



# IndexMatic 3.x EXPERT

Independent Index Builder  
& Full-Text Search Engine  
for InDesign CC / CS6 / CS5 / CS4

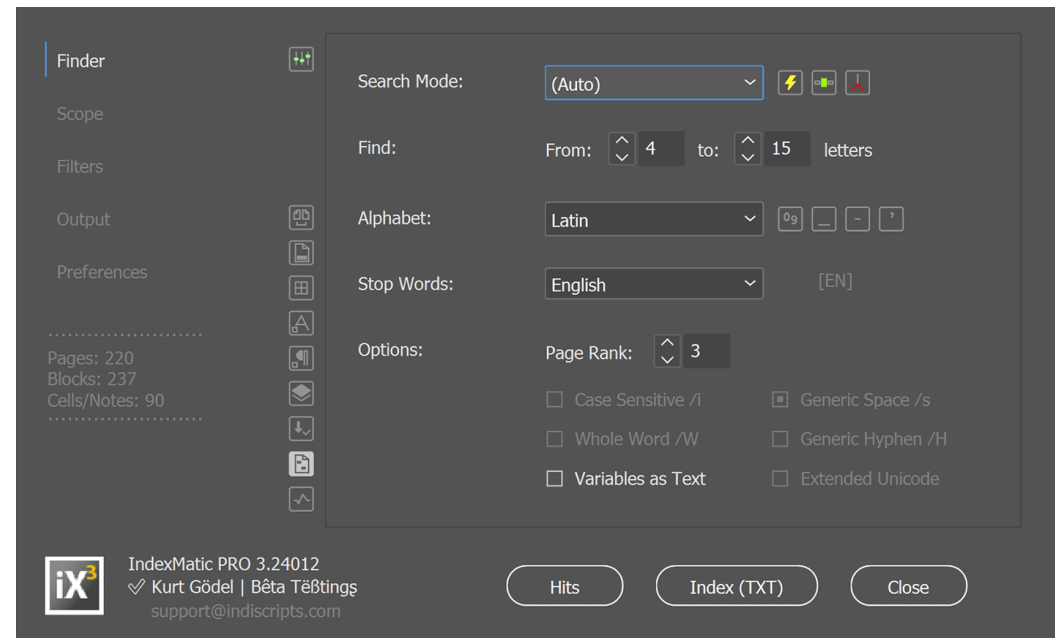
---

USER MANUAL



## 1. Description

IndexMatic<sup>3</sup> is a full-featured index builder for Adobe InDesign®. It comes with advanced parameters and a unique query system that efficiently scans and reports text data from finalized documents. This program has several usages: automatic indexing of an entire book based on frequent tokens, fine-tuned index creation based on word lists and/or regular expressions, full extraction of terms having determined styles applied, quick scan or statistical “Hits” report focusing on particular layout areas: page ranges, footnotes, tables, etc. IndexMatic<sup>3</sup> is the *expert* version of IndexMatic<sup>2</sup>; it offers about fifty new features and extends its linguistic field to more than twenty different alphabets, which enables it to address hundreds of languages.



## 2. System Requirements

- Mac OS X / macOS (10.6 or later),  
or Windows 7 / 8 / 10 / 11.
- CPU with a minimum clock rate of 3 GHz.
- Main memory (RAM) of at least 4GB.
- 800×600 pixel screen-resolution or greater.
- Adobe InDesign (CC/CS6/CS5/CS4).



IndexMatic<sup>3</sup> supports all InDesign versions from CC (2024) to CS4, Mac/Win.

## 3. TRY vs. PRO Version

You can download a free tryout version of IndexMatic<sup>3</sup> from: <https://indiscripts.com/blog/public/scripts/IndexMatic3Try.zip>. It offers all the features of the PRO release, but it will limit the output to 50 index entries.

**NOTE** We strongly encourage you to install and test the TRY version before you purchase the PRO license of the product. Always make sure that your system meets the requirements.

The dialog window of IndexMatic<sup>3</sup> has a very similar look-and-feel in both Mac OS and Windows environments. The following languages are available (depending on your InDesign locale):

- ▶ English (*default*)
- ▶ French
- ▶ German
- ▶ Spanish
- ▶ Italian
- ▶ Russian

# Getting started



## 1. Before you install

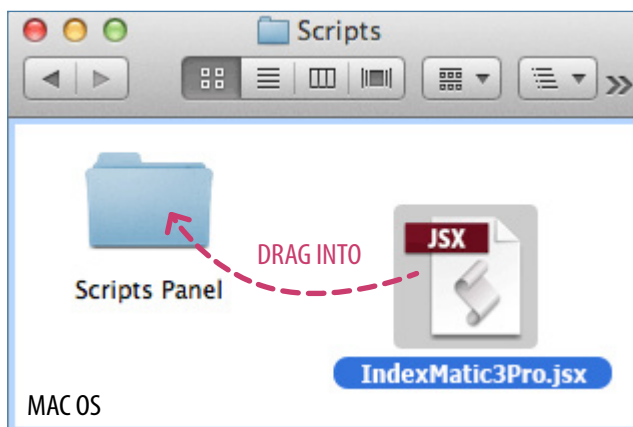
IndexMatic<sup>3</sup> resides in a single file: **IndexMatic3Pro.jsx**. When you download the file from your private link, however, it is originally zipped.

The first step is to unzip the .ZIP file so you can place **IndexMatic3Pro.jsx** at the desired location (see below).

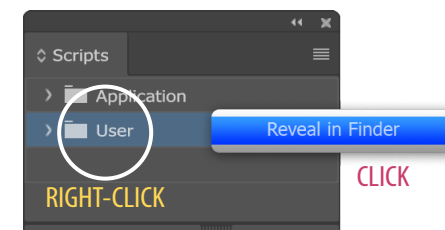
As a precaution before you go on, save your working files and restart InDesign in a clean session.

## 2. Installing in Mac OS

- 1) In InDesign, open the Scripts panel as follows:
  - Window ► Utilities ► Scripts (CC, CS6, CS5), *or*
  - Window ► Automation ► Scripts (CS4).

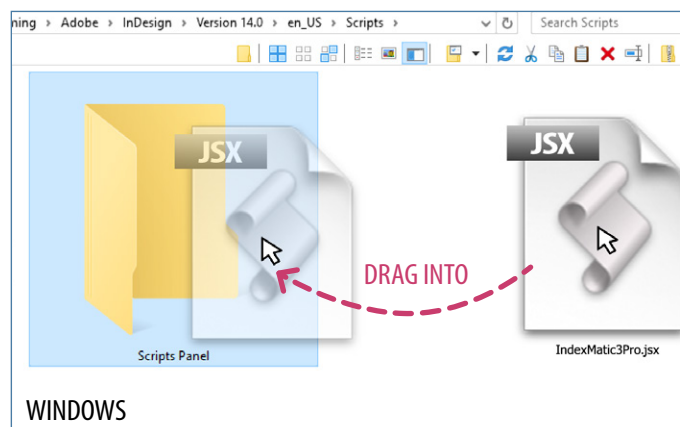
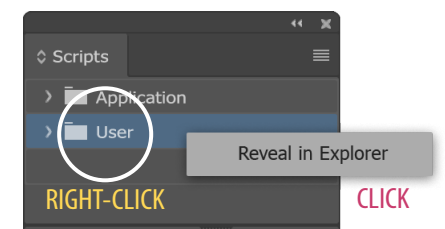


- 2) You see there two main folders: Application and User. Right-click the User folder and pick “Reveal in Finder”.
- 3) You should now see a Scripts Panel folder. Drag **IndexMatic3Pro.jsx** into there. Congratulations, IndexMatic<sup>3</sup> is now installed!



## 3. Installing in Windows

- 1) In InDesign, open the Scripts panel as follows:
  - Window ► Utilities ► Scripts (CC, CS6, CS5), *or*
  - Window ► Automation ► Scripts (CS4).
- 2) You see there two main folders: Application and User. Right-click the User folder and pick “Reveal in Explorer”.



# Getting started



- 3) You should now see a Scripts Panel folder. Drag **IndexMatic3Pro.jsx** into there. Congratulations, IndexMatic<sup>3</sup> is now installed!

## 4. Installing an update

If you are notified that an update of the product is available, simply download the new package from your private link, then unzip and install the file **IndexMatic3Pro.jsx** over the previous one, i.e. at the same location.

The new version is instantly functional, and your global settings and preferences are all preserved.

## 5. Running IndexMatic<sup>3</sup> from the Scripts panel

Once the installation is done, switch back to InDesign. You can run IndexMatic<sup>3</sup> from the Scripts panel as follows:

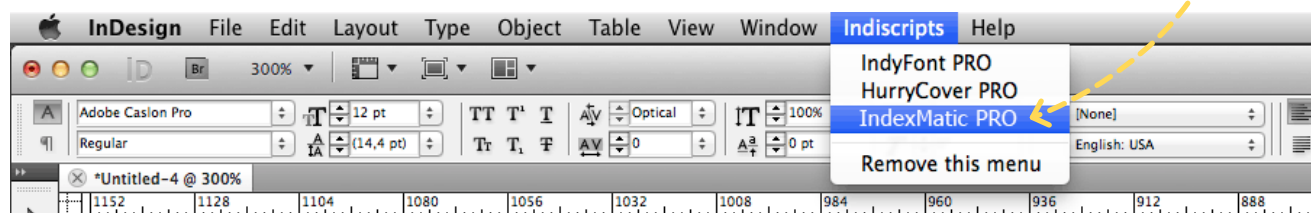
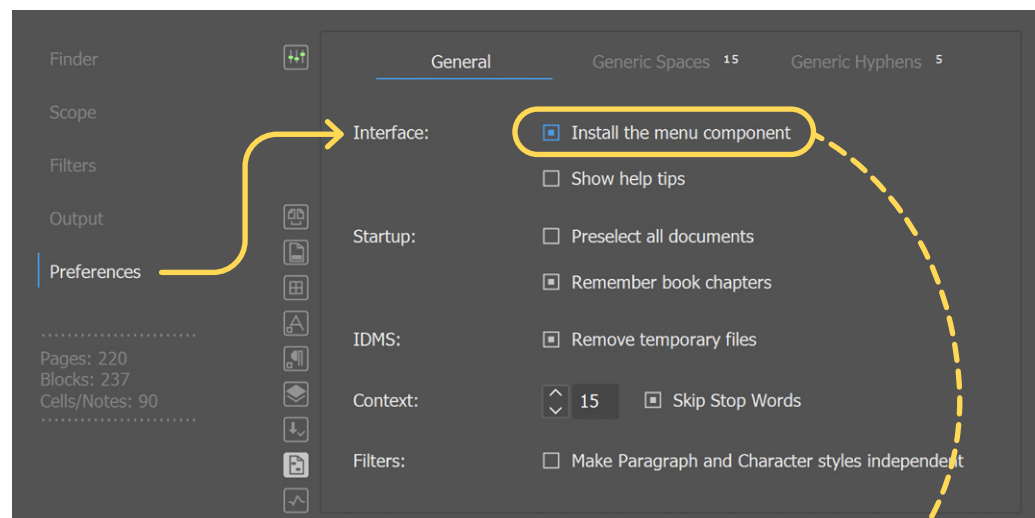
- 1) In InDesign, display the Scripts panel via:
  - Window ► Utilities ► Scripts (CC, CS6, CS5), or
  - Window ► Automation ► Scripts (CS4).
- 2) Look for **IndexMatic3Pro.jsx** in the User folder, then double-click on it.

## 6. Running IndexMatic<sup>3</sup> from the Indiscripts menu

To have IndexMatic<sup>3</sup> available in InDesign's menu bar, run the script once (as previously detailed), go to **Preferences** and turn on the option *Install the menu component* (see the screenshot below). Then validate the dialog.

You can now run IndexMatic<sup>3</sup> going into:  
Indiscripts ► IndexMatic PRO

Tired of continually digging into the Scripts panel? Make IndexMatic<sup>3</sup> available in a dedicated menu!





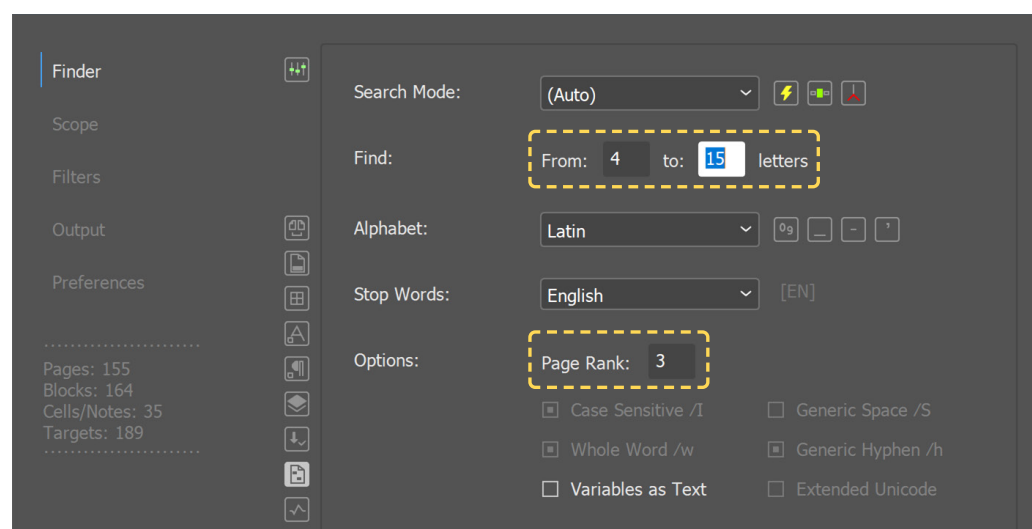
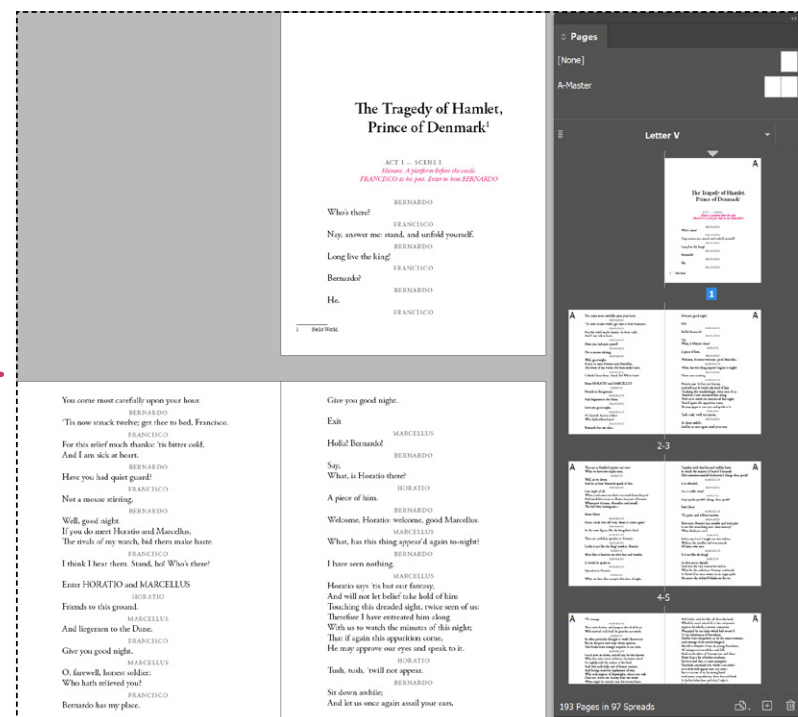
## 7. Create your first “Automatic Index”

Having installed the script, you are ready to experiment its *automatic* mode and generate a raw index. By this, we mean a lazy word report: no enhancement, no human intelligence! IndexMatic<sup>3</sup> will just grab and output any sequence of characters that looks like a simple word.

- 1) Open an InDesign document (with some text content!) If other documents are available, the script will invite you to select your target document(s) before entering the main dialog.
- 2) Run IndexMatic<sup>3</sup>.
- 3) From the **Finder** panel (first item of the side menu),
  - Go into the **Search Mode** list and select *(Auto)*;
  - In **Find**, set **From**: 4 and **to**: 15 letters (so you eliminate short and long words);
  - Check that **Alphabet** and **Stop Words** fit the writing system and language of your document, e.g. *Latin* and *English*.

**NOTE** Do not worry if a warning symbol ⚠ is shown to the right of the **Stop Words** list. We will fix that soon.

- 4) Now—still in the **Finder**—let’s adjust **Options** ► **Page Rank**. This number from 1 to 9 tells IndexMatic<sup>3</sup> to include words that appear at least *n* times on a given page (*n* being the Page Rank). 2 or 3 are the recommended values when processing short documents in *Automatic* mode. Use a higher value if you want.



# Getting started



**TIP** To increase/reduce the **Page Rank**, you can use the up/down arrow keys. This shortcut works in any numeric field.

5) Activate the **Output** panel and go first to the **Sorting** subpanel (horizontal menu). In the **Language** list, select the desired option if not already fine. It determines how index entries are to be alphabetized. This should solve the conflict we have mentioned above about the **Finder** ► **Stop Words** option.

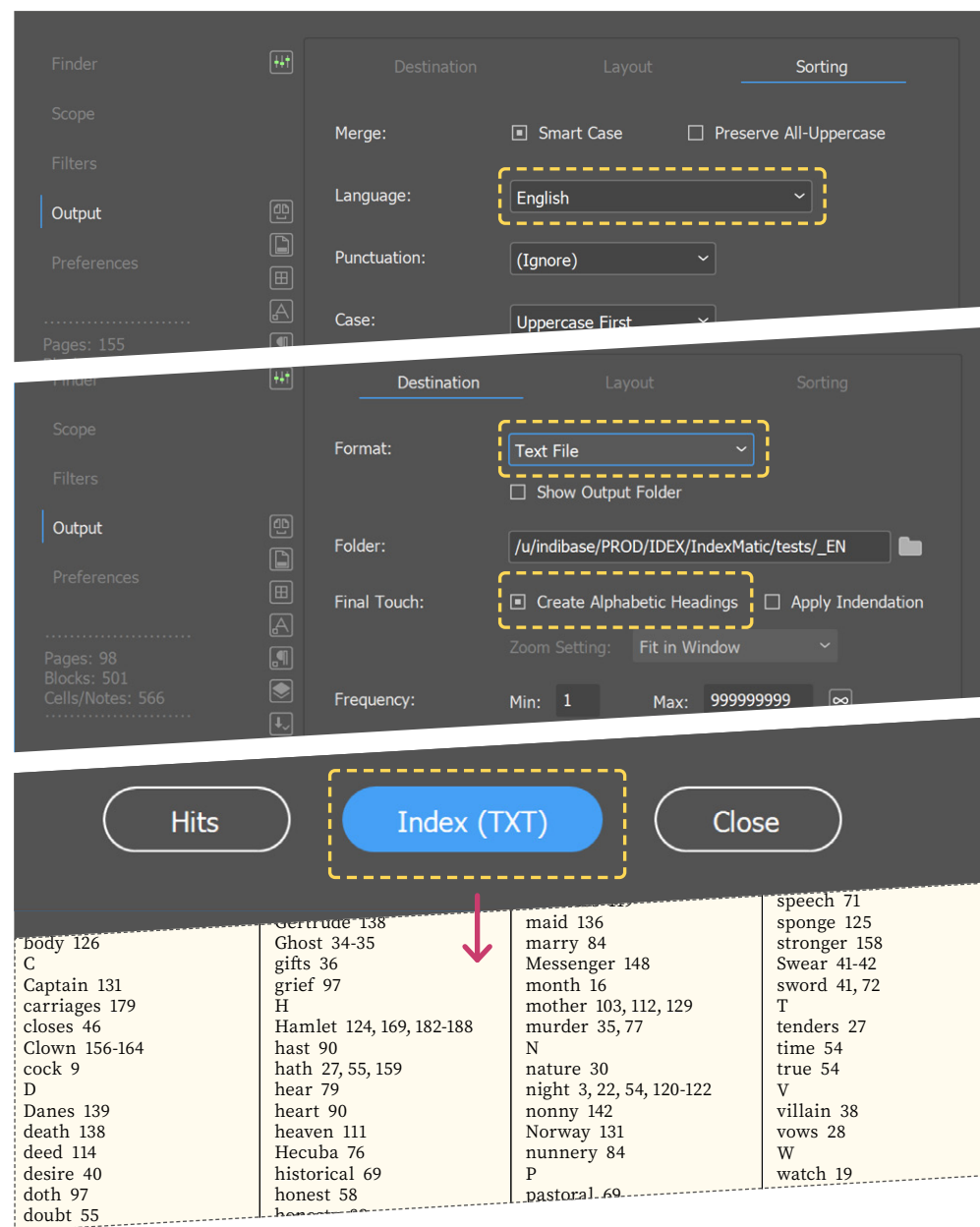
**NOTE** Most languages have their own ordering rules. If your language is not listed here, or if multiple languages are involved in your document, select a more general system like *European Ordering Rules*, *DUCET* or *UTF16*. (Those appear at the top of the list.)

