sets.
- If the reference is composed using the prefix/suffix naming convention (as shown in the example), or if the referenced clauses are stored in the Super Folder or Super Glossary, Pathagoras can quickly find desired clause without more in the clause name. See [Search Order Rules](#)^[304].
- If the reference is not in prefix/suffix style, and the referenced clauses are not in the Super Folder or Super Glossary, the target clause either (1) must be found in the same folder as the initial Interview form (and this is likely to be true) or (2) must be fully qualify (with drive and folder designators).

17.5 Other DAB functions

DAB text can be called in as the **value to a variable**. In this way, you theoretically could use your Instant Database to create entire documents.

Typically the value of a variable is a name, address, size, color, quantity, etc.. Instead of those value 'types,' you want to provide the entire content of a document (perhaps property description, or a product description. You can simply type as the variable's value a <<document name>>

reference. Hopefully it is easy to see how this opens up a much wider range of document assembly possibilities. (NOTE: You do not even have to know how to spell the target document. If you press shift-click in the Instant Database text box where you want to insert the document call, Pathagoras will present a window that lets you navigate to the file.)

When the variable is processed, it is replaced by "<<document name>>." On its next pass through the document, Pathagoras will 'see' that value and (because it is DAB text), and process it. Following the normal [Order of Search](#)^[304] rules, Pathagoras will locate the desired text and place it onto the screen. It quite literally is blink of an eye fast.

Further, you could include in your document the following: "[<<document name1>>/<<document name2>>/<<document name3>>]" as a **multiple choice** variable, and offer the end user a choice of 'documents' to insert. And still further, you could use **!groups!** to tie a multiple choice variable such as the above to the answer to a preceding multiple choice variable.

The possibilities are endless. One more: If you use Excel to bring in values to variables, a value that you can insert in your Excel spreadsheet might be a <<document name>>. Keep in mind that Pathagoras doesn't care how the value gets inserted into the document. If it's there between double angle brackets, Pathagoras will process it.

DAB text can be **nested within an <<*Options/Optional*>> block**. This opens up a wide range of document assembly possibilities. By properly constructing the Options/Optional block, you can pose a simple question to the end user within the Options block which, when answered, calls in the appropriate document or clause. For example, let's assume the following <<*Options*>> text block resides in the source clause of a letter being written to a purchaser of goods. The purpose of the clause is advise the recipient of the letter what the shipping costs would be in various situations. Assume names within the DABs are existing documents in the user's system.

```
<<*Options(radio)*Order Amount:$0 - $100/$101-$200/$201-$500/$501 + *<<Full
charges>>/<<20pct discount>>/<<50 pct discount>>/<<Free shipping>>>>
```

When the above text block is encountered during Pathagoras' top-to-bottom 'processing' of the document, Pathagoras will present the 'questions' "\$0 - \$100" "\$101-\$200" "\$201-\$500" and "\$501 +" onto a pop-up form for selection. The user makes the appropriate selection and based on that selection, the appropriate <<document name>> value is returned to the screen (albeit only briefly).

On its next pass through the document, Pathagoras then 'sees' that value and (because it is DAB text), begins to process it. Following the normal [Order of Search](#)^[304] rules, Pathagoras locates and places the desired text onto the screen. It quite literally is blink of an eye fast.

iNote: The "<<" and ">>" marks are not special keyboard or ASCII code. They are the 'lesser than' and 'greater than' characters above the 'comma' and the 'period' on your keyboard.

i While not precisely the same, implementation of above can bring powerful "If . . . / Then . . ." logic to your documents. See a further discussion of this at this link.

See Also:

Suppress Processing

[Order of Search Rules](#)^[304]

Clause Sets

If. . . / Then . . . logic

Instant Database: Double Angle Bracket Functions

The Pathagoras System

Document 'Dis'-assembly

Part



XVIII

18 Document 'Dis'-assembly

Document 'dis-assembly' is the process by which an existing document is deconstructed into its component parts. The individual pieces are saved as separate clauses ('building blocks') in a designated folder or glossary. These pieces form the clauses from which a wide variety of new documents can be assembled.

Building documents from 'building blocks' (clauses) is one of the two basic approaches to 'document assembly.' The alternative method is 'template based' assembly. There, you start with a complete (actually, overbuilt) document and, by answering questions strategically placed within the document, you remove those portions of the template that are not needed for the project.

Which approach you use is one of the more important choices you will make in setting up your system. Each has its advantages. It really is a personal choice. Pathagoras works well with either. (The Main Manual has a fuller discussion and comparison of clause-based ('building blocks') versus template based document assembly.

You probably already have many documents that are ideal for dis-assembly. The ones found in your 'office forms' folder certainly qualify. So do the ones found in treatises and form books to which you subscribe, or those that you received at Continuing Professional Education courses.

Some easy dis-assembly techniques are discussed in the following pages. All techniques for disassembling documents into building blocks are discussed and fully illustrated in the separate pamphlet called [Document Disassembly \(Creating Building Blocks from Whole Documents\)](#).

18.1 Manually

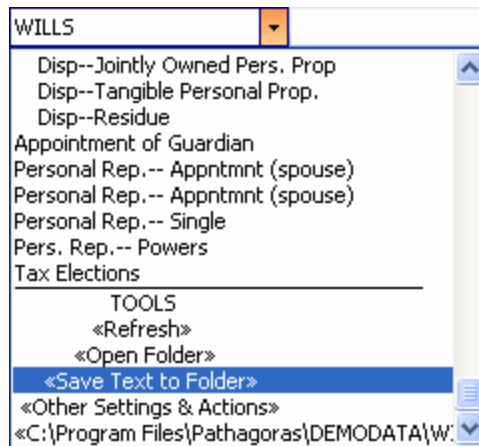
Document dis-assembly can be nothing more than highlighting a piece of text that you want to save as a separate clause, pasting it into a new document, and then saving that new document into the appropriate folder. The next time you display a book using the Document Assembly button, the new document will instantly and automatically display alongside the other documents in the folder. (If a DropDown List is using that folder as the source of its content, you may have to 'Refresh' the DropDown List for the new document to show. But this is a one-time only refresh.)

The dis-assembly techniques discussed below are simply automated versions of the above process.

See this discussion on [Last Paragraph issues](#)³¹⁴.

18.2 Using DropDown List

Highlight the text you want to save. Click on the DropDown List into which you wish to save the highlighted text. Locate the element in the Tools section near the bottom of the list titled 'Save text to Folder'.

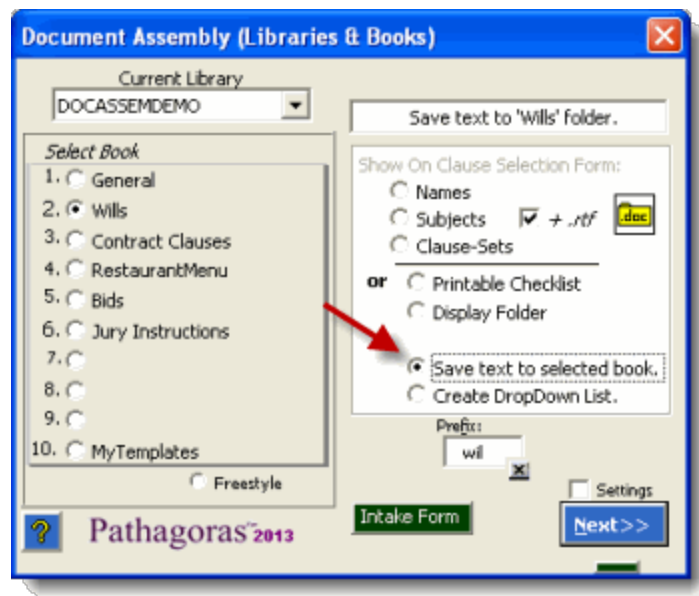


Follow the prompts to name the document and provide an optional subject. The new document will be present in the DropDown List then next time you display it as well as being present in the standard Word/Windows Explorer screens if you later choose to navigate that way.

See this discussion on [Last Paragraph issues](#) ³¹⁴.

18.3 Using Libraries & Books Screen

Highlight the text you want to save. Click the Document Assembly button. When Libraries & Books Screen displays, and select the book into which you wish to save the highlighted text. The screen expands to show you several choices. Click the one titled "Save text to selected book."



Press Next and follow the prompts to name the document and provide an optional subject.

18.4 Last Paragraph Issues

As you re-assemble 'dis-assembled' documents, the formatting of the new (or augmented) document may not appear the way you wish. This is invariably a function of the styles and formatting of the base documents. See Styles discussion for the benefits and the frustrations of this crucial aspect of Word.

Therefore, before you assemble your first document from your new collection of dis-assembled text, take a peek at each document. Look especially at the formatting, indenting, etc. associated with the last paragraph marker. (If you don't 'see' it, click the 'Show All' button (the pilcrow character (¶)) on the Home tab. It is that last paragraph marker that affect the appearance of not just the last paragraph, but the next paragraph you type or call in. If the 'Enter' on the last line of the document suggests a number, bullet, tab, indent or otherwise a modification from 'Normal,' you need to consider if this is what you want.

If you want the next paragraph that you type or insert (or that will be called from document assembly selections) to be independent of the previous text, make sure that you add a new 'blank and normal'. Be sure to 'un'- number, 'un'-bullet, 'un'-indent, 'un'-tab or otherwise undo the formatting so that the next block of text comes in 'clean.'

If after all that undoing, you still have something you don't like, use the 'Format Painter' to normalize things. Click 'Format Painter' in a 'normal' section of text to capture that 'good' formatting. The 'Format Painter' button will be toggled 'depressed' (i.e., 'on'). Move your mouse to over the last line of the document and left-click the mouse. That should apply the proper formatting.

18.5 Automatic Paragraph Numbering

A beautiful sight to behold during paragraph assembly is inserting a paragraph(s) that contains automatic paragraph numbering controls and seeing everything ending up perfectly numbered even though the original number of paragraphs is not the same as the final number. Plus, when you need to add another paragraph (whether at the end of an existing list, or smack-dab in the middle of it), the paragraphs end up properly renumbered.

To give credit where credit is due, the magic of all of this is all Word, and 'none' Pathagoras. But even Word cannot handle it if the numbering styles (and sometimes even the numbering source) among the paragraphs being assembled or the clause being inserted, are not consistent.

So when you disassemble a clause with an APN, if you intend to use that paragraph with other paragraphs that following the same numbering scheme, you will be perfectly ok. But don't expect a paragraph introduced by a bullet, or a paragraph from an 'outline' that you want to insert in a simple numbered list to 'behave' the way you want. It cannot (except accidentally so). The schemes do not match, and Word (and Pathagoras) cannot put that square peg into the round hole.

If you are trying to disassemble a piece of text that currently contains APN coding that you want to insert in a wide variety of settings, you won't want to include the paragraph marker (the pilcrow) that closes the paragraph. The coding that controls APN is in that pilcrow. (Note: If you include the pilcrow, the text will still come in just fine during document assembly or via a DropDown Lists, but likely will not be numbered, indented, bulleted, etc. the way you expect.