6) Go back to the **Destination** subpanel (horizontal menu) and select the following options:

- **Format:** *Text File*.
- **Show Output Folder:** off.
- **Folder:** (select or enter the output folder path).
- **Create Alphabetic Headings:** on.

7) Click the **Index (TXT)** button at the bottom of the dialog. The resulting file will show up in your default text editor.

Thanks to the Page Rank filter, a limited number of entries should appear in the list, each associated to a decent number of page numbers. This is obviously *not* a fine-tuned index, but the automatic mode makes it possible to identify crucial concepts very quickly.



# Welcome to IndexMatic<sup>3</sup>



So, what can IndexMatic<sup>3</sup> do for you? Book indexing is a serious task—and a skilled job!—that should in no way be downscaled to automatic indexing. A finding-and-matching program cannot extract concepts that are only *implied* in the text. Without human intelligence—or AI!—it cannot detect all major topics wrapped in the document. Yet human and automatic indexers share a common subgoal: collecting text data, document locators, in order to make information retrievable. IndexMatic<sup>3</sup> has been designed with these subtasks in mind. It provides tools—searching, filtering, counting, sorting, formatting—that help reduce human intervention to what cannot be automated: *understanding*.

## 1. The Big Picture

Let's start with a few *simple* problems:

- You want to extract the entire vocabulary of a document—in order to make statistics, detect spelling issues, build a lexicon, etc.
- Your client (publisher, author) sent to you a list of proper names for which you must identify all occurrences in the book.
- You are about to complete the layout of a huge catalog using special markers, layers, styles and/or number patterns, and you now need to index all product references.

- Your recipe book has ingredients in different categories (liquids, spices, fruits...); you want to index them with respect to topics, subtopics—or even more sublevels.

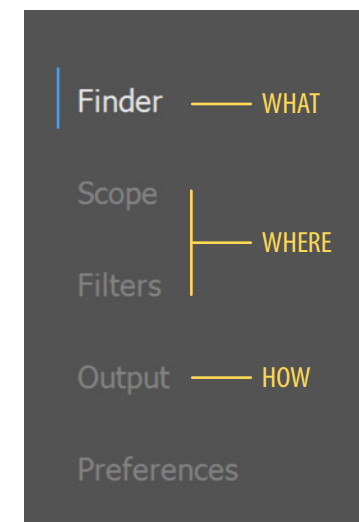
IndexMatic<sup>3</sup> has a straightforward approach to address all this:

- 1) *What are we looking for?* That's the purpose of the **Finder** panel, a very granular query engine explored in the next chapters.
- 2) *Where should we look?* In other words, which pages, areas, styles, layers, etc., are to be inspected? That's the purpose of the **Scope** and **Filters** panels.
- 3) *How should the results come out?* That's the purpose of the **Output** panel, which is about exporting, formatting and sorting index entries.

**TIP** Familiarize yourself with the WHAT / WHERE / HOW trichotomy, it will help you navigate intuitively the interface.

Ultimately, IndexMatic<sup>3</sup> just deals with InDesign's texts and page numbers!

In our first problem (*vocabulary extraction*) the program doesn't even have to handle page numbers. It must rather focus on text features like frequency, occurrences, character ranges. The WHAT parameter is somehow unspecified, as all depends on which



# Welcome to IndexMatic<sup>3</sup>



particular text pattern is to be considered a *part of speech*. That's why the query engine allows you to enter regular expressions like `/\w{5,}/2` or `/\m\w+(-\w+)?/`. (Do not panic, this manual will clarify all that!)

In the *proper names* and *recipe book* problems, the search terms are determined in advance (or could be determined from a *vocabulary extraction* stage). You can then submit to IndexMatic<sup>3</sup> a comprehensive word list, or more precisely a QUERY LIST. A query can specify rules regarding case-sensitivity, page rank, etc., and special commands. In the *recipe book* problem, we need to group ingredients by topics: this is done at the query level too.

In the *catalog* problem, various approaches are possible. They either involve refining the *WHAT* or narrowing the *WHERE*. If your product references share a normalized text pattern or rely on a hidden marker, a single regular expression will capture them. If a dedicated character style (*MyProdStyle*) rules them all the procedure is much simpler: enter the query `./+/` in the **Finder** and tell IndexMatic<sup>3</sup> to scan only *MyProdStyle* data (from the subpanel **Filters** ▶ **Character Styles**).

Finally, if your indexable text (products, recipes, etc.) resides within a particular page range, or only in the tables of your document, go to the **Scope** panel and define your workspace.

**NOTE** Conceptually, **Scope** and **Filters** options serve the same general purpose of selecting document subparts that the engine will investigate. Styles and layers are addressed from the **Filters** panel.

## 2. Inside the User Interface

### ▶ Side Menu

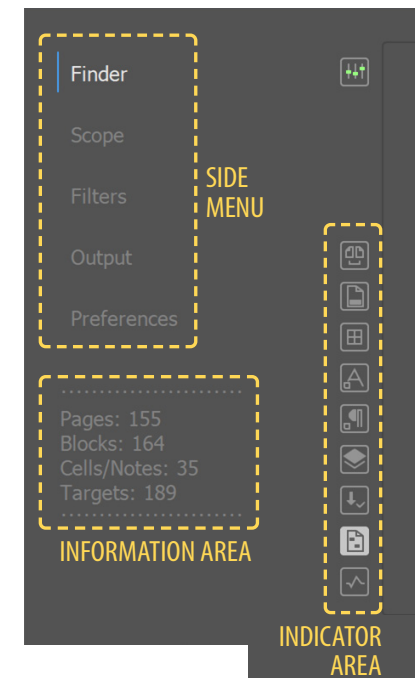
The vertical **SIDE MENU** (to the left) allows you to navigate through the main panels: **Finder**, **Scope**, **Filters**, **Output**, **Preferences**. Just click a menu item to activate the corresponding panel.

### ▶ Information Area

Just below the side menu is the **INFORMATION AREA**. It appears a few seconds after the dialog shows, and displays some basic counters like the number of pages and text containers (**Blocks**) found in the target document(s). If available, **Cells/Notes** indicates the total number of table cells, footnotes and endnotes that IndexMatic<sup>3</sup> has identified so far. Information may be updated gradually as the program collects data in the background. If style-driven **Filters** are active, you will also see **Ranges** and **Targets** counters. **Ranges** reports the overall number of *text style ranges* found in the document(s); **Targets** tells how many text units meet the current filters.

### ▶ Indicator Area

Indicators are simple two-state (on, off) icons that let you visualize settings at a glance. The **INDICATOR AREA** brings together important *flags* having significant effects on index processing. The table below summarizes the meaning of each indicator (when turned on). Don't worry if some features are still unclear to you. Since every indicator reflects the state of an option, you



# Welcome to IndexMatic<sup>3</sup>



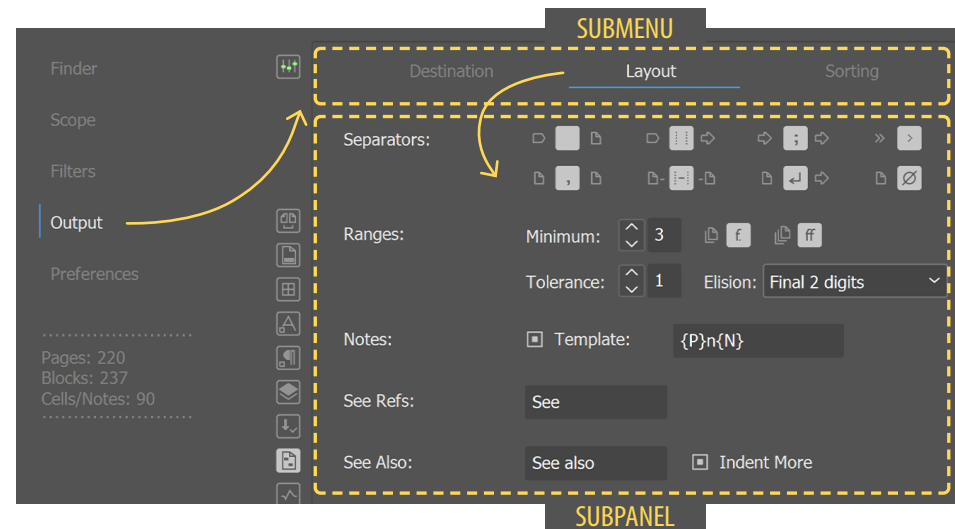
can go into the corresponding panel (FROM column) to check or change the corresponding parameter. Indicators simply remind you that this or that option presently has a custom, non-default setting.

INDICATOR	MEANING	FROM
<b>Page Range</b>	Some page range (e.g. 120-150) is selected. Other pages won't be indexed.	<a href="#">Scope</a>
<b>Footnotes/Endnotes Only</b>	Search restricted to footnotes and/or endnotes. Main text containers won't be inspected.	<a href="#">Scope</a>
<b>Tables Only</b>	Search restricted to table cells. Main text containers won't be inspected.	<a href="#">Scope</a>
<b>Character Style Filter</b>	Targeting specific character style(s). Outer texts will be ignored.	<a href="#">Filters</a>
<b>Paragraph Style Filter</b>	Targeting specific paragraph style(s). Outer texts will be ignored.	<a href="#">Filters</a>
<b>Layer Filter</b>	Targeting specific layer(s). Texts on other layers will be ignored.	<a href="#">Filters</a>
<b>Original List Order</b>	Original Query List order is preserved. This inhibits other sort options.	<a href="#">Finder (Query List)</a>
<b>Page Rank</b>	Page Rank (> 1) globally set. Matches under the threshold won't appear.	<a href="#">Finder</a>
<b>Min/Max</b>	Minimum and/or maximum conditions are specified regarding either frequency or shared pages.	<a href="#">Output</a>

**TIP** Click on any active indicator to get the related control automatically selected.

► *Submenus and subpanels*

Some panels ([Filters](#), [Output](#), [Preferences](#)) provide an internal submenu arranged horizontally at the top of the panel. It allows you to switch between sections referred to as SUBPANELS. For



example, the [Output](#) panel has three subpanels: [Destination](#), [Layout](#), and [Sorting](#). Click the submenu item to activate the corresponding subpanel.

PANEL	SUBPANELS
<a href="#">Finder</a>	
<a href="#">Scope</a>	
<a href="#">Filters</a>	<a href="#">Character Styles</a>   <a href="#">Paragraph Styles</a>   <a href="#">Layers</a>
<a href="#">Output</a>	<a href="#">Destination</a>   <a href="#">Layout</a>   <a href="#">Sorting</a>
<a href="#">Preferences</a>	<a href="#">General</a>   <a href="#">Generic Spaces</a>   <a href="#">Generic Hyphens</a>


**NOTE** Although without a subpanel, the [Finder](#) provides slightly different “views” depending on the active Search Mode.

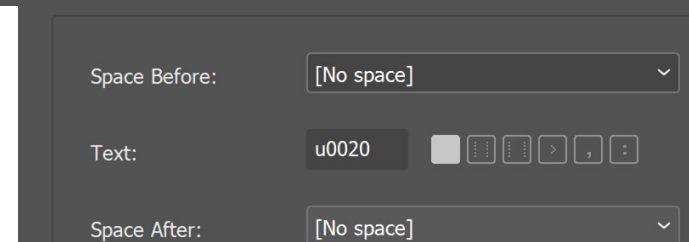
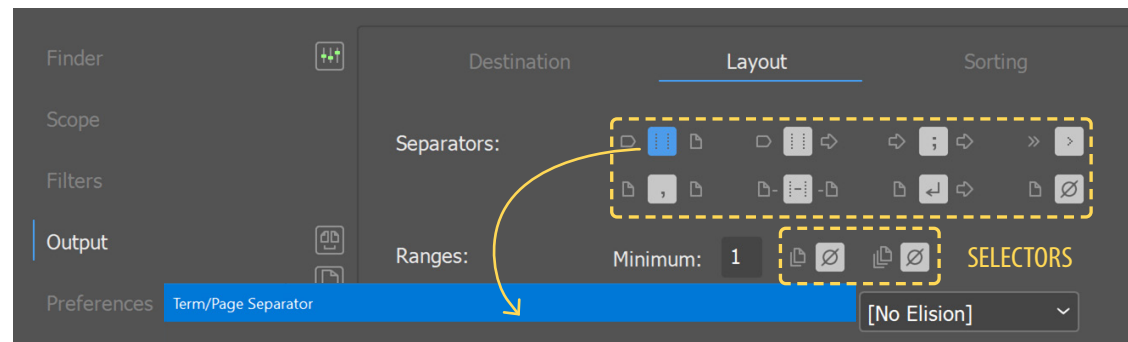
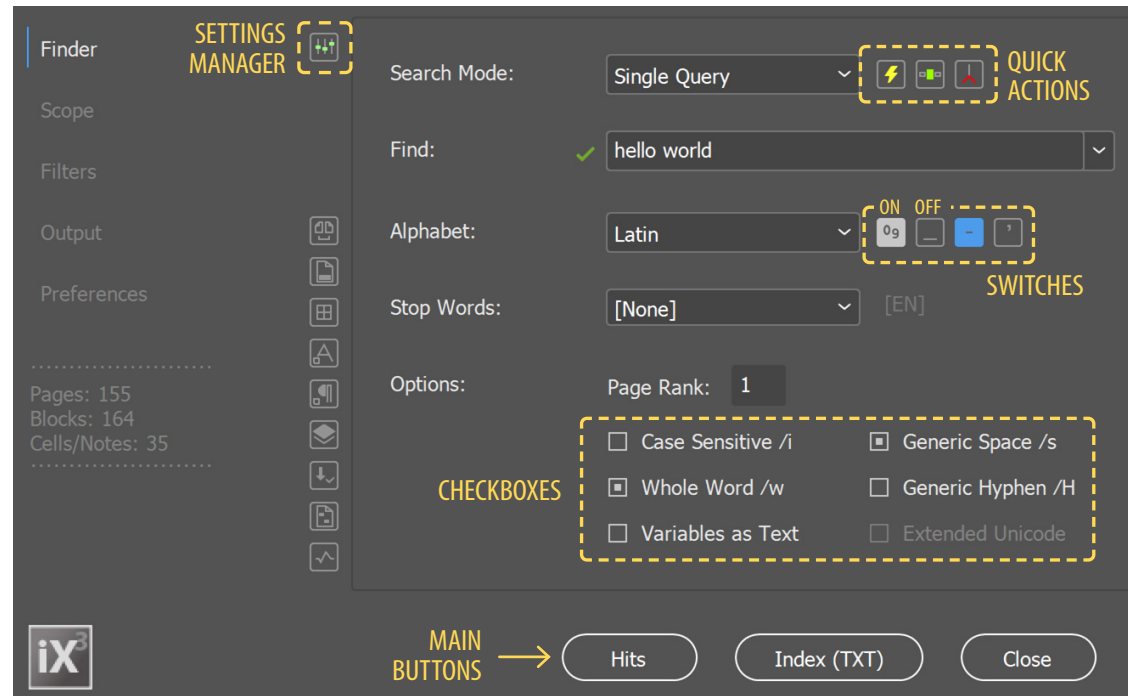


# Welcome to IndexMatic<sup>3</sup>



## ► Buttons

- 1) Always visible, the MAIN BUTTONS (**Hits**, **Index...**, **Close**) are grouped at the bottom of the dialog box. They command the essential tasks.
- 2) Colored ICON-BUTTONS, smaller, can run quick actions or more specific tasks. The **Settings Manager** button  (above the set of indicators) lets you export or import your settings.
- 3) SWITCHES are iconic checkboxes. They have the same appearance as indicators but you can turn them on or off at will by simple click. When a switch is selected, you can also toggle its value by pressing the spacebar.
- 4) CHECKBOXES are basic on/off commands with a descriptive label. The **Finder** panel has a group of checkboxes that specify general find options like *Whole Word*, *Variables as Text*, etc. Sometimes a checkbox is greyed out, meaning that the command is presently unavailable or controlled by another parameter.
- 5) Finally, the **Output ► Layout** subpanel hosts a special kind of button known as SELECTOR. Selectors allows you to edit and fine-tune separators or small sequences of punctuation marks. When you click a selector, a dedicated dialog shows up (see the below figure).





# Welcome to IndexMatic<sup>3</sup>



## ► Lists

- 1) DROP-DOWN LISTS allow you to choose one value from a pre-defined set of options. The top figure shows the **Output ► Destination ► Format** drop-down list in its pull-down state—the item *Text File* being selected.

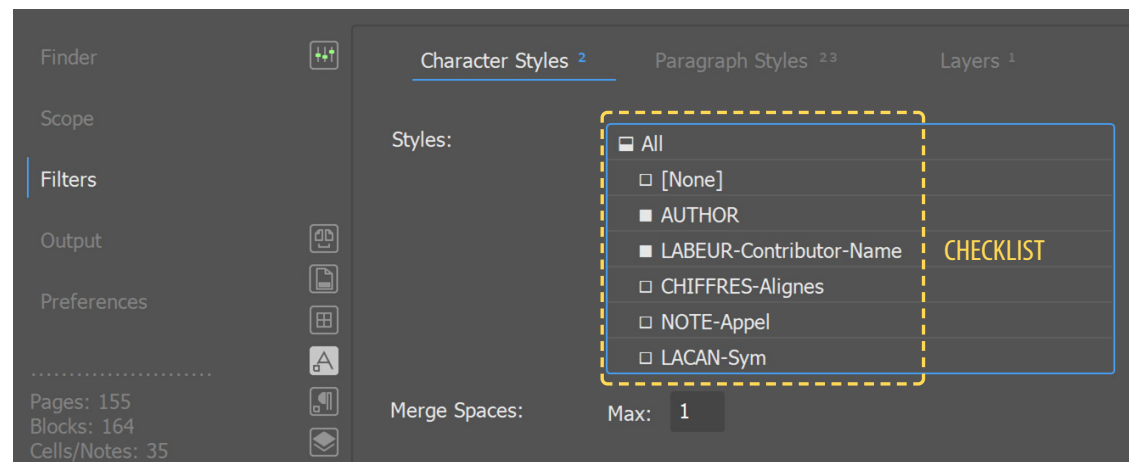
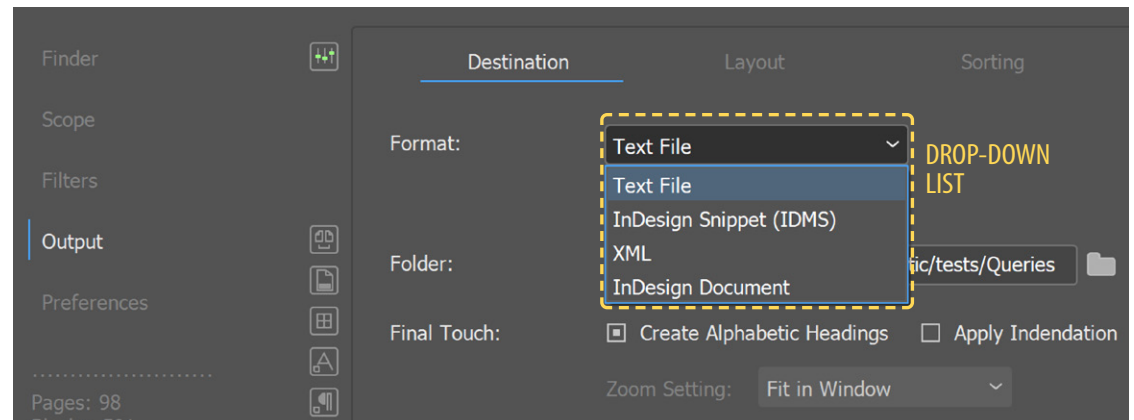
**TIP** To quickly navigate through a long drop-down list, type the initial letter of the desired entry on the keyboard. Re-type the same letter to cycle between matches.

- 2) CHECKLISTS represent options that you selectively include or exclude by clicking items. They are arranged hierarchically (as the branches of a tree) so you can (de)select either a particular node or an entire branch. Checklists are convenient for choosing elements organized in groups, subgroups, etc.


**TIP** When a branch is partially selected, its checkbox becomes a half-filled square . Click it to select all descendants. Checklists also support the shortcuts Ctrl A (*selects all items*) and Ctrl Alt A (*deselects all items*).

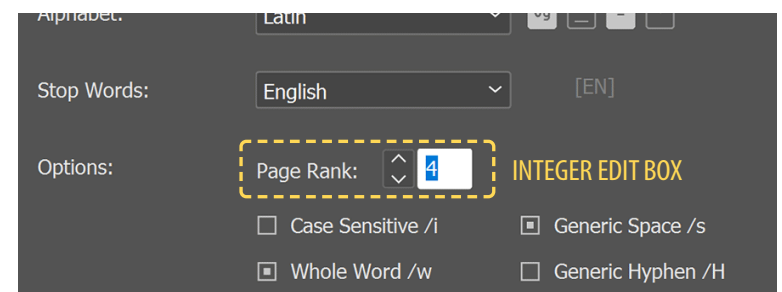
## ► Edit Boxes

Most IndexMatic<sup>3</sup> text fields are simple EDIT BOXES where you can enter strings or numbers. Some have a limited length or a specific range of values. A good example is the **Page Rank** box, which only supports integers from 1 to 9. (Invalid inputs are flagged with a pop-up message.)



from version 3.23111

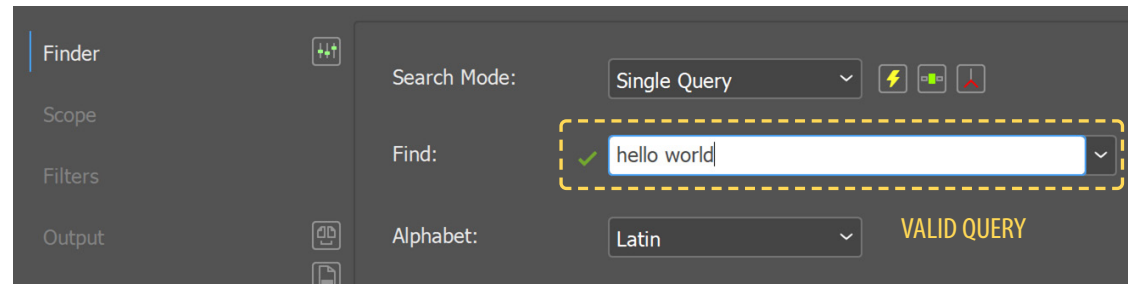
All integer edit boxes are accompanied by a “stepper” button  allowing you to increase or reduce the value.



# Welcome to IndexMatic<sup>3</sup>



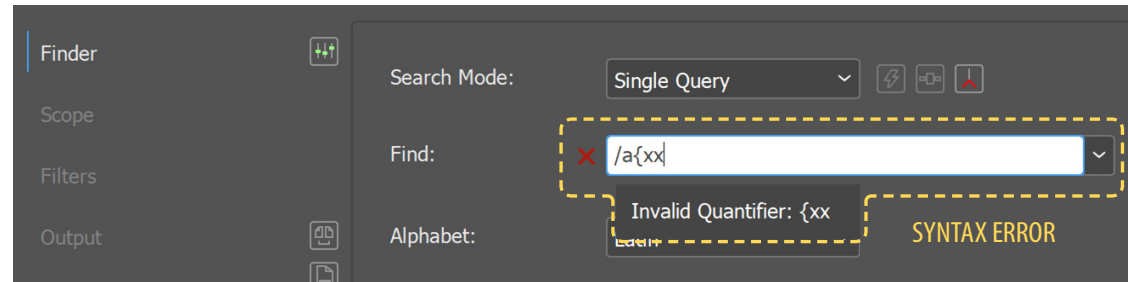
The **Single Query** editor (*see screenshot*) is a bit more sophisticated. As you are typing, IndexMatic<sup>3</sup> parses the syntax of your query and displays a green checkmark ✓ if all is fine. If an error is detected, a red cross ✗ and a warning message inform you that the input cannot be validated.



**NOTE** ⚠ Whenever you exit an edit box that contains invalid data, the previous (valid) value is restored.

### 3. Migration from IndexMatic<sup>2</sup> to IndexMatic<sup>3</sup>

Although IndexMatic<sup>3</sup> provides completely new features, IndexMatic<sup>2</sup> users can leverage their experience to get started with the expert version. Here are the most noticeable changes:



FIELD	MAIN CHANGES
Scope	<ul style="list-style-type: none"> <li>↻ Precise selection among all available documents and/or book chapters.</li> <li>↻ Advanced Page Range selector.</li> <li>+ Detects duplicated page names and warns on collisions.</li> <li>↻ Precise selection of layers (now managed in the <b>Filters</b> panel).</li> </ul>
Context	<ul style="list-style-type: none"> <li>↻ Footnotes/Tables options now available in <b>Scope</b> ▶ Containers.</li> <li>↻ Anchored/inline and hidden blocks now managed in <b>Scope</b> ▶ Containers as Inner (resp. Hidden) Frames.</li> <li>+ Scans cells in tables and sub-tables at any depth.</li> <li>+ Full support of endnotes.</li> </ul>

FIELD	MAIN CHANGES
Styles	<ul style="list-style-type: none"> <li>↻ Styles now managed in the <b>Filters</b> panel.</li> <li>+ Precise selection of character styles (resp. paragraph styles).</li> <li>↻ Removed the “Full style ranges” option, now controlled from <b>Filters</b> ▶ Character Styles ▶ Merge Spaces.</li> </ul>
Search	<ul style="list-style-type: none"> <li>↻ Query Lists are no longer limited in size (external text editor).</li> <li>+ Syntactic validation.</li> <li>+ Instant preview of matches using the Quick Test button.</li> <li>+ Matches “in context”.</li> <li>+ Fine-grained control of markers and hidden characters.</li> </ul>

FIELD	MAIN CHANGES
... Search	<ul style="list-style-type: none"> <li>+ Advanced Query List options and directives (<b>include</b>, <b>format</b>...)</li> <li>+ Supports of favorite queries.</li> </ul>
Options	<ul style="list-style-type: none"> <li>+ Added Generic Hyphen.</li> <li>+ Variables as Text option (interprets InDesign variables as pure text).</li> </ul>
Alphabet	<ul style="list-style-type: none"> <li>↻ Support of mixed alphabets.</li> <li>+ Added 20+ non-Latin alphabets (Cyrillic, Greek, Hebrew, Arabic...)</li> </ul>
Output	<ul style="list-style-type: none"> <li>↻ Now managed from <b>Ouput</b> ▶ Destination and ▶ Sorting.</li> <li>↻ Improved IDML and XML exports.</li> <li>+ Ability to generate Hyperlinks.</li> <li>+ Alphabetic Headings.</li> </ul>

FIELD	MAIN CHANGES
Page Report	<ul style="list-style-type: none"> <li>↻ Now managed in <b>Ouput</b> ▶ Layout.</li> <li>↻ Fully customizable separators.</li> <li>+ Range elision, special suffixes (<i>f</i>, <i>ff</i>).</li> <li>+ Note numbers insertable in locators.</li> </ul>
Hits	<ul style="list-style-type: none"> <li>↻ Additional fields and options.</li> </ul>
Misc.	<ul style="list-style-type: none"> <li>↻ New query engine (faster, safer).</li> <li>↻ Multi-level topics at any depth.</li> <li>+ Ability to filter out stop words.</li> <li>+ Added min/max conditions on term frequencies and shared pages.</li> <li>+ Ability to import InDesign indexes.</li> <li>+ Full control of <b>Generic Spaces</b> and <b>Generic Hyphens</b>.</li> <li>+ Import/export your settings (JSON).</li> </ul>

# Welcome to IndexMatic<sup>3</sup>



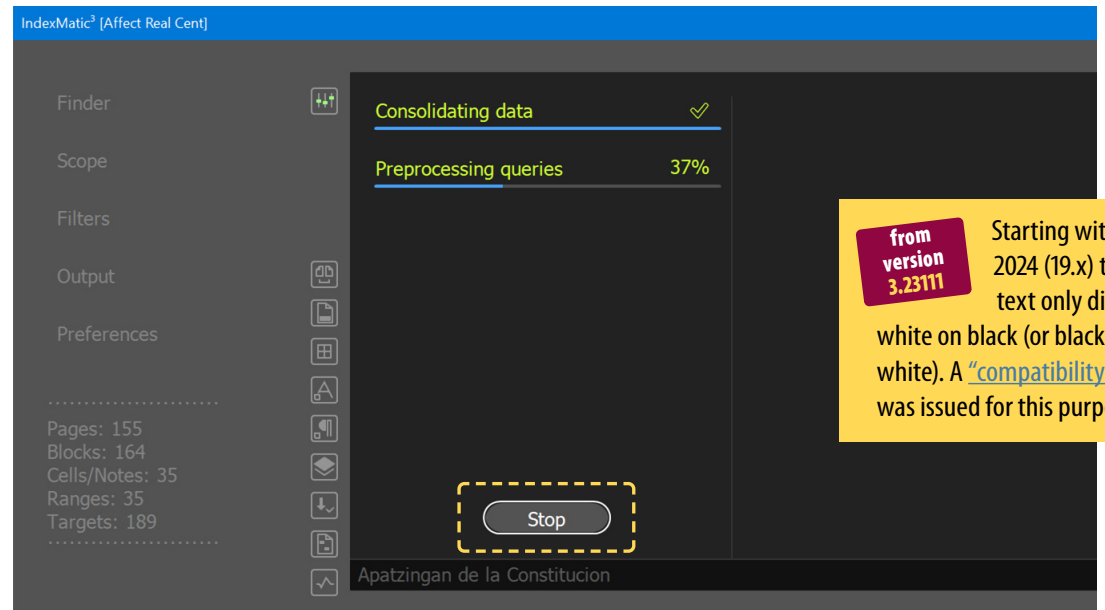
## 4. Live Preview in the Console

IndexMatic<sup>3</sup> allows you to test queries (⚡, 📄 and Hits buttons) and preview results right in the CONSOLE, a region of the main window that displays tasks in progress.

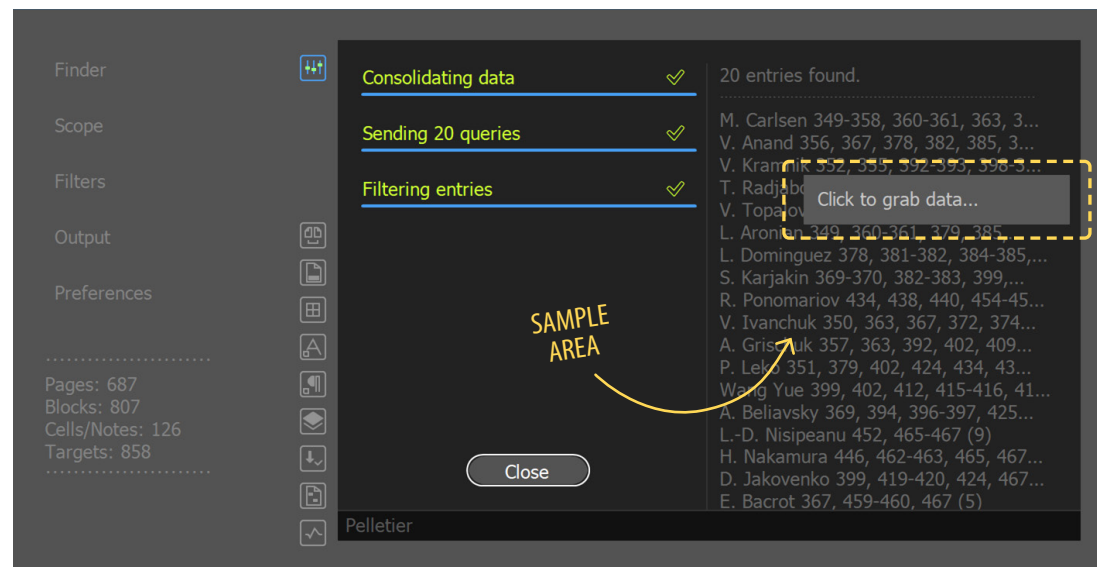
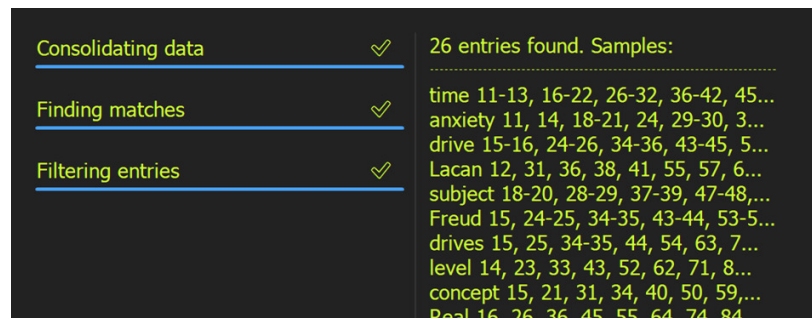
Some tasks (“Preprocessing queries”, “Finding matches”...) can be interrupted using the Stop button. This is very useful when you realize that a command is abnormally time-consuming—as it may happen with greedy requests.

The console gives you visual feedback of samples taken from the documents you are scanning or indexing. When all tasks are complete, you can close the console and the main dialog will re-appear.

In some circumstances (Hits, Matches-in-Context 📄...) you also have the option to “grab data” into a plain text file: hover the mouse over the SAMPLE AREA and click.



from version 3.23111 Starting with InDesign 2024 (19.x) the console text only displays white on black (or black on white). A “compatibility” version was issued for this purpose.



# Scope and Filters



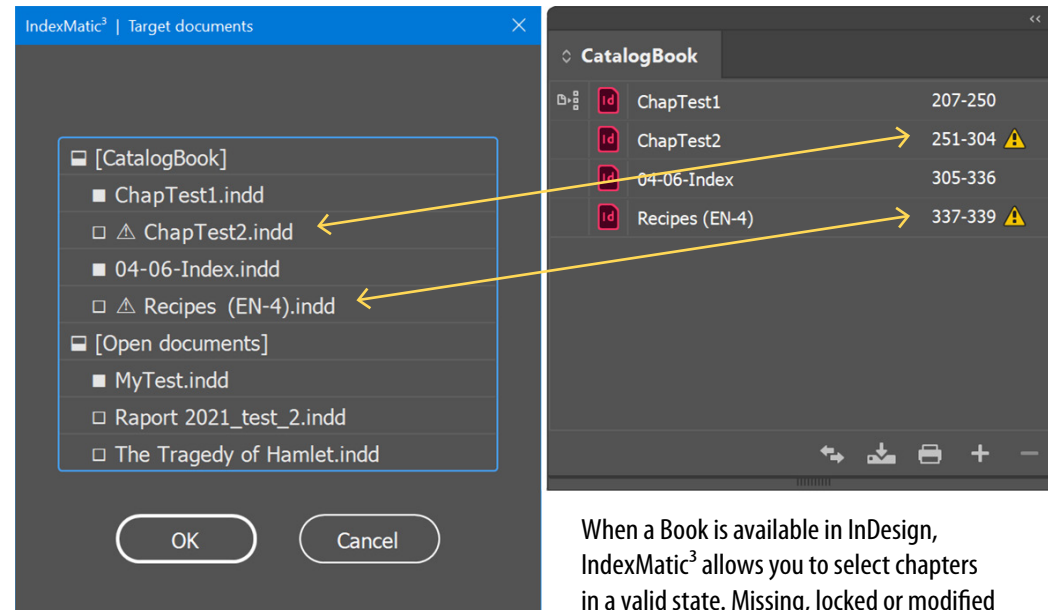
Once specified, documents, pages, and text containers determine a global search area—the **SCOPE**—that IndexMatic<sup>3</sup> will investigate in depth. The program won't look for anything outside the scope and it optimizes search tasks with respect to the scope, so you have everything to gain from defining this area as precisely as possible. In addition, a set of **FILTERS** (custom selection of styles and/or layers) can further reduce the scope to the minimum indexing workspace.

## 1. Setting your Target Documents

IndexMatic<sup>3</sup> can target one or more documents at once, assuming that the respective page numbers do not overlap. (If they do, you will be warned.)

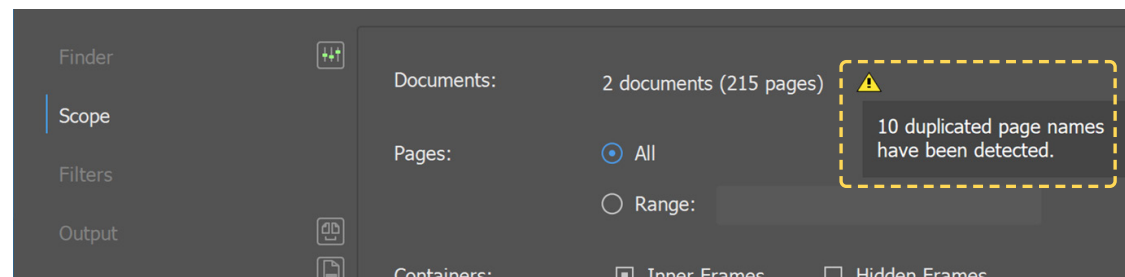
When several documents are available in InDesign while IndexMatic<sup>3</sup> is starting up, a dialog box appears allowing you to select the **TARGETS** to load in the scope.

- 1) Run IndexMatic<sup>3</sup>.
- 2) In the **Target Documents** dialog, click the checklist nodes to select or deselect items.
- 3) Click OK to confirm your selection.  
IndexMatic<sup>3</sup> completes its initialization and creates a scope based on your targets.



When a Book is available in InDesign, IndexMatic<sup>3</sup> allows you to select chapters **in a valid state**. Missing, locked or modified documents are excluded from the scope.

It is of utmost importance that your documents be valid and have consistent page numbers. Otherwise, a warning message would be displayed in the **Scope** panel. You can still generate index data, but they will refer to ambiguous page names.



# Scope and Filters



**NOTE** Behind the scenes scoped documents are ordered with respect to page numbers and a unique name is associated to the set. IndexMatic<sup>3</sup> uses this name to recover settings applied to a particular scope when re-used later.

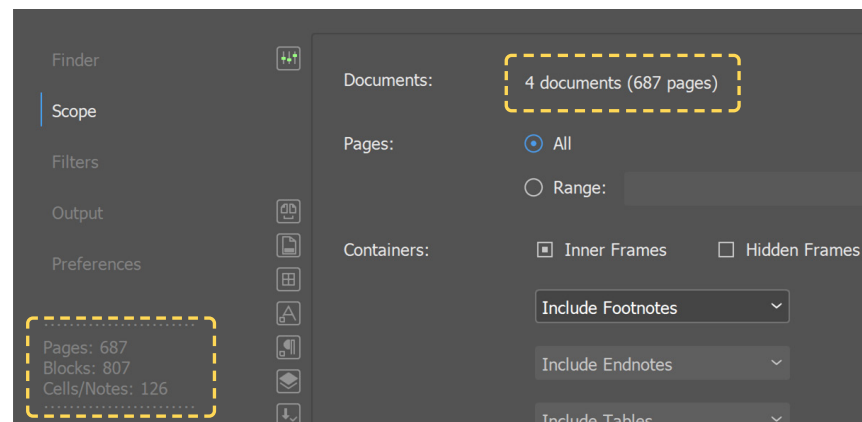
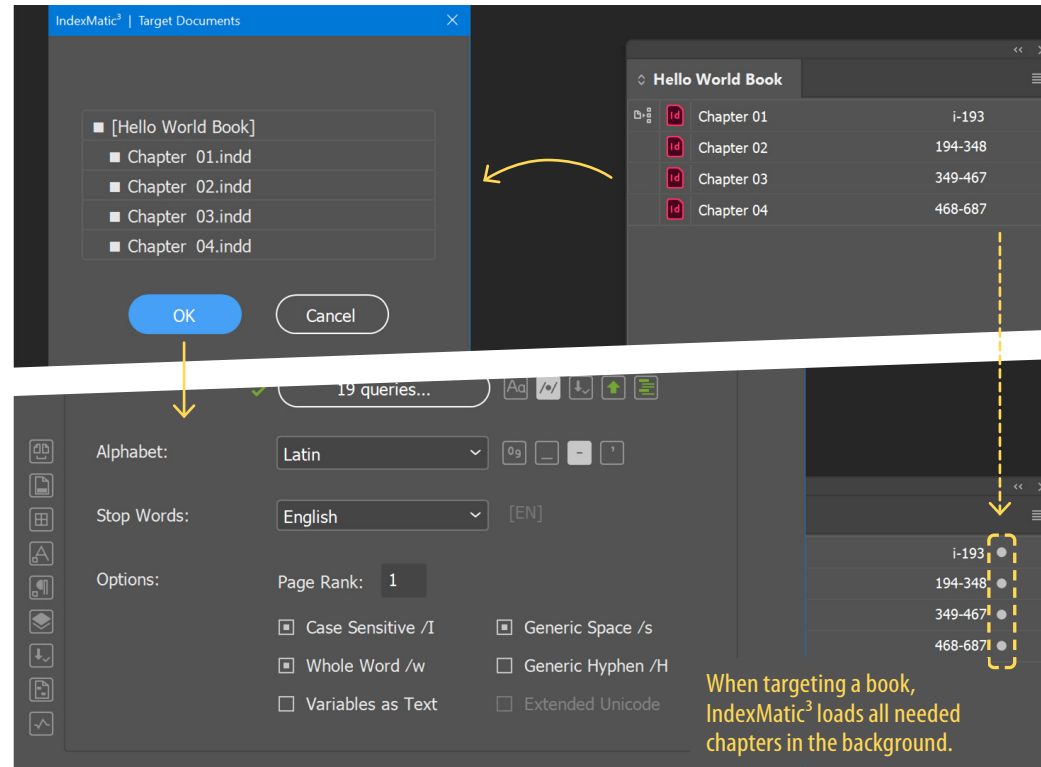
## 2. Indexing a Book

**DEFN** An InDesign book (.indb) is a collection of documents that can share styles, parent pages, and other items. You can sequentially number pages in booked documents.