You can always renumber or correct bad numbering using Word tools. The 'Format Painter' is likely the easiest tool. You might also force the added text into the previous, properly numbered paragraph, and then hit 'Enter'. That 'Enter' should cause the APN of the previous paragraph to be applied to the new text.

The Pathagoras System

Debugging Variables & Options Text

Part



XIX

19 Debugging Variables & Options Text

Pathagoras provides several tools to help you to 'debug' problems with variables, robust <<*options*>> and {simple options} text.

Paint the [brackets] and <<*Options*...>> blocks:

Pathagoras can 'paint' the the various brackets, boundary markers and other coding element so it is easier to distinguish them from the surrounding text. Click 'Pathagoras Features | Authoring/Editing | General Editing Tools | Mark-up Tools' and click 'Paint' (See red arrow in Figure below.)

- 'Paint' is also a default tool in the Alt-Q (QuickPicks) menu. (This is the quickest way to 'paint' if you've not reassigned QuickPicks.)
- You can also 'paint' by typing the word 'paint' on a blank line of the document followed by <Alt-G>.

Structure Checker:

To determine if you have an equal number of brackets in the document, click **Pathagoras Features | Wizards | Structure Checker**.

- Structure Checker is also a default tool in the Alt-Q (QuickPicks) menu. (This is the quickest way to 'paint' if you've not reassigned QuickPicks.)

Other Debugging tips:

Copy a smaller section of the document (especially if you believe you know where the error point resides) into a new document and process from there. I

The Pathagoras System

QuickLinks

Part



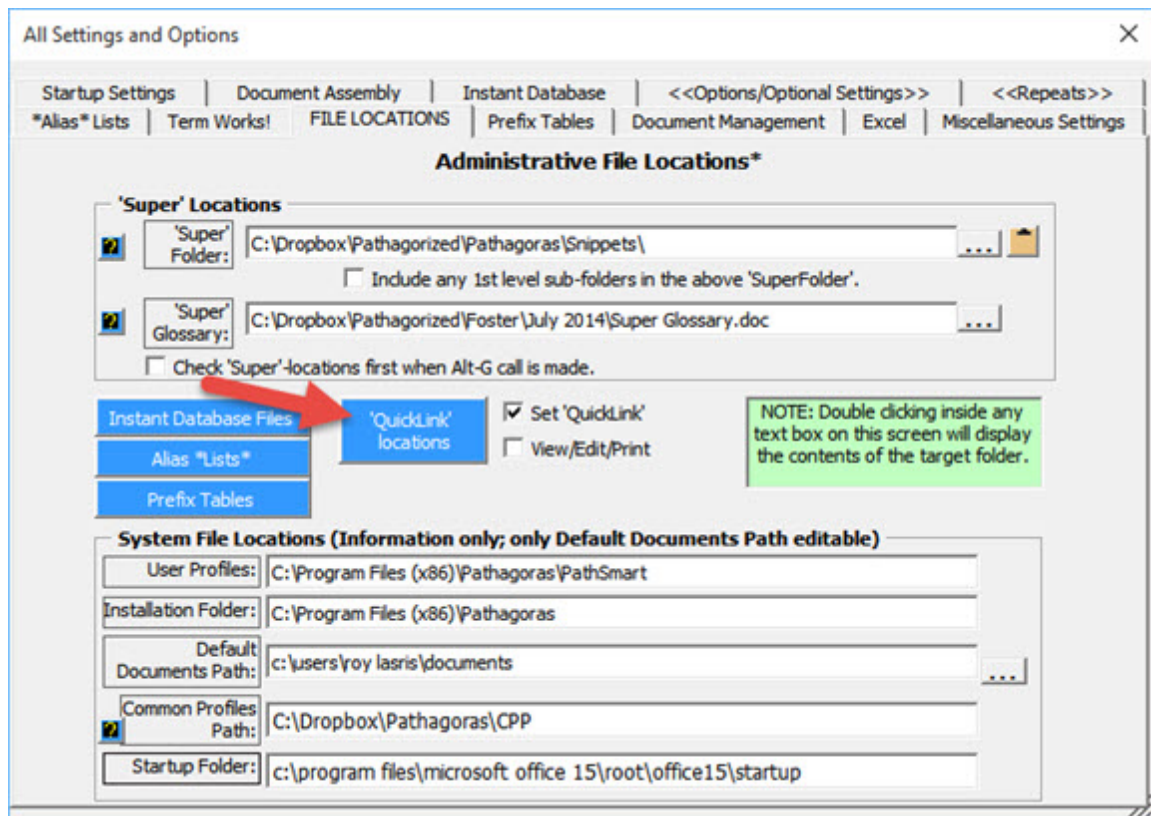
XX

20 QuickLinks

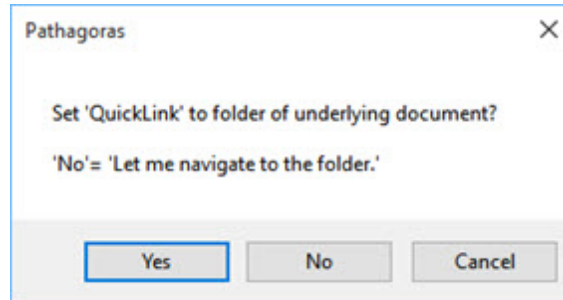
QuickLinks allows you to pair a word or short phrase with a folder you wish to quickly display. Think 'clients'. Type the QuickLink name, press <Alt-G>. and Pathagoras will instantly display the target folder. No single feature will save you more time navigating your system

Creating a QuickLink, Method 1:

1. Click the Pathagoras Settings screen and then the **All Settings** button.
2. Click the 'File Locations' tab.
3. Click the **QuickLink** button. (Make sure the 'Set QuickLink' box to the right is checked.)



3. Answer the prompt that appears next (shown below). If the underlying document is in fact in the folder you want to set, say 'Yes'. Otherwise, say 'No' and then navigate to the target **QuickLink** folder.



4. Once you have navigated to the target folder, press the '**Select**' button to lock in the choice.
5. At the next prompt, give your **QuickLink** a name. Any term will do. It can be a single word or a meaningful phrase. Press OK and Pathagoras will confirm your QuickLink name and its target.

Creating a QuickLink, Method 2:

1. Type the desired (but as yet unassigned) QuickLink term onto a blank line on an editing screen. Press <Alt-G>.
2. Pathagoras will search through the [Hunt Path](#)³⁰⁴ looking for a document of that name. (<Alt-G> is Pathagoras' 'g'et command.) Because it (presumably) didn't find a matching document or other trigger term, Pathagoras will let you know of this 'failure.'. It will present a list of options, the 4th one being 'Create a QuickLink.' Select that one..
3. Follow the resulting prompts and navigate to the target folder. Press 'Select' when there. that locks in the selected folder. All done. (NOTE: The target folder can be located anywhere on your system. C:/ drive; network drive, anywhere.)

Using your QuickLink:

To Open Folders:

Simply type your **QuickLink** name or phrase on any line at the left margin. Press <Alt-G>.

To Open Sub-folders:

If you want a sub-folder of the primary target, type the QuickLink name followed by a slash and at least the first few characters of the sub-folder. *E.g.*, '**clients\jones, ro**'. Press <Alt-G>.

- If you typed the full sub-folder name, Pathagoras will display the File Open dialog with the sub-folder displayed.
- If you typed a partial name (e.g., **clients\jo**), Pathagoras will display a dialog displaying those folders that start with what you have typed. Select the desired sub- folder from the list.
- If you totally mistyped the name, Pathagoras will display the parent folder.
- If you have set a 'default folder' (see Main Manual for those instructions), you can just type a slash followed by <Alt-G>. If you want a sub-folder, type a slash, the first letter or letters of the sub-folder, followed by Alt-G. *E.g.*, **\jo'<Alt-G>**.

To Save files:

If you want to **SAVE** your on-screen document **TO** your **QuickLink** target folder, type the **QuickLink** name on any line at the left margin and press <Alt-S>. A normal Word/Windows Save As screen will open, displaying the target folder. Name your document in the normal fashion. (Don't worry about the **QuickLink** you typed. Pathagoras will erase it, and the line it occupies before you save the document.)

[Editing QuickLinks](#)

Display the QuickLinks menu (see step 1, above). Select the 'View/Edit/Print' box before pressing the 'QuickLinks' button. A 'NotePad' screen that will next appear. Just follow the same pattern: "*QL Name*", "*QL Path*" (two parts separated by a comma, enclosed in two sets of quotes) to edit or even add QuickLinks

[Notes:](#)

QuickLinks are computer specific. Multi-license owners will have to set **QuickLinks** for each computer.

Even if the target folder assigned to your **QuickLink** is not precisely the one you want, it may be a closer starting point to your goal than navigating from ground zero. Consider using **QuickLinks** as a regular navigation tool.

The Pathagoras System

Support

Part



XXI

21 Support

21.1 Customer Service

Pathagoras prides itself on providing prompt, useful and personal customer service. While we hope that this Manual and the other instructional materials are helpful, you can still count on the 'personal touch' of Pathagoras' customer service as a 'first line' of help as well.

We truly enjoy hearing from our customers and potential customers. While we cannot promise that you will never receive a voice-messaging service if you call us, more likely you will receive a live person at the other end.

But if you do get our voice mail, just leave a message. We will call you back promptly.

Contact information is spread across as many places as we could find so that you do not have to hunt for an email address or telephone number, and we repeat it here. Let us know if we can ever be of service.



Pathagoras
Innovative Software Products of VA, LLC
Roy Lasris, President

E-Mail: info@pathagoras.com (*I personally read all e-mails sent to this address!*)

Website: www.pathagoras.com

telephone #s: +1 866-PATHAGOras (1-866-728-4246) (tollfree)
 +1 (757) 877-2244 (USA) (direct line, not tollfree)
 +1 (757) 898-7374 (evenings, weekends)

Address:
117 Chisman Landing
Seaford, VA 23696 USA (I read all mail)

Reporting a 'bug':

We ask you to report any and all program 'bugs' you encounter. This includes 'suspected bugs.' If you have had a problem, do not hesitate to tell us about it.

- Most error messages that Pathagoras and Word generate are rather generic in nature, and most of the time not helpful in pinpointing the precise source of the problem. If you receive an error message while Pathagoras is in operation, and can duplicate it at will, please send us a list of the steps that will generate the error. If we can duplicate it on our end, we can come up with a much quicker fix.
- We would also greatly appreciate your sending any screen shots that may help us to better identify the problem.

- Sending us actual text of documents that didn't quite work would be good too. (We will not share your documents with anyone. We understand the proprietary nature of your personal or business work.)

Remote Assistance:

If you call us for guidance, or to report a bug that can be best explained visually, we likely can most effectively address the situation via 'virtual presence.' In other words, we can (and would like to) virtually sit at your machine. That way we can see the same thing that you are seeing at the same time that you are seeing it.

- This can be readily accomplished via a wide variety of remote assistance tools.
- Our preferred meeting tool is GoToMeeting®, a product of the Citrix Corporation. It is easy to use, offers quick response time and allows us to sit 'together' while we share information on your computer.
- If you feel a virtual meeting would be helpful, don't hesitate to say "Can we do a GoToMeeting?" (If you are otherwise covered under the Annual Support Agreement, there is no charge for the meeting. And don't worry about costs on our end. We have none beyond the annual subscription that we pay for GoToMeeting.)
- To activate remote assistance after we have agreed to a session, display the Utilities/Settings screen and click the Miscellaneous tab. Click the red "Remote Assistance" button. Enter the Meeting ID number that we will provide in the text box and click the Remote Assistance button again to make the connection.
- If you have a Remote Assistance program that you would prefer that we use, just let us know.