There's nothing special in indexing a book with IndexMatic<sup>3</sup>, except that the script must be able to open all chapters in the background. Depending on the complexity of the documents, InDesign may be unable to open all these files at once. Make sure your book doesn't contain an unreasonable number of chapters.

**NOTE** ⚠ InDesign can open about 120 documents. Note also that IndexMatic<sup>3</sup> does not allow you to work on multiple books at once. Before running the script, open a single book in the Books panel.

- 1) Open a book file in InDesign and run IndexMatic<sup>3</sup>.
- 2) In the **Target Documents** dialog, select the book node, or any subset of chapters.
- 3) Click OK. It may take some time for the main dialog to appear: this is due to the opening time of the book chapters.



The Documents area displays the **total number of pages**. (Move the mouse over the field to get detailed information about each document.)



# Scope and Filters



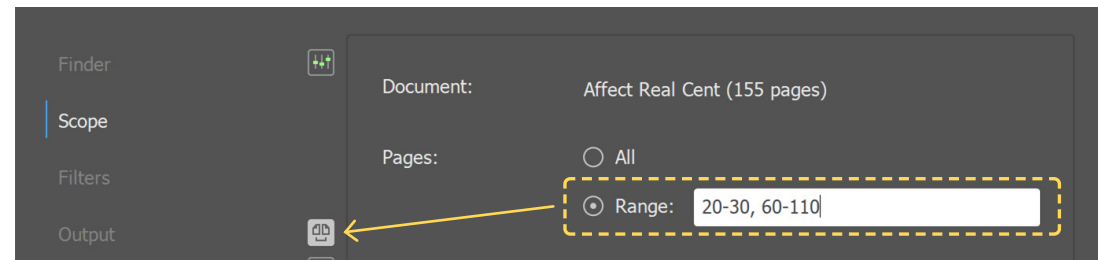
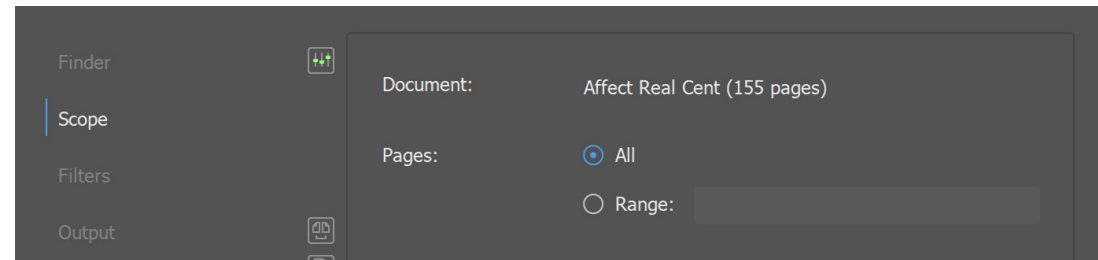
## 3. Reducing the Page Range

By default, all pages of the active scope will be scanned. (To restore this option, click the radio button **Scope ▶ Pages ▶ All**). If you want to index only a particular set of pages:

- 1) Select the **Scope** panel.
- 2) Click the radio button **Pages ▶ Range**.
- 3) Enter the page numbers in the edit box. You can indicate a range by using a hyphen (e.g. “120-150”), and indicate multiple pages or ranges by using commas or semicolons.

- Pages can be specified using either their display name (123, iv, 009, ح) or a number in brackets, like “[45]”, which then refers to the ABSOLUTE POSITION of the page in the scope.
- The explicit syntax <section>:<page> is supported too, e.g. “MySection:23”.
- Open ranges are allowed: “-56” specifies all pages up to and including page 56; “78-” specifies all pages from page 78 to the end of the scope.

**NOTE** A warning symbol ⚠ may appear while you are entering a range. It warns you that one or more documents are presently excluded from the scope due to the specified page range. Out-of-range documents will be ignored by the query engine.



**NOTE** If a page specifier is ambiguous due to duplicated page names, IndexMatic<sup>3</sup> will target all matching pages. For example, if 5 is the name of some page in doc1, doc2, and doc3, then the specifier “5” will be translated into three absolute positions, e.g. “[5], [17], [42]”.

The program reminds you that a limited range is active by turning on the Page Range indicator.

## 4. Focusing on Special Text Containers

**DEFN** Text frames, text paths, table cells, and footnotes form a large group of components that we collectively refer to as text containers. Introduced in InDesign 13, endnotes flow in a special kind of text frame.

IndexMatic<sup>3</sup> basically deals with texts and locations. Most of the time, you expect it to scan all text units wherever they are. But it may happen that specific containers are your actual work area.



# Scope and Filters



For example, you may need to extract all bibliographic references from the endnotes, and not from other locations. Or you want to index product names that appear specially in tables, not those mentioned in the primary text. Conversely, you could index the main text disregarding footnote and/or endnote content.

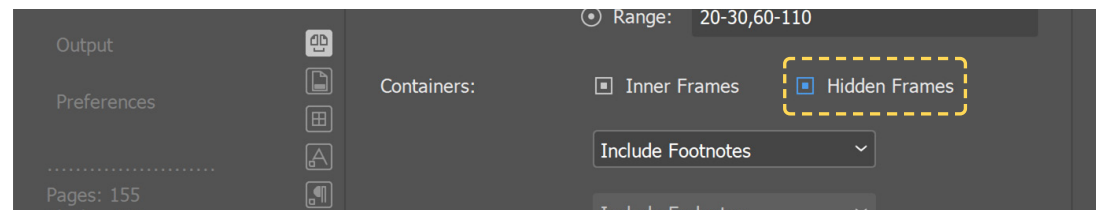
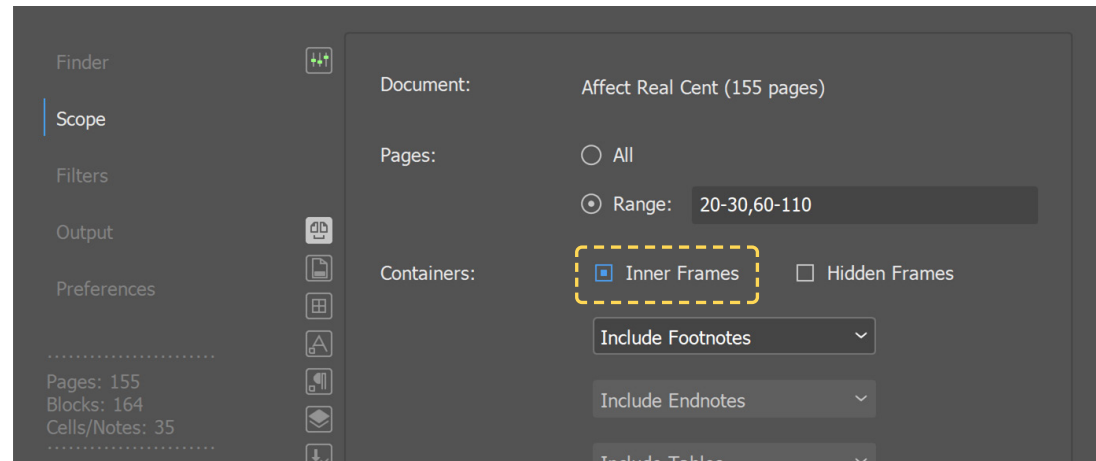
In any of these situations, the **Containers** options allow you to refine your scope as desired.

## ► *Inner Frames*

Check **Scope** ► **Containers** ► *Inner Frames* to include ANCHORED, NESTED, and GROUPED text frames in the scope. This option is recommended if your layout has a complex structure and texts can be found within sub-objects, that is, “pasted into” a parent object, anchored or parts of a group, button, multi-state object.

**TIP** Inner frames are a bit slower to scan than primary frames. If your index just has to focus on the main story (made up of single or threaded containers being direct children of the pages), you can speed up IndexMatic<sup>3</sup> by disabling *Inner Frames*.

If the *Inner Frames* option is turned off, any text and any text container within such frames will be ignored—i. e. sub-frames, footnotes, tables, etc.—regardless of other settings that may still apply to primary frames.



## ► *Hidden Frames*

Turn on **Scope** ► **Containers** ► *Hidden Frames* to include hidden objects in the scope. This option is usually off, unless you want to report text data that the readers cannot see!

**TIP** Indexing hidden blocks would make sense if they contain secret tags, keywords or topics that are relevant to your index. However, a better solution is to have such data on a dedicated layer (possibly hidden) that you can target from **Filters** ► **Layers**.

# Scope and Filters



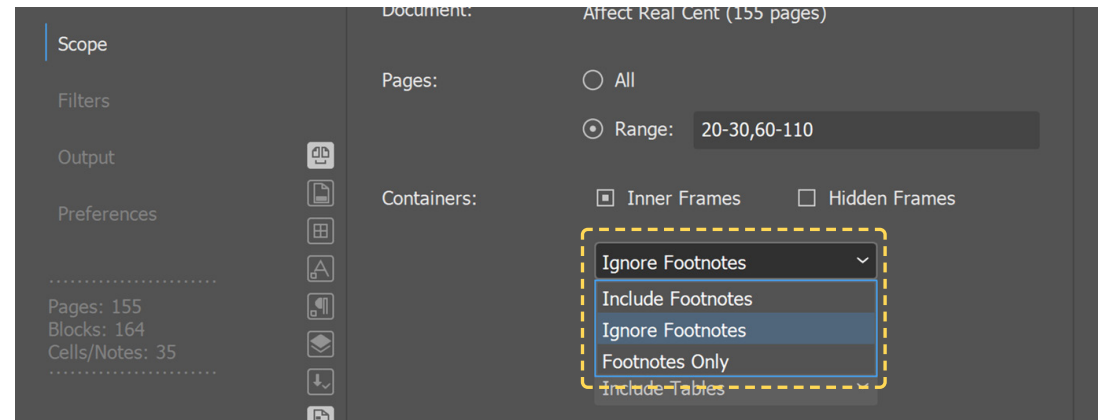
Keep in mind that the *Hidden Frames* option specifically refers to objects made invisible using InDesign's Object > Hide command. This does not apply to:

- Regular objects that belong to a hidden layer. In this case the invisibility results from the layer state; IndexMatic<sup>3</sup> may still access hidden layers with respect to **Filters ▶ Layers**.
- Hidden texts resulting from a Conditional Text presently turned off. IndexMatic<sup>3</sup> will not scan hidden texts since they are not supposed to have a definite page location.
- Overset text that cannot be shown in the layout. Here again, IndexMatic<sup>3</sup> is not instructed to scan **off-page content**.

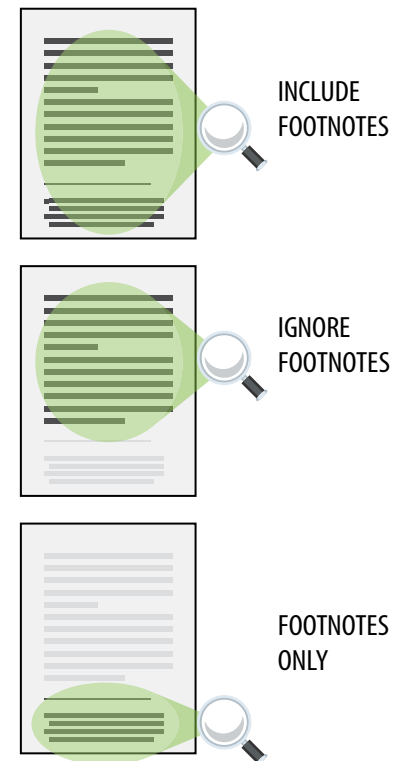
**DEFN** Each text frame contains an *in* port and an *out* port, which are used to make connections to other text frames. A red plus sign (+) in an *out* port indicates that there is more text in the story to be placed but no more text frames in which to place it. This remaining unseen text is called *overset* text.

## ▶ Footnotes

Although they are logically *attached* to a parent container (story or cell), IndexMatic<sup>3</sup> regards footnotes as independent text units, so it can retrieve their particular content and location in the stream. You can choose from the following three options:



- *Include Footnotes*. Every text container in the scope will have its footnotes explored as well. This is the most natural option, putting the main text and its footnotes on equal terms.
- *Ignore Footnotes*. All document footnotes will be ignored, that is, footnote texts won't be scanned at all. Choose this option if footnotes are irrelevant to your index.
- *Footnotes Only*. Every text container in the scope will have its footnotes scanned exclusively, not the main text. This special option is of infrequent use, but it allows you to produce a detailed report of the notes regardless of the primary content.



# Scope and Filters

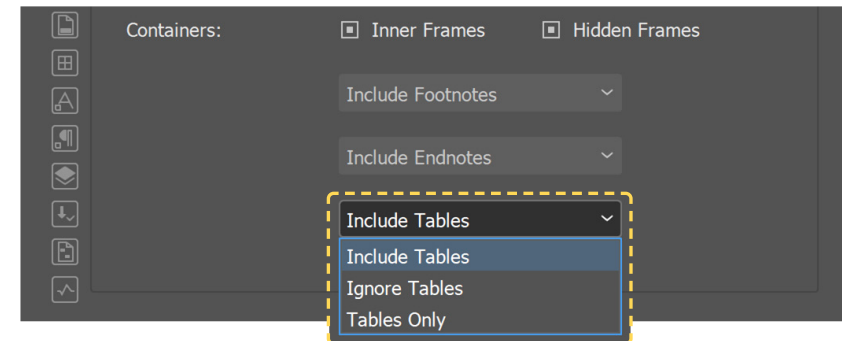
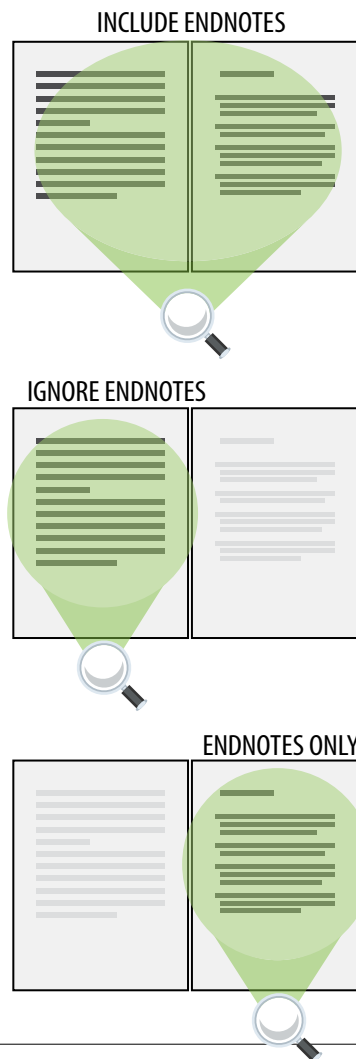


## ► Endnotes

Endnotes are organized at the end of the document in specific **ENDNOTE FRAMES**. IndexMatic<sup>3</sup> treats these frames as independent text containers. You can choose from the following three options:

- *Include Endnotes*. Every endnote frame will have its text scanned as well as the story it depends on. This is the most natural option: it puts the main text and its endnotes on equal terms.
- *Ignore Endnotes*. All document endnotes will be ignored, that is, endnote frames won't be scanned at all. Choose this option if endnotes are irrelevant to your index.
- *Endnotes Only*. Every story in the scope will have its endnote frames scanned exclusively (not the story itself). This special option allows you to produce a detailed report of the notes regardless of the primary content.

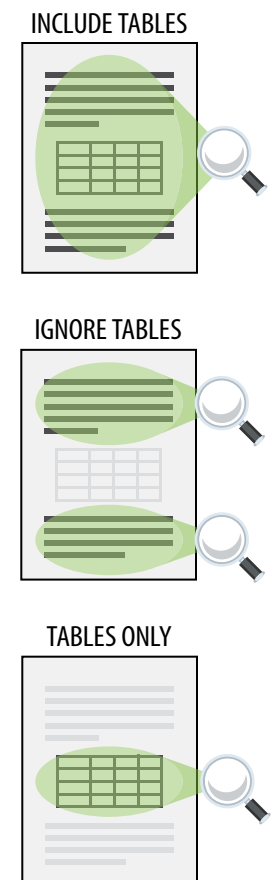
**NOTE** Endnotes belong to actual text frames that could *per se* be excluded from the scope due to page range, hidden state or other filters. So, when you include endnotes, make sure that other restrictions don't prevent them from being scanned.



## ► Tables

InDesign tables are somehow *anchored* in text streams, but IndexMatic<sup>3</sup> regards each table cell as an independent text container. You can choose from the following three options:

- *Include Tables*. Every text container in the scope will have its tables explored as well. This is the usual option, putting the main text and table data on equal terms.
- *Ignore Tables*. All document tables will be ignored, that is, table cells won't be scanned at all. Choose this option if table data are irrelevant to your index.
- *Tables Only*. Every text container in the scope will have its tables scanned exclusively, not the main text. This option allows you to specially focus on table data.



# Scope and Filters

## 5. Multiple Conditions on Text Containers

Unless otherwise specified, conditions that reduce the scope must all be met. For example, the settings

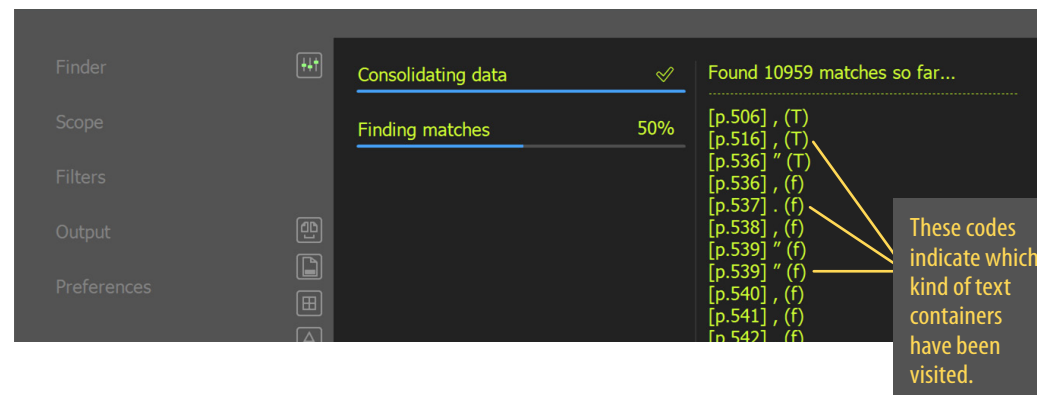
*Page Range: 100-200 + Inner Frames: off + Tables Only*

specifically target tables from page 100 to page 200, excluding those that belong to nested, anchored or grouped text frames.

An exception to this rule is the mixing of selectors weighing on footnotes, endnotes, and tables. Those are independent, because they refer to text units which are inherently disjoint: a table cell cannot embed a footnote (although it may own a footnote reference), a footnote cannot belong to an endnote frame. Hence, even if you select the options *Footnotes Only + Tables Only*, IndexMatic<sup>3</sup> considers that it must explore both table cells (wherever they are) and footnotes. The same is true for the settings *Footnotes Only + Endnotes Only*: all notes will be explored.

A consequence of this law is that you cannot target all tables, *including those nested in notes*, without also targeting the notes themselves. If you need more granularity, you still have the option to apply style filtering...

**NOTE** In the particular case where table cells are excluded from the scope while footnotes and/or endnotes are included, any table found inside a note would be ignored.



## 6. Container Codes and Clusters

While reporting samples in the console, IndexMatic<sup>3</sup> may display informative codes in parentheses. In the example shown above, a search has been launched to analyze the punctuation marks of a document. The codes indicate that the matches were found in either regular text frames (T) or footnotes (f). These abbreviations represent text containers according to the following table:

CODE	CONTAINER	↔	DESCRIPTION
(T)	Regular Text Frame	+	Any text frame (excluding endnote frames).
(P)	Text Path	+	Special container whose text is rendered along the path of a spline item.
(E)	Endnote Frame	+	A specialized text frame that handles endnotes.
(f)	Footnote	+	Special text unit rendered at the bottom of a regular frame.
(c)	Frame Cell	-	Table cell found in a frame of type (T) or (E).
(d)	Footnote Cell	-	Special case of a table cell nested in a footnote.

The uppercase codes (T), (P), and (E) indicate text frames and independent containers.

The lowercase codes (f), (c), and (d) indicate footnotes and cells.

The + sign in the table indicates elements that support CHAINING.

# Scope and Filters



Chainable containers are a key concept in IndexMatic<sup>3</sup>. They determine the fundamental text strands that the script can handle, either for searching matches or for assigning page locations.

Consider the examples below. Figure A shows two threaded text frames that may or may not belong to the same page. Of course this thread could run across the entire document (through other chained containers).

Figure B shows the (very) special case of a text path chained with a regular frame. Note that the substring “*maios sam*” must be identified in the stream, even though “*maios*” (last word in the text path) and “*sam*” (first word in the text frame) belong to two distinct categories of containers.

In Figure C, a footnote is *continued* into the next (threaded) footer. The substring “*Qui omniam*” plainly belongs to that footnote, regardless of visual break or even page break.

Cases A, B, and C are examples of chained text containers. Each chain defines, in IndexMatic<sup>3</sup> terminology, a TEXT CLUSTER. A cluster is a consistent stream that might be shared across multiple locations but should still be treated as a continuous text unit.

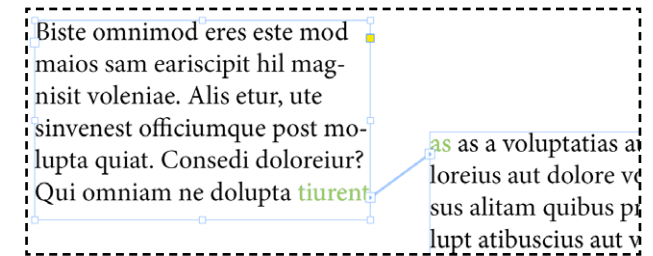
Figure D shows a table flowing in two threaded frames. However, cells themselves cannot be divided into subparts and each of them is assumed to form a single text unit. So every cell is a cluster *per se*.

To sum up, a cluster represents the longest string of text that IndexMatic<sup>3</sup> can investigate at a time. All clusters are perfectly sealed from each other. A cluster may have multiple paragraphs and may run across multiple pages.

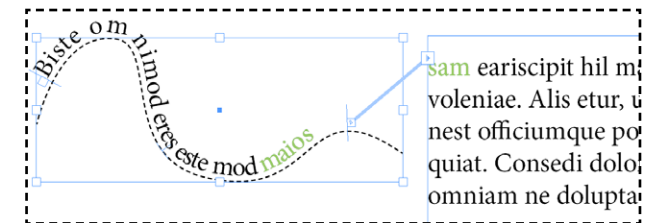
**NOTE** Since footnotes and cells are independent clusters, fully detached from the parent stream where they visually appear, you may ask yourself what “footprints” they leave in that parent stream. These are special characters (markers, anchors) that you will manage from the **Finder**.

**NOTE** Although contiguous, a cluster may be somehow “truncated” (i.e., incompletely visited) due to **Scope** restrictions like Page Range, etc. For example, if a continued footnote ends in a page that is excluded from the scope, that part won’t be indexed.

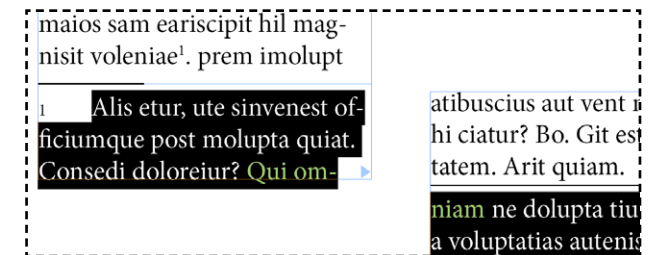
## A THREADED FRAMES



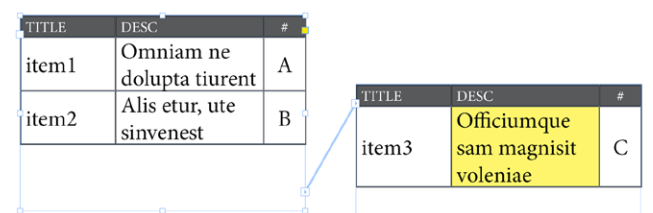
## B THREADED TEXT PATH AND FRAME



## C CONTINUED FOOTNOTE



## D TABLE IN A THREAD



# Scope and Filters




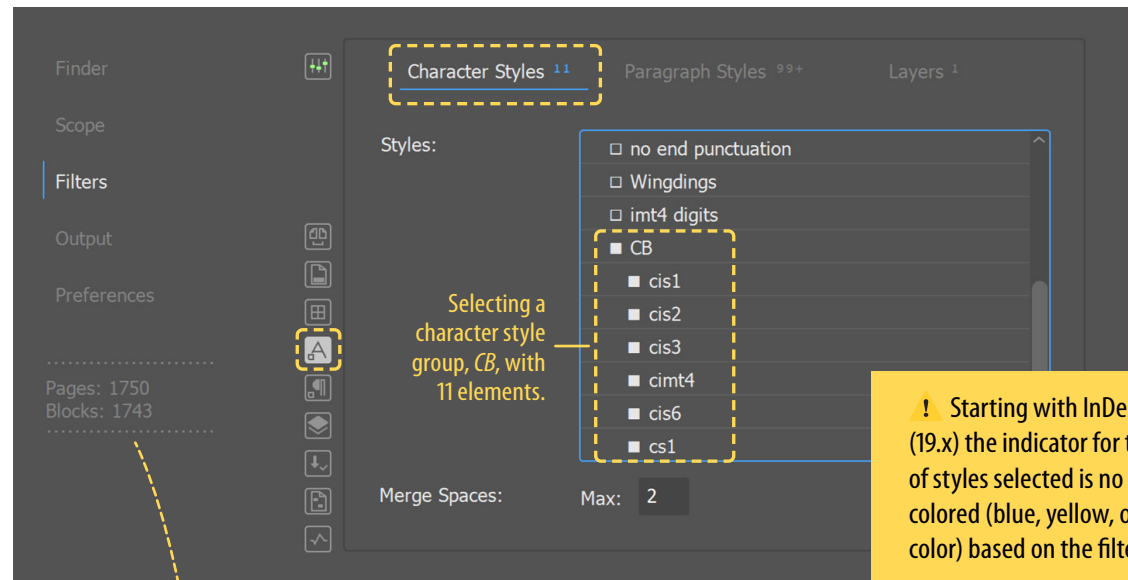
## 7. Targeting Character Styles

**DEFN** A text style range is a continuous range of identical text formatting attributes.

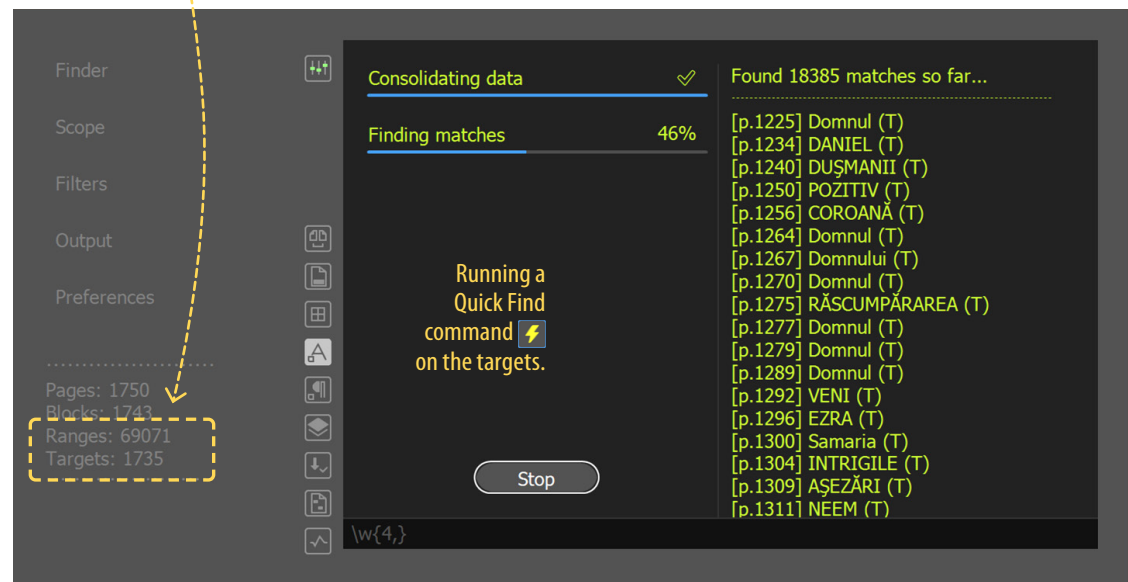
While the **Scope** panel basically defines which pages and text containers IndexMatic<sup>3</sup> will scan, **Filters** ► **Character Styles** allows you to select particular style ranges within the candidate clusters. In a way this further reduces the search area in creating subclusters, that is, the specific cluster parts (also called **TARGETS**) that satisfy the filter.

When you select a subset of styles in the **Character Styles** checklist, several changes take place:

- The number of selected nodes is displayed in blue at the top of the panel (11 in our example, *see screenshot*).
- The Character Style indicator  lights up, informing you that character style filtering is active.
- IndexMatic<sup>3</sup> prepares the clusters to be sliced into consistent targets. As soon as a command is executed in the console, the information area is updated and displays two numbers:
  - *Ranges* (69,071 in our example) is the total count of text style ranges available in the document—it is computed once, since it doesn't depend on the active filters;
  - *Targets* (1,735 in our example) reflects the number of targets found in the active scope with respect to filters.



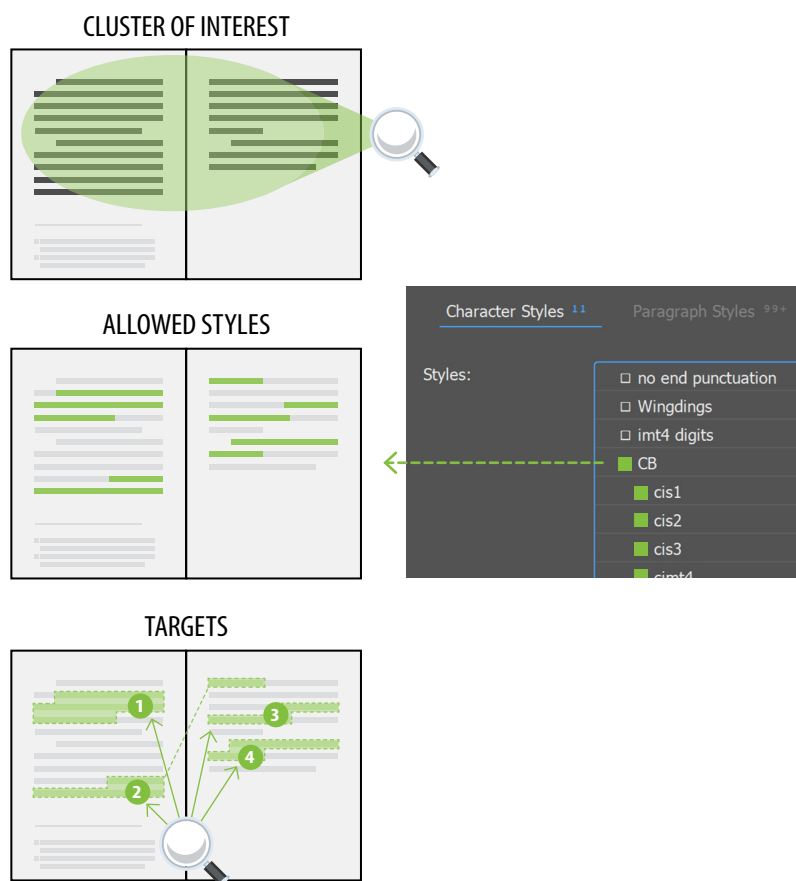
! Starting with InDesign 2024 (19.x) the indicator for the number of styles selected is no longer colored (blue, yellow, or neutral color) based on the filter status.





# Scope and Filters

The mechanism behind style filtering is easy to visualize. Given a cluster of interest (e. g. a two-page story), the program identifies every text range having one of the allowed styles applied (in our example, any style of the *CB* group). Then it forms the targets by merging continuous ranges.



This last step ensures that targets will only be split if strictly necessary. Suppose that the line

*It is a SAD and beautiful* WORLD.

has five distinct character styles applied (one for each color), every style being allowed by the filter anyway. IndexMatic<sup>3</sup> will create a single target,

It is a sad and beautiful world.

rather than five separate elements.

## 8. Stepping Over Unstyled Spaces

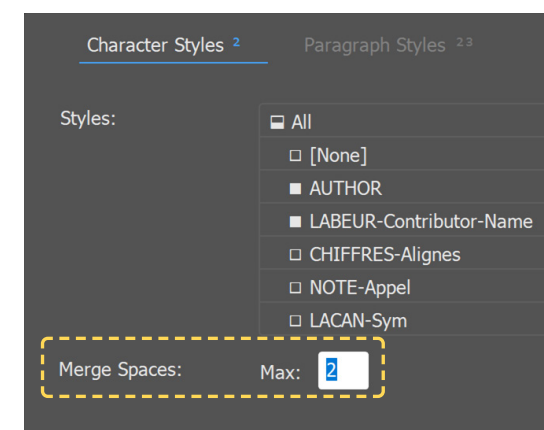
In our previous example, a slight problem may arise. Most of the time, the white spaces between two character style ranges have no specified style—i.e., their own style is *[None]*—or may be inconsistently formatted relative to the present filter:

*It is a* | *SAD* | *and* | *beautiful* | *WORLD.*

But you likely don't expect these spaces to break the cluster into five pieces:

It is a ; sad ; and ; beautiful ; world.

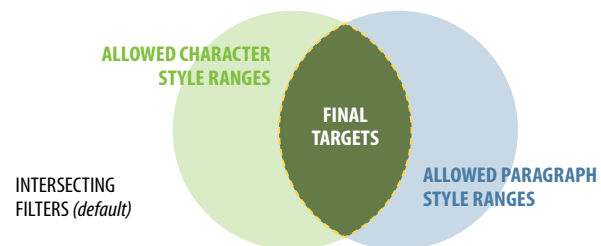
To prevent this from happening, go into **Filters** ► **Character Styles** ► **Merge Spaces** and increase the *Max* value to the number of space characters always allowed between two accepted ranges. (The optimal value is usually 1, the greatest is 9.)



## 9. Targeting Paragraph Styles

**Filters** ► **Paragraph Styles** works exactly the same as the **Character Styles** filter, so we won't repeat the story! (Go back to → **Targeting Character Styles** to refresh your memory.)

However, it is worth explaining how the two filters interact. If both a set of paragraph styles *and* a set of character styles are selected in the respective checklists, then IndexMatic<sup>3</sup> will look for text ranges that both have one of the allowed paragraph styles *and* one of the allowed character styles.

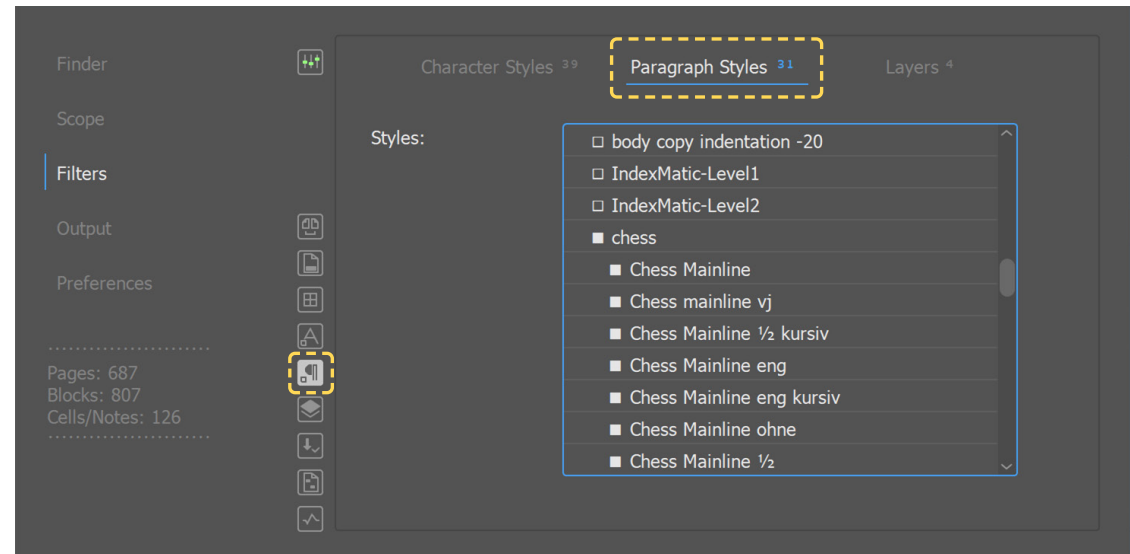


Visually, this amounts to considering the intersection of the underlying style ranges.

**UPD.** Rather than an intersection, you can now force the engine to perform a union of styles (i.e. combine them independently) by activating the option **Preferences** ► **Filters** ► **Make Paragraph and Character styles independent**.

from version 3.24011

In practice, these two filters are rarely used together as they serve different purposes. Paragraph styles often help you focus on semantic dimensions (main content, notes, etc.); they work



well in automatic mode, with word lists, or with regular expressions that capture lexical features. On the other side, character styles are usually dedicated to data markup—at least in systematic documents such as product catalogs—so they work fine with queries that grab either the style range as a whole, or a strictly formatted part of it (codes, numbers, etc.)

⚠ Keep in mind that a paragraph style remains consistent across an entire paragraph, while distinct character styles may appear within one paragraph.

## 10. Style Overrides

IndexMatic<sup>3</sup> is insensitive to style overrides or any text setting (font, size, weight, color, etc.) that you do not address from a style specification.

This design choice has been carefully considered. Integrating the huge range of settings that you enjoy in InDesign would have

# Scope and Filters




dramatically weighed down the usability of the user interface. Basically, this would have led us to rebuild the Find / Change dialog entirely.

Fortunately, InDesign offers you the best tool to get around this limitation... the Find / Change dialog itself! Indeed, if you need to index all text in bold italic and there is no style dedicated to this formatting, you can almost instantly create a *BoldItalic* character style and apply it consistently to every text range that meets your criteria. And what is true of “bold italic” is true of *any* other set of text features available in InDesign.

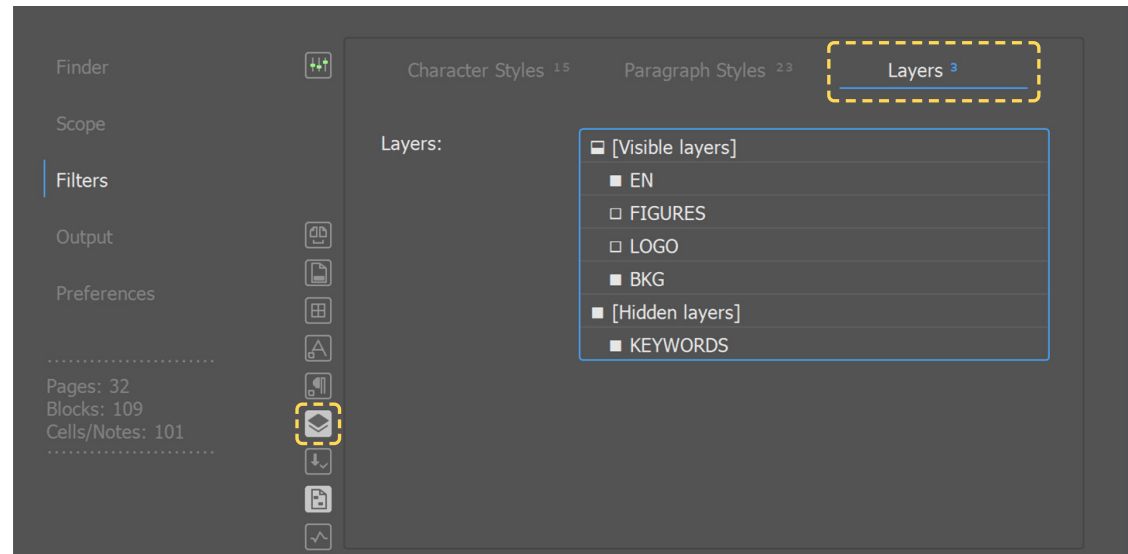
**NOTE** ⚠ For the time being, IndexMatic<sup>3</sup> is insensitive to nested and GREP styles, in the sense that it cannot detect in the text the actual application of these character styles as dictated by the parent paragraph style.