GoToMeeting® is a registered trademark of the Citrix Corporation

21.2 Non-technical Support

Pathagoras Author & Staff For Sale!

I am for sale! Yes, me personally, and my staff, too.

We are available to help you set up Pathagoras, establish your Pathagoras network if you have multiple licenses, create or refine your books and libraries, etc.

- **Let us Pathagorize You:** If you simply do not have the time or the energy to Pathagorize your forms and create a system, we can handle it all for you. We have skilled and talented 'Pathagorizers' on staff who can quickly respond to your every request.
- Send one or more of your current systems. We will set up a Pathagorized model system in return. What you will get back is a complete, immediately usable, book. We will also send you instructions on how to place it onto a new or existing library shelf. Since the returned material will be a collection of standard Word documents, you will be able to augment or freshen the text on your own as needed.

Our rates are posted on the website, or call us for a quote. Your savings in future document assembly time will more than recapture the investment.

(If you want to take us up on this offer, we suggest that you send us just one of your systems, not all of them. When you get it back, study what we did and how we did it. Hopefully, then, you will feel so confident in how simple and easy it really is that you will attempt to 'Pathagorize' the next system yourself.)

- **Private Lessons:** If you choose not to read the manuals (I hate reading manuals too), you can simply 'buy' me to provide extended lessons and some 'on-site' (via GoToMeeting) guidance on how to get set up and fully operational with Pathagoras. Put your whole office staff in front of a computer and a speaker phone and we will be all set. And when you consider how far I can get in that time, that can be quite a deal for you.
- **"Will you travel?"** Heck yes! I would love to! (Get me out of my law office, please!) Of course, the airplane ticket and accommodations would be on you. My on-site charges are a bit higher, but the work that I could accomplish in that day or two (setup and training) should make the investment very worthwhile.

Index

- ! -

!GroupNames!
Explicit vs. Implicit 292

- " -

"/" vs. "/OR" 176
"?" 25

- (-

(exp)
Expiry date 81

- * -

Alias DropDowns 126
Aliases
DropDown Lists 160
Set command 268

- / -

/NEGOPT 196, 208

- { -

{Optional}
Compare to <<Optional>> 205

- < -

<\$>Currency Function (IDB) 97
<%> Percent Function (IDB) 99
<<*Ask*. . . prompts
Administrative text 254
Elements 254
<<*Ask. . .
<<*Break*>> 274
<<*Get*>> command 271

<<*If*. . . prompt 260
<<*Process*>> command 274
<<*Set*
Math 270
<<*Set*. . . command 268
<<document name>> call 302
<<Double Angle Bracket Functions>> 307
<<Optional>>
Compare to {Optional} 205
<Alt-D> 46
<Alt-O> 194
<F>ormat Function (IDB) 100
<Fr>actions function (IDB) 101
<P>aragraph Function (IDB) 102
<S>pell-out Function (IDB) 96

- 7 -

7-day Plan 25

- A -

Actors and their Roles 87
Adding terms
DropDown Lists 144
Administrative Text 175
and Aliases 123
Options/Optional 198
Syntax 198
Advanced Array
Instant Database 53
Age Calculations 76
Age math 81
Alias
Default value 127
Prepared Lists 118
Alias *Lists*
and !Groups! 122
Creating 115
Definition 114
Embedding 125
Excel 119
Rules 115
Sharing 126
Alias Table
(Embedded) 125
Aliases

Aliases
 and Administrative text 123
 Sentence assembly 130
 All or Nothing -- not 19
 Alphabetize
 Instant Database 55
 Variables 55
 Alt-G
 DropDown List calls 148
 QuickLinks 320
 Alt-S
 QuickLinks 320
 Analyze range of values 257
 Anatomy
 AskTable 297
 Optional Text Blocks 218
 Options Text Blocks 218
 Repeat Blocks 228
 And
 Last one in list 39
 Arguments
 (and) 180
 (day) (Date) 84
 (daylong) (Date) 84
 (ext) (Date) 84
 (long) (Date) 84
 (mon) (Date) 84
 (month) (Date) 84
 (num) (Date) 84
 (num4) (Date) 84
 (or) 180
 AskOptions 256
 Connector (Options) 185
 Conneector 256
 Cumulative (Options) 182
 If (all, any, not) 266
 Merge (Repeat) 239
 Radio 179, 256
 Repeat (and;or) 229
 Repeat(return) 233
 Array
 Instant Database 53
 'Ask' prompts
 Automatic creation 277
 Examples 289
 General 252
 Repeat 236
 AskTable

Anatomy 297
 AskTables
 Hover-over text 260
 AskValueInRange prompt 257
 Auto-create
 Variables 85
 Automatic Creation of Variables 85
 AutoReScan
 Default Setting 57
 Highlight 57
 Temporary disable 57
 AutoScan 57

- B -

Beginner's Guide 25
 Beyond assembly 24
 Blank Variables 61
 Book
 Actually a 'pointer' 20
 Defined 3
 Bracketed variables 36
 Break 274

- C -

Calculate
 Age 76
 Date Difference 76
 Date Math 76
 Calculator 55
 Calendar 74
 Calls to <<files>> 307
 Caption
 Example 243
 Capturing Data
 Instant Database 46
 Cascading logic 263
 Case (Logic prompt) 284
 Checklist for Upgrading System 29
 Clause Selection Screen
 Repeat Function 240
 Clipboard
 DropDown Lists 152
 Clippy 25
 Colon indicates Title 202
 Comma usage 57