## 11. Targeting Layers

**Filters** ► **Layers** works as other filters. Just select in the checklist the specific layers that IndexMatic<sup>3</sup> should inspect.

The Layers indicator  lights up whenever layer filtering is active. The scope is then reduced to text containers that belong to one of the allowed layers.

If some layers are hidden, you can globally select either the *[Visible Layers]* branch or the *[Hidden Layers]* branch, or both.



## 12. Disabled vs. Full-State Filters

In whatever **Filters** subpanel, the number of explicitly selected items appears in blue, indicating that a subset of the whole collection is active. This blue coloring vanishes when the filter either is disabled (no item selected), or has no actual effect because all items are selected. The disabled state is therefore equivalent to the *full* state in terms of filtering.

However, having all items selected makes a slight difference in your settings: those explicit items can be saved and restored later. So, should new styles or layers be added to the document(s) in the meantime, the restored state of the filter may no longer include *all* items. Whereas a disabled filter is restored as is.

⚠ A bug prevented the restoration of selected layers across sessions. This problem has been corrected in **v. 3.24012**.

# Scope and Filters



## 13. Multi-Document Scope and Filters

In a multi-document environment, scope and filter settings like page ranges or style names are addressed globally. This sometimes leads to attributes being presented together which might in fact only apply to particular documents. For example (see screenshot), the character styles *Italic* and *SmallCaps* only exist in three out of four documents.

As a result, your **Scope** and/or **Filter** settings could materially exclude a particular document in its entirety, which is probably not your goal. IndexMatic<sup>3</sup> anticipates such inconsistencies and warns you that some document is presently excluded due to either page range, layer selection, and/or style selection.

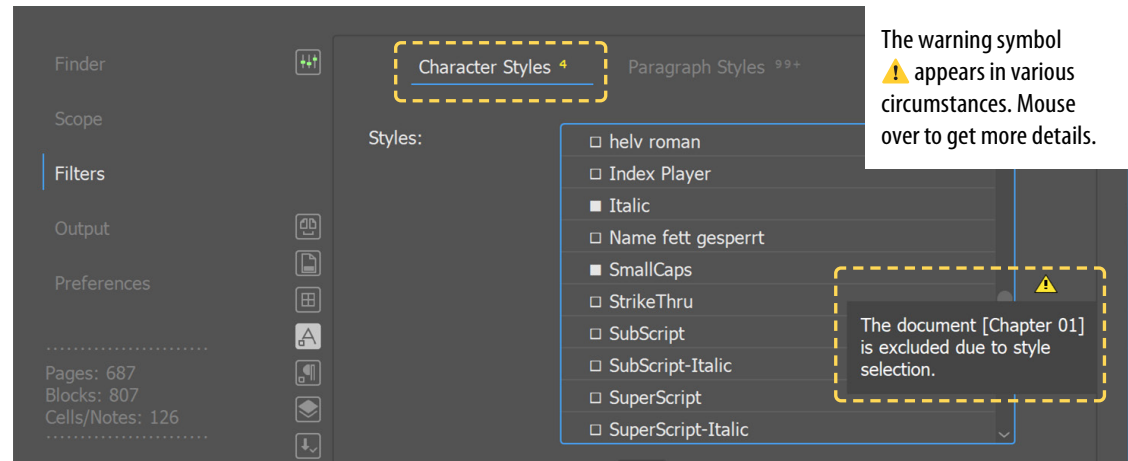
**NOTE** A filter having a warning symbol will display the number of selected items in yellow (except in InDesign 2024).

## 14. Off-Page Content and Ignored Elements

**DEFN** In InDesign, it is commonly accepted that a content is off-page when the container does not touch any page area.

IndexMatic<sup>3</sup> ignores off-page content, text containers on parent pages, overset texts, and hidden texts (due to disabled conditions). Such elements have no actual page location.

Consider the following examples:



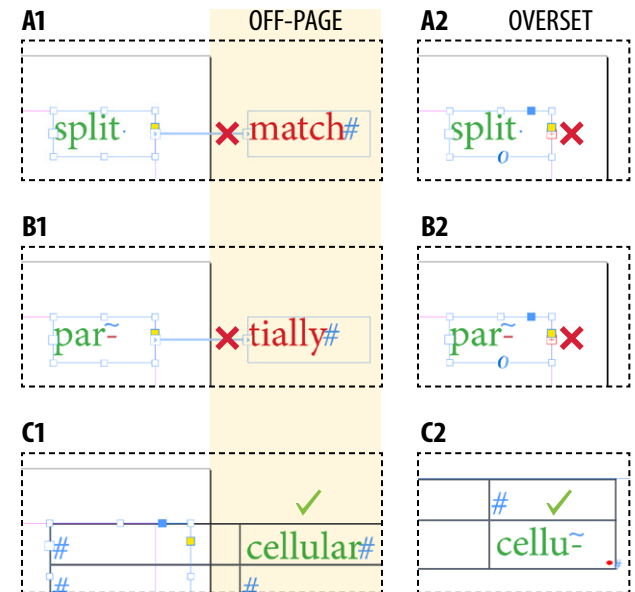
• In **A1** and **A2**, the expression “*split match*” cannot be seen entirely, hence only the word “*split*” is identified.

• In **B1** and **B2**, only the substring “*par*” is seen, not the word “*partially*” (hyphenated).

Cases **C1** and **C2** are exceptions:

• In **C1**, the cell “*cellular*” is scanned—as the table belongs to a frame that itself belongs to a page.

• In **C2**, although the cell overflows, the word “*cellular*” is entirely retrieved too.



# Finder and Basic Queries



The **Finder** is the first panel you see when the dialog shows up. Why? Because this is very likely where you will spend most of your time! At its core level, IndexMatic<sup>3</sup> is a search engine: it seeks **MATCHES** in your scoped document(s). The Finder is the place where you send **QUERIES** for that purpose.

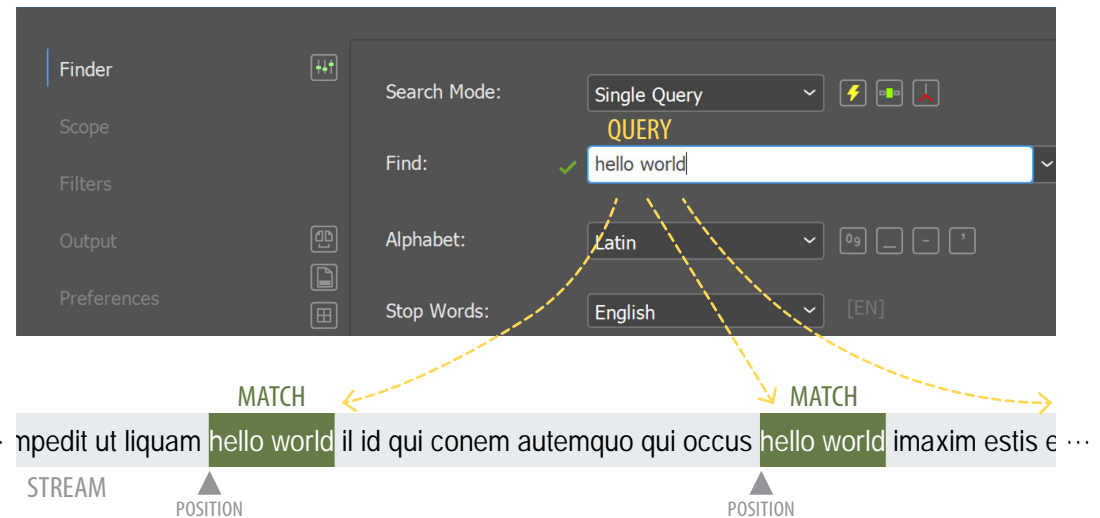
## 1. About Queries, Matches, and Locators

From now on, don't worry about scope and filters anymore. Just imagine a continuous text **STREAM**, like a long strip of paper, which contains the full speech, the complete data sequence, that you want to parse and index.

**NOTE** We called that stream a text cluster, and we learned that multiple independent clusters may in fact run in parallel, but once you understand how IndexMatic<sup>3</sup> handles one stream it is easy to generalize.

On the strip are printed only characters. Styles, colors, and even page breaks are gone. Indeed, if you have to find the occurrences of “hello world” in that stream, other text attributes no longer matter since you have filtered the areas you are interested in beforehand.

Technically, you will ask IndexMatic<sup>3</sup> to search the string “hello world”, then to report every occurrence encountered between the beginning and the end of the strip of paper. That's it.



### ► Query

The command you submit to the **Finder** is known as a **QUERY**. In the present case the query is **hello world** and the engine will search for exactly that character string. (Certain assumptions or restrictions might come into play, but that's not important for now).

### ► Match(es)

On our strip of paper, every piece of text that satisfies the query is referred to as a **MATCH**. In our basic example, each match will—unsurprisingly—contain the string “hello world”. Should seventeen matches be found in total, we say that the query has **CAPTURED 17 matches**. Every match has a unique **POSITION** in the stream. How the program internally manages those positions is of no interest, what you have to know is that any position finally determines a page location.

### ! IMPORTANT RULE

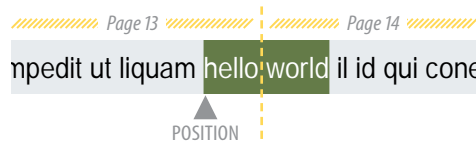
No query (and then no match) can address multiple paragraphs. The paragraph is the largest “unit of text” that IndexMatic<sup>3</sup> can handle at a time. For that reason, ends of paragraph and similar characters (`\r`) are internally treated as **BREAK markers**.  
→ More on special characters in [Managing Markers and Special Characters](#)

# Finder and Basic Queries



## ► Locators

By convention, the position of a single match is translated into a **LOCATOR** (usually a page number, sometimes with a note number). It refers to a region of a document where the reader can see the first character of the match.

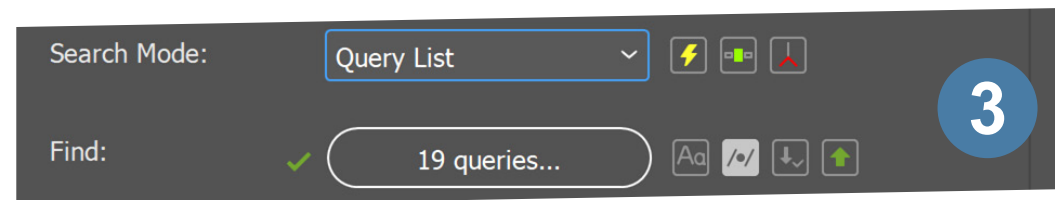
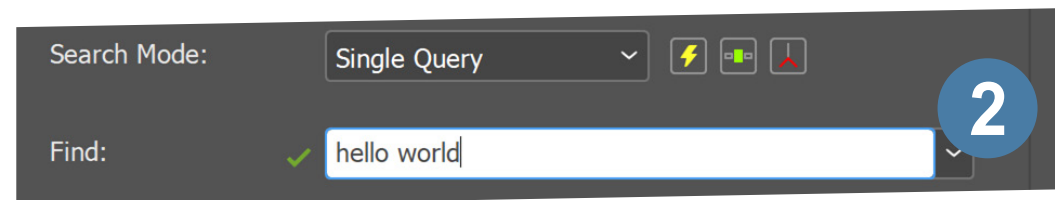
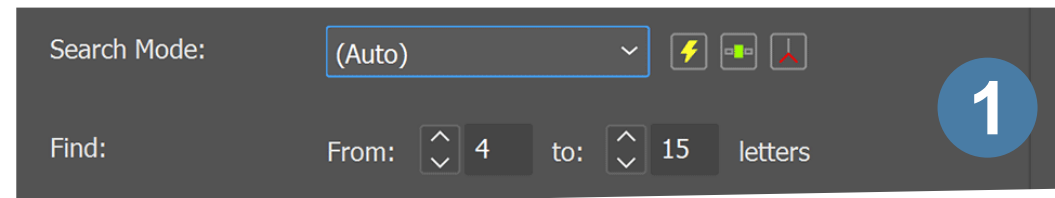
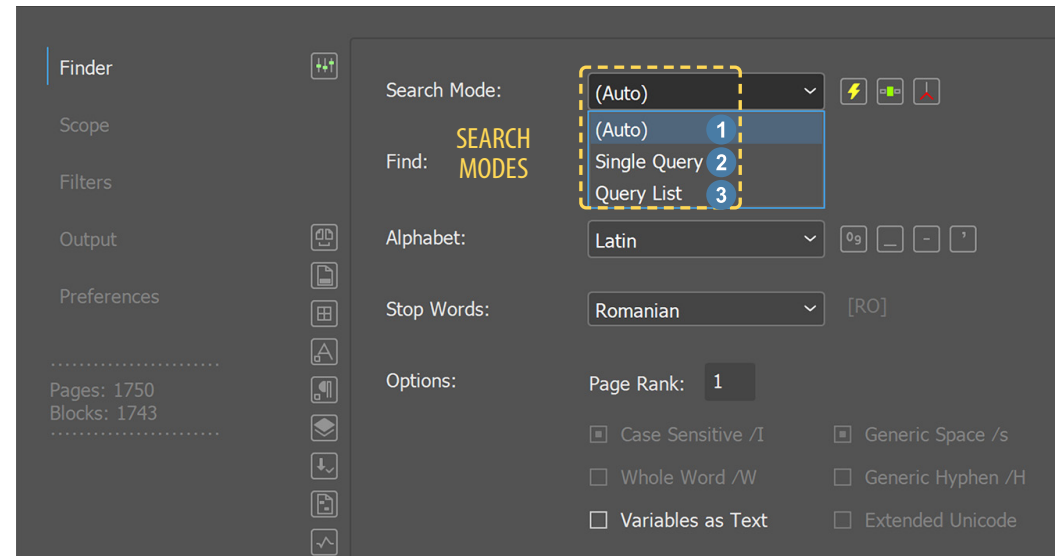


Suppose that an occurrence of “*hello world*” spans two pages (starting on page 13, ending on page 14), then the match is attached to page 13 anyway.

## 2. About Search Modes

IndexMatic<sup>3</sup> offers three ways of sending queries (**Finder** ► **Search Mode**), each mode having a dedicated interface:

- **(Auto)** **1** only captures single words without space, based on an implicit query that identifies any sequence of letters whose length is between a minimum and a maximum. (A few extra characters such that the hyphen are allowed, depending on the **Alphabet** options—see next pages).
- **Single Query** **2** sends an explicit query that can either identify a **LITERAL** expression (like `hello world`), or a **REGULAR** expression (like `/hello \w+/`). In the latter case, the query starts with a slash character (`/`).





# Finder and Basic Queries




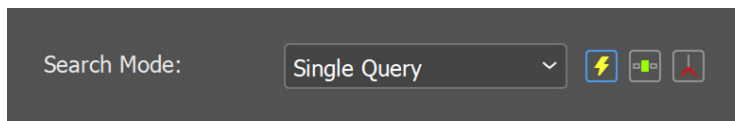
- **Query List** 3, the most advanced mode, loads a set of queries from a plain text file, sends them in sequence and collects all matches.

**NOTE** In *Single Query* and *Query List* modes, a particular query can be either literal (*token query*) or regular expression (*regex query*). In this chapter we focus on token queries, as they do not involve syntactical subtleties.

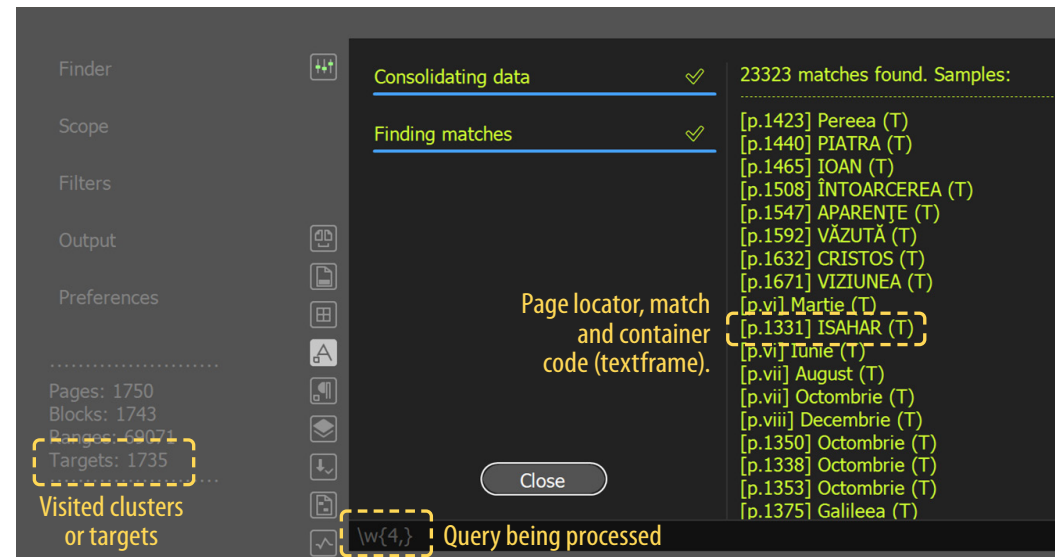
## 3. Previewing Matches

In whatever **Search Mode**, you can preview a subset of the captured matches without further refinement. It's a handy way to check that your query can at least find something in the scope.

- 1) Activate the **Finder** and select your **Search Mode**.
- 2) Once your query is set (and/or the underlying parameters), click the Quick Matches button .



- 3) The console opens and displays samples, in no particular order, associated to a page locator [in brackets] and a container code (in parentheses). The total number of matches is indicated too.
- 4) Click the **Close** button to hide the console.



Previewing matches involves at least two tasks, “Consolidating data” and “Finding matches”. The first task prepares the text clusters (taken from the scope) to be scanned. Although it may be time-consuming with huge documents, it is optimized to run quickly on subsequent calls, even if you have significantly changed your scope settings or filters in the meantime.

Matches sampled from a long document (1,750 pages) using a regex query that captures words having at least four letters. The console only shows about fifteen out of 23,323 matches found.

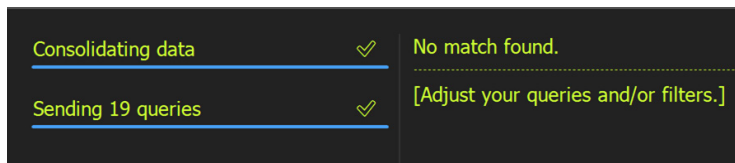
**TIP** You can interrupt these tasks during their progress by clicking the temporary available **Stop** button.

**NOTE** In the next chapters you will discover that *matches* and *hits* are **distinct concepts**. For the time being, just keep in mind that previewing matches exactly reports text samples as captured in the document(s), i. e., in their original form. → **Processing Hits Reports**

# Finder and Basic Queries



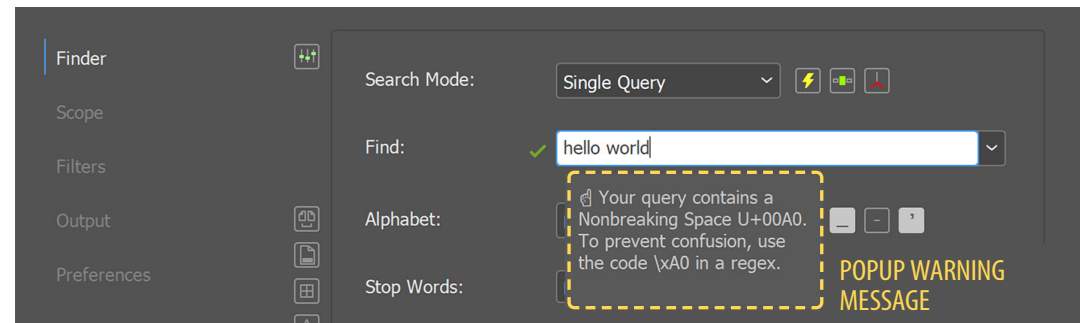
If the Quick Matches button or another search command cannot retrieve any match, the console displays a “No match found” message:



In such an event,

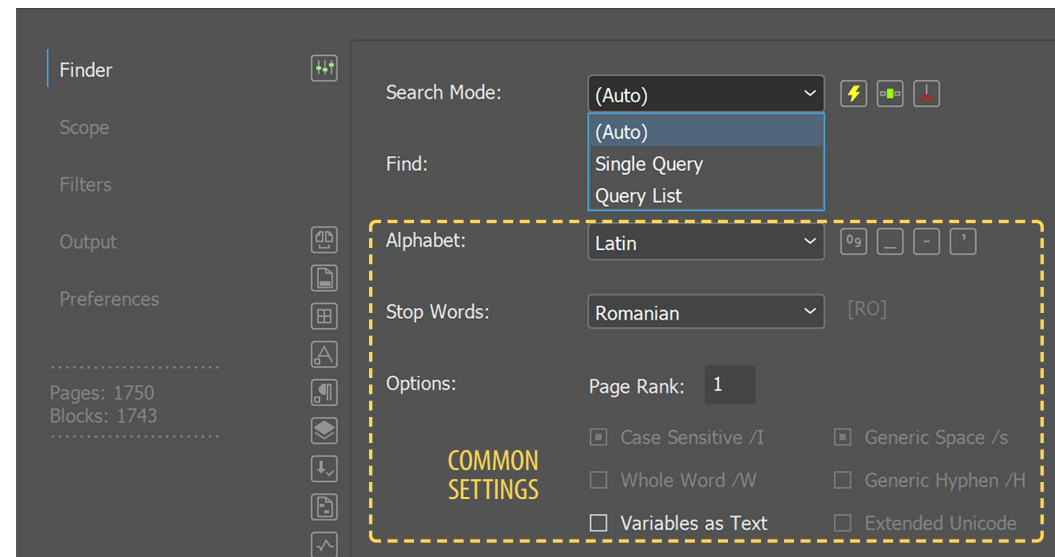
- Make sure your query is properly written; e.g. you might have omitted the leading slash / of a regex query.
- Check that no undesired flags are active in the indicator area; e.g. you might have left a page range or a style filter active, which is no longer satisfied.
- Review your search options, especially the *case sensitivity* and *whole word* flags (introduced in → [Query Options](#)).
- Beware of invisible characters that may pollute your query.

A well-known issue is the presence of nonbreaking spaces or similar special characters resulting from a copy/paste operation. IndexMatic<sup>3</sup> will then display a popup message warning you that the Find field contains a character you might not even know is there.



## 4. Common Search Settings

Below the Search Mode interface is a set of settings shared between all modes. No matter what you ask the search engine, these parameters determine how to interpret any query regarding the alphabet in use, case sensitivity and other aspects.



# Finder and Basic Queries

## ► *Alphabet*

**Finder** ► **Alphabet** lets you define the writing system that your queries want to address. This is a crucial setting, as it decides which Unicode characters will be identified as word letters. The selected ALPHABET directly influences the behavior of automatic mode, regex queries (at least, most of them) and even literal queries if *Whole Word* is active.

**NOTE** In IndexMatic<sup>3</sup>, the effect of metacharacters like `\w`, `\l` or `\m` entirely depends on the selected alphabet.

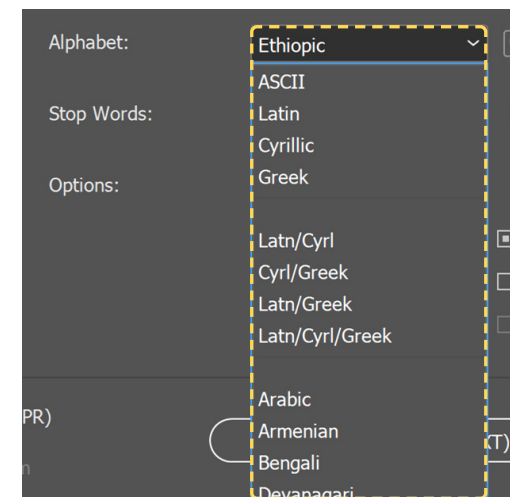
The below table lists the supported alphabets, including mixed sets like *Latn/Cyrl*, *Cyrl/Greek*, etc. On scanning a new document,

IndexMatic<sup>3</sup> tries to guess the alphabet that fits best the text. If it fails, select a different entry in the drop-down list.

In addition, the switches to the right of the **Alphabet** allows you to forcibly append special characters to the set of letters:

- 09 Digits (0123456789)
- \_ Underscore: U+005F LOW LINE
- Hyphens
- ' Apostrophe

By default, these switches are turned off.



ALPHABET	RELATED UNICODE BLOCKS
<b>ASCII</b>	26 ASCII letters: [a-zA-Z]
<b>Latin</b>	Basic Latin, Latin-1 Supplement + Latin Extended blocks.
<b>Cyrillic</b>	Cyrillic, Cyrillic Supplement + Cyrillic Extended blocks.
<b>Greek</b>	Greek and Coptic + Greek Extended blocks.
<b>Latn/Cyrl</b>	Letters of <b>Latin</b> and <b>Cyrillic</b>
<b>Cyrl/Greek</b>	Letters of <b>Cyrillic</b> and <b>Greek</b>
<b>Latn/Greek</b>	Letters of <b>Latin</b> and <b>Greek</b>
<b>Latn/Cyrl/Greek</b>	Letters of <b>Latin</b> , <b>Cyrillic</b> and <b>Greek</b>
<b>Arabic</b>	Arabic, Arabic Supplement + Arabic Extended blocks + Presentation Forms

ALPHABET	RELATED UNICODE BLOCKS
<b>Armenian</b>	U+0530..U+058F
<b>Bengali</b>	U+0980..U+09FF
<b>Devanagari</b>	Devanagari + Devanagari Extended
<b>Ethiopic</b>	Ethiopic, Ethiopic Supplement + Ethiopic Extended blocks
<b>Georgian</b>	Georgian, Georgian Supplement, Georgian Extended
<b>Gujarati</b>	U+0A80..U+0AFF
<b>Hebrew</b>	U+0590..U+05FF
<b>Kannada</b>	U+0C80..U+0CFF
<b>Khmer</b>	U+1780..U+17FF

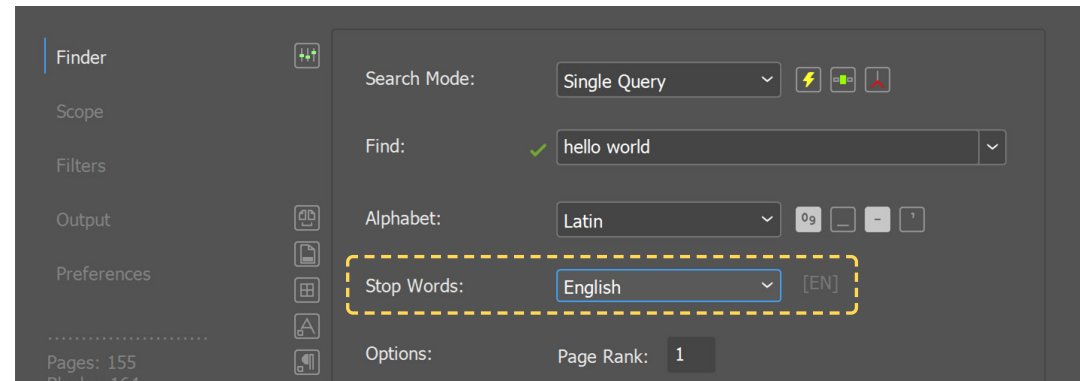
ALPHABET	RELATED UNICODE BLOCKS
<b>Lao</b>	U+0E80..U+0EFF
<b>Malayalam</b>	U+0D00..U+0D7F
<b>Myanmar</b>	Myanmar + Myanmar Extended blocks
<b>Oriya</b>	U+0B00..U+0B7F
<b>Sinhala</b>	U+0D80..U+0DFF
<b>Tamil</b>	Tamil + Tamil Supplement
<b>Telugu</b>	U+0C00..U+0C7F
<b>Thai</b>	U+0E00..U+0E7F
<b>Tibetan</b>	U+0F00..U+0FFF

# Finder and Basic Queries



- With the ASCII alphabet, turn on and to return the metacharacter `\w` to its usual behavior in regular expressions. This also helps you capture alphanumeric codes in *Automatic* search mode.
- Turn on to treat as alphabetic letters the characters U+002D HYPHEN-MINUS, U+2010 HYPHEN and U+2011 NON-BREAKING HYPHEN. With this setting, the *Automatic* search mode and the `/\w+/` regex will also capture compound words like “*mother-in-law*”, or prefixes/suffixes like “*meta-*” or “*-ing*”.
- Turn on to treat as alphabetic letters the characters:
  - U+0027 APOSTROPHE,
  - U+2019 RIGHT SINGLE QUOTATION MARK,
  - U+055A ARMENIAN APOSTROPHE,
  - U+059C HEBREW ACCENT GERESH,
  - U+05F3 HEBREW PUNCTUATION GERESH,
  - U+FF07 FULLWIDTH APOSTROPHE.

**NOTE** In most languages it is irrelevant to load in the alphabet, as the apostrophe can hardly be equated with a letter in the sense of “being part of a word”. However, this option is useful in some circumstances. Apostrophes are found in names (d’Artagnan, O’Neill) and this structure is common in French. In Breton, the trigram *c’h* is a letter in its own right. Also, you may sometimes need to analyze elided forms as lexical units.



► Stop Words (also called “*insignificant words*”, or “*noise words*”)

**DEFN** In computing, stop words are words which are filtered out before or after processing of natural language data.

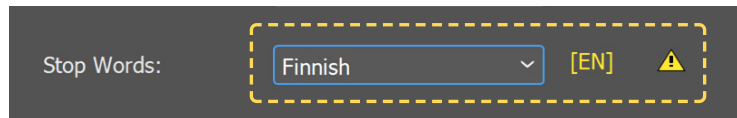
**Finder** ► **Stop Words** provides, with respect to the **Alphabet**, a list of languages. Each entry then refers to a predefined set of **STOP WORDS**. These are filtered out after processing index entries, so this option does not affect matches: any captured match remains visible in the Quick Matches view, even if it is a stop word.

**NOTE** IndexMatic<sup>3</sup> internally stores stop words for 50+ languages. Depending on the active alphabet, only the relevant languages are shown. To disable this feature, select **Finder** ► **Stop Words** ► *[None]*.

Since there is an implicit link between the **Stop Words** language and the language selected in **Output** ► **Sorting**, the ISO code of the latter—e.g. [EN], [FR], etc—is displayed next to the **Stop Words** field so you can check their consistency. If the two entries


Noise words can be automatically removed while processing index entries in English (*all, an, and, any, are, been, can, did, do, does, each, for, go, has, he, her, if, in, is, it, make, must, of, off, or, self, she, should, than, that, the, there, they, to, under, very, want, was, were, what, when, which, who, will, with, would, you, your, and many more*). IndexMatic<sup>3</sup> detects about **1400 stop words** in that language.

# Finder and Basic Queries



do not match (as in the figure) the ISO code appears in yellow and a warning symbol is shown. To fix this, do one of the following:

- Select a **Stop Words** language that is consistent with the sorting language.
- Click the ISO code (in yellow) to activate the **Output** ▶ **Sorting** panel and then adjust the sorting language.
- Ctrl-Click the ISO code to automatically adjust the sorting language without leaving the **Finder**.

**UPD.** Editable stop word lists are supported from v. 3.23052. Select the option *[Custom]* and click the  button to edit your list.

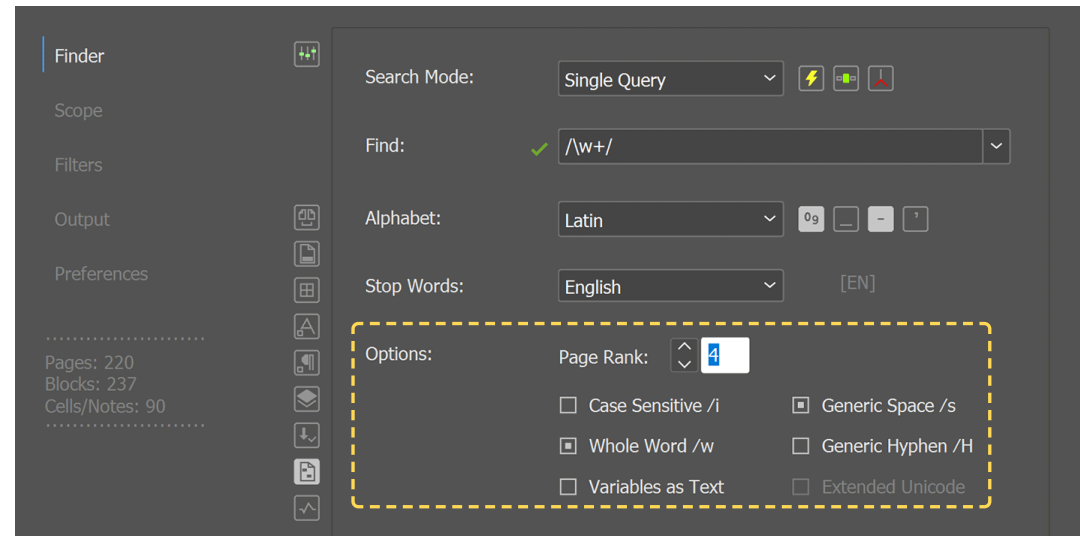
from  
version  
3.23052



## ▶ Query Options

**Finder** ▶ **Options** is a special area that more finely controls the behavior of your queries. All settings here are referred to as **DEFAULT** options since a particular query may have parameters (known as **FLAGS**) that override this or that option.

**NOTE** It may happen that a default option is temporarily grayed out because another setting (usually at the query level) supersedes

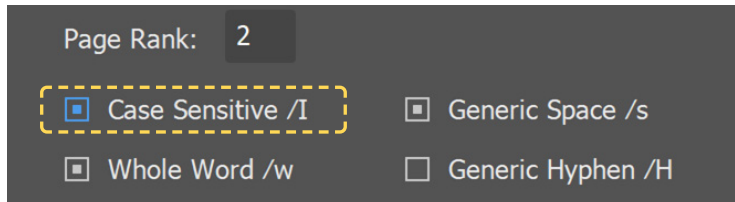


the default attribute. In *(Auto)* search mode, most default options are inhibited since they are given an automatic value.

- **Page Rank.** An integer between 1 and 9 (inclusive). This parameter does not affect matches, it sets the following condition on the final entries: a term will be reported for a given page locator *if and only if* the underlying matches were captured at least *N* times on that page (*N* being the Page Rank). The transparent value is obviously 1.

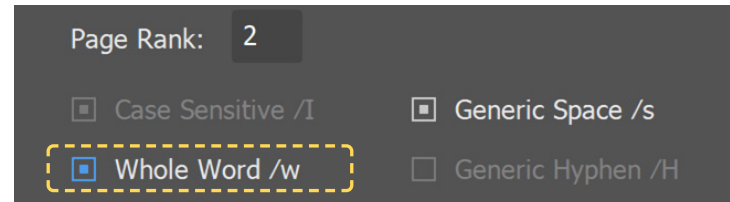
**TIP** Increasing the **Page Rank** is the easiest way to limit the number of index entries by selecting only the most notable occurrences of a term. The assumed rule is that an expression which is repeated often on a page is likely to be semantically important at that location. This is not always the case, but once stop words has been removed, the page rank tends to become quite a relevant measure.

# Finder and Basic Queries



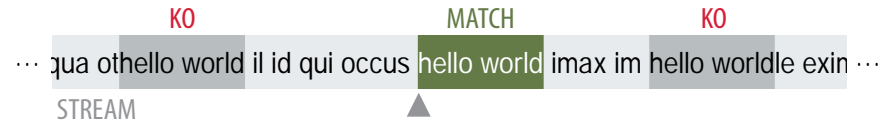
- **Case Sensitivity** /I (on) /i (off)  
 A query is case sensitive if it discriminates between upper-case and lowercase letters, so all letters in the match are required exactly as they are typed in the query: `hello world` will strictly capture “*hello world*”, not “*Hello World*” or “*Hello WORLD*”. Turn this option off to relax case sensitivity: `hello world` will then capture any variation of the expression.

**NOTE** As soon as a query is case insensitive, it is no longer true that a match necessarily coincides with the original character string expressed in that query. Case variants like “hello world” and “Hello World” could then be considered distinct index entries, although both captured from the same command. This eventuality is taken care of in [Output](#) ► [Sorting](#) ► [Merge](#).



- **Whole Word** /w (on) /W (off)  
 A whole-word query requires two conditions to be met:
  - The first character of a match cannot be preceded by a letter of the active alphabet;
  - The last character of a match cannot be followed by a letter of the active alphabet.

In the below example, only the middle expression is a valid Whole-Word match of `hello world`:



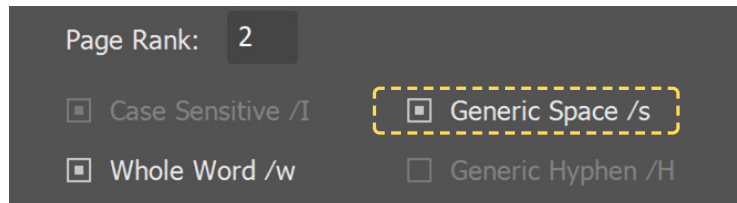
**NOTE** As such, a whole-word query does not put any particular condition on the internal characters of the match, so it can retrieve simple words as well as more complicate expressions, even an entire sentence.

**NOTE** By design the *(Auto)* search mode will only capture whole-word matches.

The symbols /I vs. /i, /w vs. /W, etc, are just reminders reflecting the syntax of FLAGS in advanced queries.  
 → [Adding Local Flags](#)



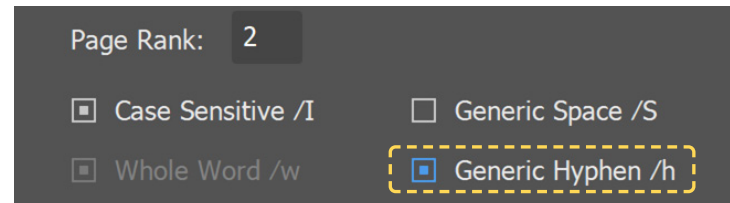
# Finder and Basic Queries



- **Generic Space (GS)** /s (on) /S (off)  
A Generic-Space query interprets a usual space (in the query) as denoting any white space (in the matches). The query can then grab unexpected variants of the space character, in particular nonbreaking spaces, tabs, en spaces, etc. Turn this option on to guarantee that `hello world` will capture all “`hello world`” forms whatever the nature of the inner space. Turn it off to exclusively capture regular space characters (U+0020).

**NOTE** The key idea behind the *GS* option is to save you from entering regular expressions when your queries only have to eliminate pointless space disparities. The usual space then works as a wildcard.

**TIP** In IndexMatic<sup>3</sup>, you can fully customize the set of white spaces recognized by the Generic Space option. Go into [Preferences](#) ► [Generic Spaces](#). → [Fine-Tuning Generic Spaces](#)



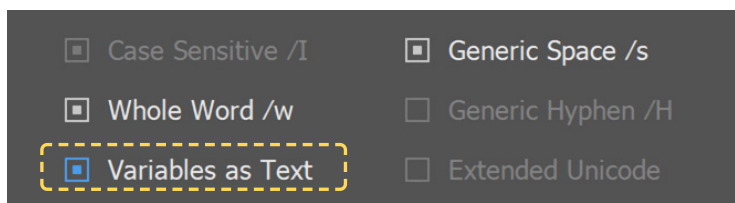
- **Generic Hyphen (GH)** /h (on) /H (off)  
A Generic-Hyphen query interprets an ASCII hyphen (in the query) as denoting any hyphen or dash character (in the matches). The query can then grab unexpected variants of the hyphen, in particular nonbreaking hyphen, Unicode hyphen, en dash, etc. Turn this option on to guarantee that `hello-world` will capture all “`hello world`” forms whatever the nature of the inner hyphen. Turn it off to exclusively capture U+002D HYPHEN-MINUS.

**⚠** The discretionary hyphen (U+00AD SOFT HYPHEN) is never captured, since IndexMatic<sup>3</sup> automatically removes it before processing queries.

**NOTE** The key idea behind the *GH* option is to save you from entering regular expressions when your queries only have to eliminate pointless hyphen disparities. The ASCII hyphen then works as a wildcard.

**TIP** In IndexMatic<sup>3</sup>, you can fully customize the set of characters recognized by the Generic Hyphen option. Go into [Preferences](#) ► [Generic Hyphens](#). → [Fine-Tuning Generic Hyphens](#)

# Finder and Basic Queries

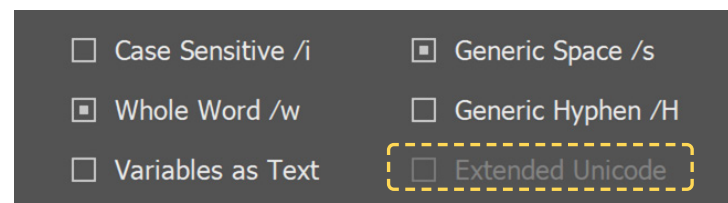


- **Variables as Text**

**DEFN** A text variable is an item you insert in your document that varies according to the context. For example, the Last Page Number variable displays the page number of the last page of the document.