- Commands
 - <<*Optional* . . .>> 172
 - <<*Options* . . .>> 172
 - <<*Repeat* . . .>> 224
- Commas
 - IDB lists 65
 - Lists 65
- Comments
 - Options blocks 175, 176
- Comments, Show
 - Default 32
- Compare
 - DocManagement vs. DocAssembly 23
- Concatenation
 - In-line 94
 - Instant Database values 94
- Conditional Text 172
- Connector
 - Argument 256
- Connectors
 - IDB lists 65
 - Lists 65
- Contact Us 324
- Content Controls
 - Processing Order 190
- Conversion
 - Simple to Advanced 19
- Convert
 - {Simple} to <<*Optional*>> 209
- Create Options Assistant 194
- Ctrl-Alt-G 320
- Cumulative
 - Plain Text 182
 - Tables 182, 193
- Currency Function (IDB) 97
- Customer Service 324
- Formatting 84
- Date Math (Instant Database) 76
- Date Math Features 74
- Debugging
 - Balancing Markers 318
 - DropDown Lists 163
 - Editing 318
- Default Value
 - Aliases 127
 - Multiple Choice Variables (#) 37
 - QuickLinks 320
 - Titled variables 43
- Default Views
 - Comments 32
 - Field Codes 32
 - Highlighting 32
 - Spelling 32
- Definition of terms 3
- Delay Processing 190
- Delete/Retain variables 61
- Deleting
 - DropDown Lists 145
- Disassembly 312
- Document Assembly
 - Techniques 140
- Document Calls 302
 - Disabling 306
- Document Dis-assembly 312
 - Last Paragraph Issues 314
 - Manually 312
 - Using DropDown Lists 312
 - Using Libraries and Books 313
- Document Logic
 - <<*If* prompt 260
 - <<*Set* command 268
- Document Packages 303
 - Alternative (document calls) 307
- Don't Increment 225
- Don't save variable 63
- Double Angle Bracket Functions 307
- Double Duty
 - DropDown Lists 165
- Double-click
 - Variables 62
- DropDown *Aliases* 126
- DropDown Lists 140
 - *Aliases* 160
- Adding Content 144

- D -

- DAB (double angle brackets) 307
- Data 52
- Data entry 46
- Data Records
 - Creating 52
 - Location 52
 - Using 52
- Date
 - Extended 84

DropDown Lists 140
 Clause Selection Screen 154
 Clipboard 152
 Community 161
 Controls 152
 Creating 'Free Hand' 141
 Debugging 163
 Delete 154
 Deleting 145
 Insert item from list 148
 'New Doc'/'Insert' toggle 148
 Open Document for editing 152
 Other Settings and Actions 154
 Panel 152
 Preview 152
 Print Lists 154
 Re-assign 154
 Repointing 146
 Sorting Characters 148
 Special 161
 Sub-folders 154
 Toggle buttons 152
 Using 148
 Variables 157

- E -

Editing
 and Testing 197
 Embed Alias Table 125
 Endless Loop 188
 Equivalency function
 Multiple Actors or possibilities 87
 Single equivalency 85
 Error Messages 324
 Example
 Interview 295
 Will 295
 Examples
 Explicit vs. Implicit 292
 Whimsical 290
 Excel
 <<*Get*>> command 271
 as Instant Database source 168
 Exceptions
 Cumulative (Options) 237
 Position Rule (Repeat/Options) 237
 Exclude Variable

IDB 63
 Expiry date 81
 Explicit settings
 !GroupNames! 292
 Extended Date 84

- F -

File Locations
 Instant Database 57, 106
 MultiChoice *Lists* 106
 Fill-in the blank
 Robust 186
 Simple 188
 Final And or Or 39
 Forcing an answer
 AskOptions(radio) 256
 Fractions
 Spell out 101
 within options blocks 176
 Functions
 <\$> Currency 97
 <%> Percent 99
 <F> Format 100
 <Fr> Fraction 101
 <P> Paragraph 102
 <s> Spell-out 96
 Instant Database 74
 Listing 103

- G -

Get command 271
 GroupNames
 Simple Optional 202
 Groups
 and Alias *Lists* 122
 Options/Optional Text 214
 Pronouns 136
 Repeat Function 226
 Variables 136

- H -

Hard Value 263
 Headers
 Repeating tables with . . . 230

Headers and Footers,
 Scan for variables in 48

Help 25, 324

Highlight variables
 Automatic 57

Highlighting variables
 Manually 318

Highlighting, Show
 Default 32

Hopeless imbalance 188

Horizontal tables 232

Hover Over Text
 Optional blocks 175

Hover-over text
 AskTables 260
 Options 260
 Options blocks 176
 Options/Optional 179

Hunt Path 304

- | -

If

(all) 266
(any) 266
(not) 266
Arguments 266
Includes 263
Multiple Comparators 264
Not 263

If . . . Then 260

If/Then

Logic Editor 278

Image Assembly 163

Implicit settings
 !GroupNames! 292

Incrementing variables 85, 225
 Repeat function 225
 Suppress incrementing 225

Initial Views 32

In-line

Concatenation 85
Math 81

'Insert Name' toggle 148

Insert text
 DropDown Lists 148

Insert/New Doc toggle
 DropDown Lists 152

Installation 28

Instant Database 57, 61

<\$>Currency Function 97
<F>ormat Function 100
<P>aragraph Function 102
<S>pell-out Function 96
Advanced Array 53
Alphabetize Variables 55
AutoReScan 57
AutoScan 57
Calculator 55
Calendar 74
Concatenation 94
Date Features 74
Date Math 74, 76
Delete blank variables 61
Delete/Retain variables 61
Equivalency function 85
Excel, as source 168
Find and Replace Tool 68
Location of Records 106
Math 76
Math & Date Math (in-line) 81
Miscellaneous Features 61
Personalize 57
Pointing to Records 106
Power Tools 55
Print Lists 55
Repointing Records 106
Resize (wider/taller) 55
Saving records 106
Screen Shots 69
Settings & Tools 57
Sex Change Mask 68
Sort Variables 55
Titled variables 43
Tree Service 106
Whole Folder S&R 55

Instant Database Masks

Location 106

Instant Database Records

Location 106

Instant Database Screen

Basics 46
Elements 46
Tour 46
Utilities 46
Width 46

Instant Interview 306
 Instant Recall
 DropDown Lists 148
 Interview
 External answer source 283
 Interview Wizard
 Print 284
 Interviews
 Introduction 252
 Recalling values 282
 Saving answers 281
 Introduction 2
 IQ scores 257

- L -

Length of variables 57
 Library
 Defined 3
 List
 Aliases 114
 Data Records 52
 DropDown 140
 Multichoice (aliases) 114
 Variables 55
 Lists
 Connectors 65
 Final 'And' or 'Or' 39
 Separators 65
 Little Checkboxes
 Instant Database 53
 Locating IDB records 106
 Location
 Alias *Lists* 124
 Instant Database Records 106
 Pointing/Repointing 106
 Sharing 106
 Logic
 Advanced (Interviews) 252
 Assistant 278
 Cascading 263
 Case 284
 Editor 278
 Hard Values 263
 If/Then 278
 Math 264
 Multiple Comparators 264
 Logic expressions 260

Logic Math 260

- M -

Math
 <<*Set* 270
 Age 81
 Calculate Age 76
 Date difference 76
 Date Math 76
 In-line 81
 Interviews 264
 Math (In-line) 81
 Matter 52
 Matter Records
 Creating 52
 Definition 52
 Location 52
 Using 52
 Matter Records (IDB)
 Locating 106
 Sharing 106
 Tree Service 106
 Merge
 Repeat command 239
 Microsoft Assistant 25
 Modifying variable names 36
 MultiChoice *Lists*
 Creating 115
 Definition 114
 Multiple choice text blocks
 Robust options 176
 Robust, syntax 191
 Simple Options 202
 Multiple Choice Variables
 Default selection 37
 Multiple Comparators 260
 Multiple Comparators (logic) 263
 Multiple comparitors
 If 264

- N -

Navigation
 DropDown Lists 154
 QuickLinks 320
 Negative Optional 196

Negative Optional 196
 Robust 175
 Simple Optional 208
 NEGOPT 208
 Nesting
 Maximum 188
 Options/Optional Text 188
 Repeats 242
 Simple Options 202
 'New Doc'/'Insert' toggle
 DropDown Lists 148
 Override 148
 Noun-verb consistency 208
 Number
 Auto-create 85
 Number consistency 208
 Numbers
 Add hidden 299
 Spell out 96

- O -

One-Page
 'Optional' text 218
 'Options' text 218
 Variables 218
 Optional
 Processing Order 190
 Optional (simple)
 Convert to <<*Optional*>> 209
 Optional Text
 /NEGOPT 196
 Advanced 175
 Anatomy 218
 'Ask' prompt 252
 Negative Optional 175
 Robust vs. Simple 205
 Summary 218
 Optional Wizard 194
 Options
 Add hidden numbers 299
 Additional values 257
 Fed by Repeat value 237
 Processing Order 190
 Options (Simple)
 Enabling/Disabling 206
 Fill-in the blank 188
 Options Text

Anatomy 218
 Arguments, 'and' and 'or' 180
 'Ask' prompt 252
 Calling secondary documents 180
 Checkboxes 179
 Connectors 185
 Cumulative 182
 Fill-in the blank 186
 Prompts 176
 Questions 176
 Radio buttons 179
 Restrict to single choice 179
 Stop-points 186
 Summary 218
 Tables and Rows 193
 Options/Optional Text
 {Simple} 202
 Administrative Text 198
 Anatomy 191
 Creating 194
 General 172
 Groups 214
 Nesting 188
 Syntax 191
 Testing 197
 'Wizaard' 194
 Order
 Processing 276
 Order of Search 304

- P -

'Package' of Forms 5
 Pathagorizing
 Multiple Choice Variables 37
 Simple Variables 36
 'Pathagorizing' 29
 PDF Assembly 163
 Percent Function 99
 Personalize
 Defined 3
 Instant Database 57
 Plain Text
 Advantages 22
 Multiple Choice Variables 37
 Simple Variables 36
 Pleading captions 243
 Pointing

Pointing
 Concept (general) 20
 DropDown Lists 146
 Instant Database Records 106

Pointing to
 Alias *Lists* 124

Position 1 Book 304

Power Tools (IDB) 55

Prepared Lists 118

Preview
 DropDown Lists 152

Print Lists
 DropDown Lists 154
 Variables 55

Process
 All open documents 53
 Defined 3
 Options and Optional Text 197
 Processing vs. editing source text 16
 Processing vs. personalizing 16
 Testing 197
 'The' process 16

Processing Order 276
 Delay 190
 Sequence 190

Prompts
 Hover-over text 179
 Optional Text 175
 Options Text 176

Pronouns 136

- Q -

Quandries
 The final 'and' or 'or' 39

QuickLinks
 Creating 320
 Default 320
 Editing 320
 Insert text: Ctrl-Alt-G 320
 Saving 320
 Using (Alt-G) 320

- R -

Radio
 Argument 256

Radio buttons 179

Range
 vs. Position 257

Range Analysis 257

Repeat Function
 (merge) 239
 Alternatives 241
 and Options 237
 Arguments ('and'/'or') 229
 'Ask' 236
 Clause Selection Screen 240
 Cumulative exception 237
 General 224
 Groups 226
 Incrementing variables 225
 Lists 229
 Nesting 242
 Repeat(merge) 239
 Restriction 242
 Return repeat value 233
 Rows in Tables 230
 Settings 235
 Simple 224
 Tables 230
 Testing 242

Repeats
 Horizontal (tables) 232
 Incrementing variables 232
 Repeating cell 232

Repointing
 Alias *Lists* 124
 DropDown Lists 146
 Instant Database Records 106
 'Pointer' concept 20

Resize
 IDB Screen 55

Reversing documents 274

'Running' Pathagoras 28

- S -

Samples
 Aliases and MultiChoice Lists 114
 Ask Table 297
 <<*Ask*>> prompts 254
 AskRepeat with Options 237
 AskValueInRange 257
 Concatenation 94

- Samples
 - Equivalency function 85
 - Equivalency function and *Actors* 87
 - Instant Database 76
 - Interview 295
 - Math and Date Math 76
 - Operators 76
 - Options, simple/complex mix 207
 - Options/Optional Text 191
 - Pleading captions 243
 - Reciprocal Will 295
 - Repeat 243
 - Repeat with Options 237
 - Repeat, arguments 229
 - Repeat, Groups 226
 - Repeat, return value 233
 - Repeat, Tables and Rows 230
 - Repeats, Incrementing variables 225
 - Simple 76
 - Simple 'Repeat' block 224
 - Will 295
- Scan for Variables 48
- Screen Shots
 - Instant Database 69
- Search Order Rules 304
- Sentence Assembly 130
- Separators
 - / (negative optional--robust) 175
 - / (negative optional--simple) 208
 - /ANDOR (simple options) 202
 - /OR 179
 - /OR (simple options) 202
 - Lists 65
 - Multiple Choice 179
- Series
 - IDB 65
- Set command 268
- Settings 20
 - Default Views 32
 - 'Repeat' Function 235
- Sex Change Mask
 - IDB 68
- Sharing Alias *Lists* 126
- Sharing, Share
 - IDB Masks 106
 - IDB Records 106
- Simple Options
 - General 202
 - Groupnames 202
 - Nesting 202
 - Titles 202
- Simple Options Blocks
 - Convert to <<*Optional*>> 209
- Simple to Advanced 19
- SlashOR (/OR) 176
- Sort
 - IDB screen 55
 - Instant Database variables 55
- Sorting Characters
 - DropDown Lists 148
- Spell out
 - Currency 97
 - Fractions 101
 - Numbers 96
- Spelling, Show
 - Default 32
- Stop
 - phrase 186
 - points 186
- Structure
 - Options/Optional Text 191
- Structure Checker
 - Options and Optional Blocks 197
- Style
 - Captions (pleadings) 243
- Style/Caption (legal kind)
 - Example 243
- Summary
 - Optional Text Setup 218
 - Variable Setup 218
- Support
 - Non-Technical 326
 - Technical 324
- Suppress processing
 - DropDown Lists 152
- Syntax
 - Administrative text (options blocks) 198
 - Options/Optional Text 191
- T -
- Tables 193
 - Cumulative 182, 193
 - Options 193
 - Repeat Rows 230
- Technical Support 324

Temporary variable 63
 Testing
 Options and Optional Text 197
 Textboxes
 Scan for variables in 48
 Tip text 260
 Tip-text 176
 Titles
 Simple Options 202
 Variables 43
 Tree Service
 Instant Database Records 106

- U -

Unused Variables
 Delete 57
 Upgrade Checklist 29
 Upgrading Pathagoras
 Standard 25

- V -

Value
 Return repeat value 233
 Values
 Analyze ranges 257
 Variables 48, 55
 Alphabetize 318
 Auto-count the brackets 318
 Auto-create 85
 Color the brackets 318
 Completing 51
 Debug 318
 Definition 36
 Delete if blank 61
 Don't save 63
 Groups 136
 Headers and Footers 48
 Highlight 55, 318
 Incrementing w/ repeats 225
 Length 57
 Multiple Choice 37
 Print Lists 55
 Pronouns 136
 Scan for, 48
 Simple Variables 36

Temporary 63
 Testing 197
 Textboxes 48
 Titled variables 43
 Unused 57
 Variables List
 DropDown 157

- W -

Whole Folder S&R
 IDB 55
 Widen Screen
 IDB 55
 Wizards
 Optional 194
 Options 194
 WordPerfect®
 Assembly 163
 DropDown Lists 163

Pathagoras prides itself on providing prompt, useful and personal customer service. While we hope that this Manual and the other instructional materials are helpful, you can still count on the 'personal touch' of Pathagoras' customer service as a 'first line' of help as well.

We truly enjoy hearing from our customers and potential customers. While we cannot promise that you will never receive a voice-messaging service if you call us, more likely you will receive a live person at the other end. If you do get our voice mail, just leave a message. We will call you back promptly.

Contact information is spread across as many places as we could find so that you do not have to hunt for an email address or telephone number, and we repeat it here. Let us know if we can ever be of service.

Pathagoras
Innovative Software Products of VA, LLC
Roy Lasris, President
www.pathagoras.com
info@pathagoras.com
tel: 866-PATHAGOras (1-866-728-4246) (tollfree)
tel2:+1 (757) 877-2244 (USA)
117 Chisman Landing
Seaford, VA 23696 USA