Unlike other query options, *Variables as Text* acts globally on the incoming stream: when turned on, it translates every variable instance into a searchable text. For example, if your scope contains a custom text variable `Topic1` whose value is “*speculative reason*”, this string is normally undetectable because InDesign embeds the variable instance in a special character (U+0018). Yet, instead of dealing with that obscure marker, you might prefer that IndexMatic<sup>3</sup> captures the text itself, “*speculative reason*”, as a true part of the stream. This is the role of the *Variables as Text* option.

**NOTE** ⚠ If *Variables as Text* is enabled, the special character U+0018 (i.e., the variable marker) becomes itself undetectable from the query engine—since it is dynamically changed into the underlying text. Hence make sure your queries are no longer expecting to find that special character!




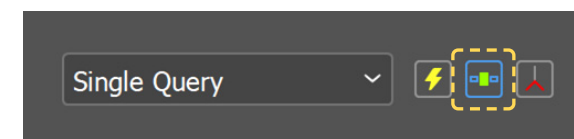
- **Extended Unicode**

⚠ This feature is not implemented in the present version of IndexMatic<sup>3</sup>.

## 5. Matches in Context (Concordancing)

In whatever **Search Mode**, you can preview and study matches *in their context*. This functionality is useful in content analysis. It also makes it possible to check the internal consistency of keywords, punctuation marks, etc., within the scoped documents.

- 1) Activate the **Finder** and select your **Search Mode**.
- 2) Once your query is set (and/or the underlying parameters), click the Matches-in-Context button .



- 3) The console opens and displays a few samples associated to a lowercase term [in brackets] and a number prefixed by ×. The former represents a typical match (called a **NODE**) and the latter indicates the number of occurrences that have been found under

# Finder and Basic Queries

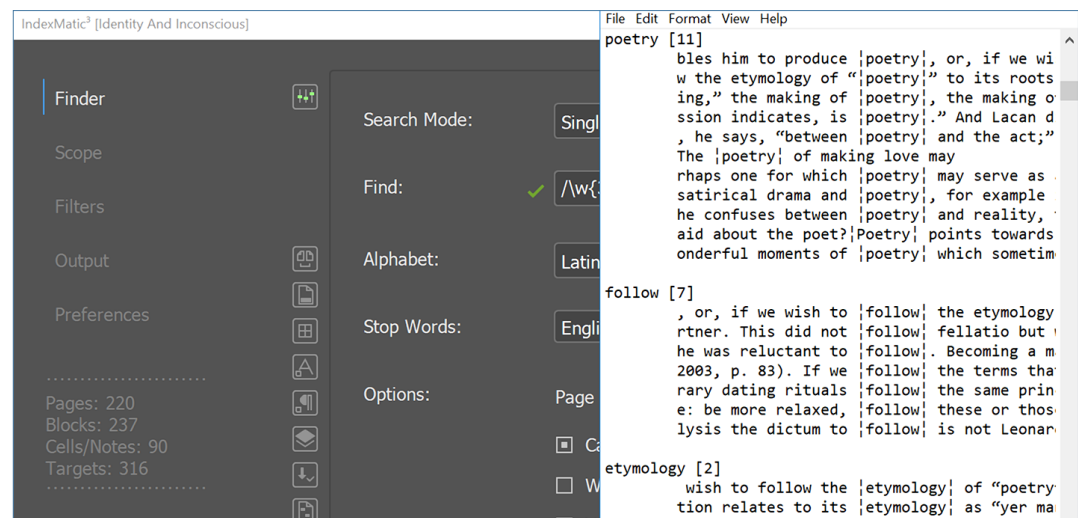
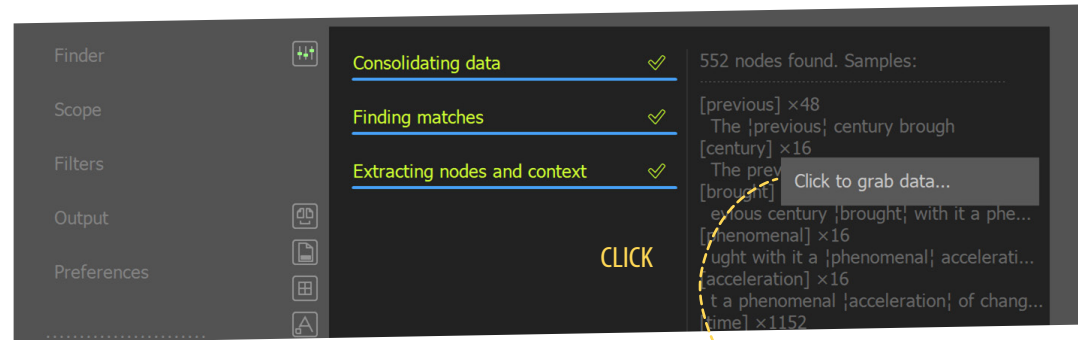
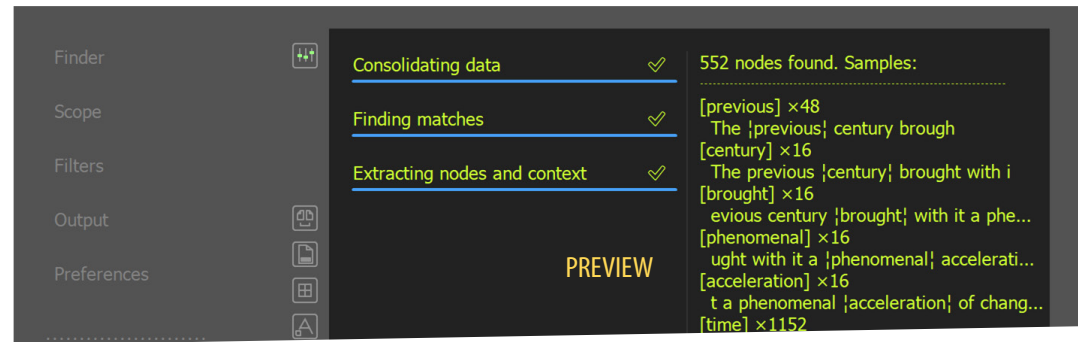


that node. As for the samples, each represents a particular match, surrounded by broken bars `|...|`, in its neighborhood.

- 4) Move the mouse over the sample area (the message “Click to grab data” pops up) and click. The complete concordance report is then loaded in a plain text file and shown in your default text editor.

Seeing matches in context involves at least three tasks, “Consolidating data”, “Finding matches”, and “Extracting nodes and context”. The first task prepares the text clusters (taken from the scope) to be scanned. Although it may be time-consuming with huge documents, it is optimized to run quickly on subsequent calls, even if you have significantly changed your scope settings or filters in the meantime. “Extracting nodes...” may involve creating a large file (depending on the dataset) but it should run fast on average.

**TIP** You can increase or reduce the context length, or force the contextual nodes to exclude stop words, from [Preferences](#) ► [General](#) ► [Context](#). → [Changing Basic Preferences](#)



# Finder and Basic Queries



## 6. Managing Markers and Special Characters

InDesign's text markers and invisible characters may complicate the extraction of text data. Sometimes you do not want to encounter them at all (as if they were not even present); sometimes you expect to detect them from a query because they provide clues for identifying neighboring items.

IndexMatic<sup>3</sup> offers a dialog that controls the behavior of these special characters one by one. Depending on the element under consideration, at most five actions are possible:

- *Keep* (K) Leave the character unchanged in the stream.
- *Break* (B) Parse it as a paragraph break.
- *Remove* (R) Remove it (as if it never existed).
- *Space* (S) Parse it as a simple space (U+0020).
- *ZWNJ* (Z) Parse it as a ZERO WIDTH NON-JOINER character (U+200C).

These actions are performed *upstream*, that is, before the query engine comes into play. The default configuration should fit your needs in most cases: Forced line breaks and footnote/endnote references are kept, End nested style here and Indent to here characters are removed, Table markers are treated as paragraph breaks, etc.

The table below lists all markers and special characters addressed by IndexMatic<sup>3</sup>; it indicates the default behavior (DEFAULT) and the possible ACTIONS that can be executed. For some characters

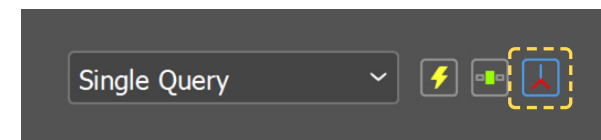
the default action is locked . For example, ends of paragraph and column/frame/page breaks will remain LINE BREAK characters to IndexMatic<sup>3</sup>, you cannot change that.

**NOTE** A few obscure character names (followed by an asterisk) are mentioned in the table. These are undocumented elements that may exist anyway in an InDesign document.

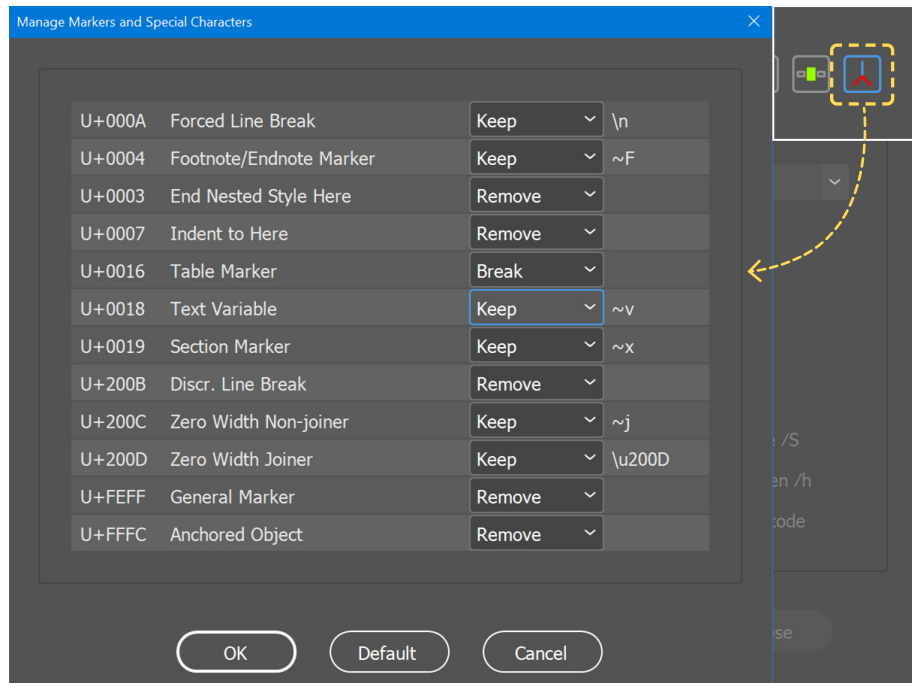
In order to customize the actions associated to markers and special characters:

- 1) Activate the **Finder**.
- 2) Click the Manage Markers button . A dedicated window shows up.
- 3) For each marker or character that requires customization, click the actions drop-down list and select the desired entry among *Keep*, *Break*, *Remove*, *Space*, *ZWNJ*.
- 4) Click **OK** to confirm your choices.

INDESIGN CHARACTER	DEFAULT	ACTIONS
Forced Line Break	U+000A Keep	K B S
Footnote/Endnote Marker	U+0004 Keep	K R S Z
End Nested Style Here	U+0003 Remove	K R S Z
Indent To Here	U+0007 Remove	K R S Z
Table Marker	U+0016 Break	K B R S Z
Table Continued	U+0017 Remove	
Text Variable (any)	U+0018 Keep	K R S Z
Section Marker	U+0019 Keep	K R S Z
Discretionary Line Break	U+200B Remove	K R S Z
Zero Width Non Joiner	U+200C Keep	K R S
Zero Width Joiner	U+200D Keep	K R S Z
End of Paragraph / Breaks	U+000D Break	
Discretionary Hyphen	U+00AD Remove	
Word Joiner*	U+2060 Remove	
Invisible Separator*	U+2063 ZWNJ	
General Marker	U+FEFF Remove	K B R S Z
Anchored Object	U+FFFC Remove	K B R S Z
Special Glyph*	U+FFFD Space	



# Finder and Basic Queries



If needed, click the **Default** button to restore the set of actions as predefined by IndexMatic<sup>3</sup>.

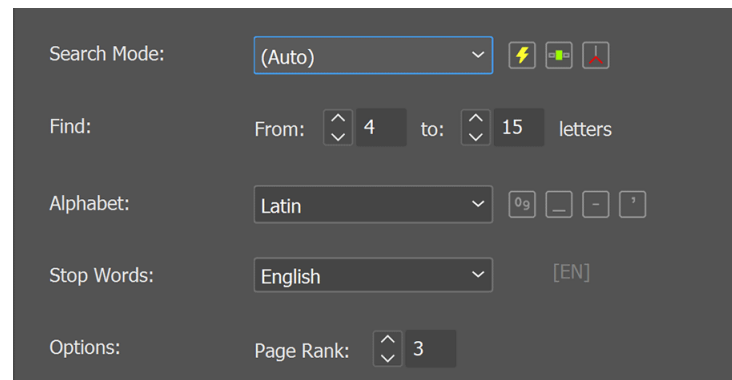
**TIP** The **Manage Markers and Special Characters** dialog gives useful information on each element: in the first column, the codepoint of the character; in the last column, the code you may use in a regex query for capturing that element—provided it is kept as is.

**NOTE** ⚠ A warning symbol related to the Text Variable marker (U+0018) will appear if that element cannot be addressed due to Variables as Text being turned on.

## 7. Automatic Word Extraction

The automatic search mode is the easiest way to scan the scoped document(s) and generate indexing terms from scratch. However, this method should be used only for roughing out work—or as a last resort when you don't have time to fine-tune your index!

- 1) Activate the **Finder**.
- 2) Select **Search Mode: (Auto)**.
- 3) In **Find From** (resp. **to**), enter the minimum (resp. maximum) number of characters that a word can contain. The word lengths range from 1 to 48.
- 4) Adjust **Alphabet**, **Stop Words**, and **Page Rank** to your needs. A page rank higher than 1 is recommended to avoid a flood of insignificant terms.



# Finder and Basic Queries



5) Run the desired command (⚡, 📄, Hits, Index...)

**NOTE** *Search Mode (Auto)* strictly captures single words without space. It implicitly sends the query engine a regex of the form `/\w{min,max}/w`, where *min* and *max* denote the minimum and maximum lengths. The whole-word flag (`/w`) is forced and other basic options are inhibited.

**UPD.** In some languages like English or French, the *(Auto)* mode also improves **stop words** filtering when the active alphabet includes hyphens and/or apostrophes: noisy substrings like "...s" or "let'..." are then detected and removed.

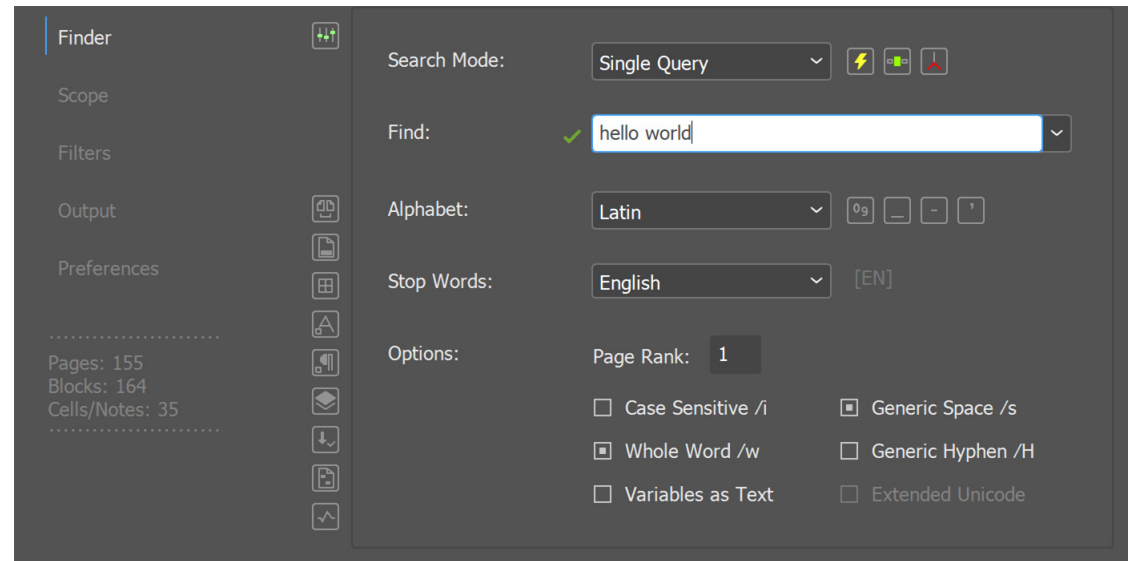
from  
version  
3.23042

**TIP** If your indexing terms are delimited by a character style (or a set of character styles) do not use the automatic search mode. Instead, go into **Filters** ▶ **Character Styles**, define the target style(s) and run a *Single Query* of the form `/.+/.`

## 8. Running Single Queries

The *Single Query* mode is ideal for testing simple expressions as well as regex queries. Sometimes, a well-crafted single query is even sufficient to produce an entire index.

- 1) Activate the **Finder**.
- 2) Select **Search Mode: Single Query**.



- 3) Type your query in the **Find** edit box.
- 4) Adjust **Alphabet**, **Stop Words**, and other search **Options**.
- 5) Run the desired command (⚡, 📄, Hits, Index...)

**NOTE** Regular expressions (i.e. regex queries) must start with a slash character (`/`). Their syntax and applications are detailed in → [Advanced Queries](#).

- TIP**
- Turn *Case Sensitive* off to make your query capture case variants.
  - Turn *Whole Word* off to let your query capture word fragments.
  - If your query contains spaces and/or hyphens, consider to turn on the *Generic Space* (resp. *Generic Hyphen*) option.



# Finder and Basic Queries



## 9. Saving/Restoring Favorite Queries

In **Finder** ► **Search Mode** ► *Single Query*, you have access to the three most recent valid queries and to a custom set of FAVORITE QUERIES (also called *favorites*).

► To restore a recent or favorite query: →

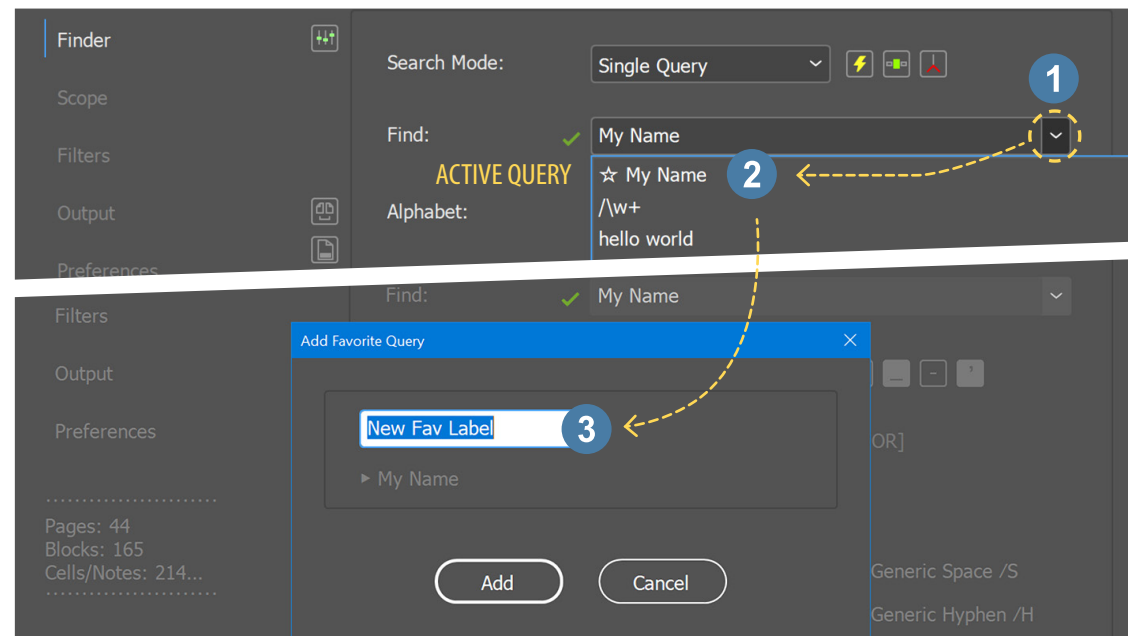
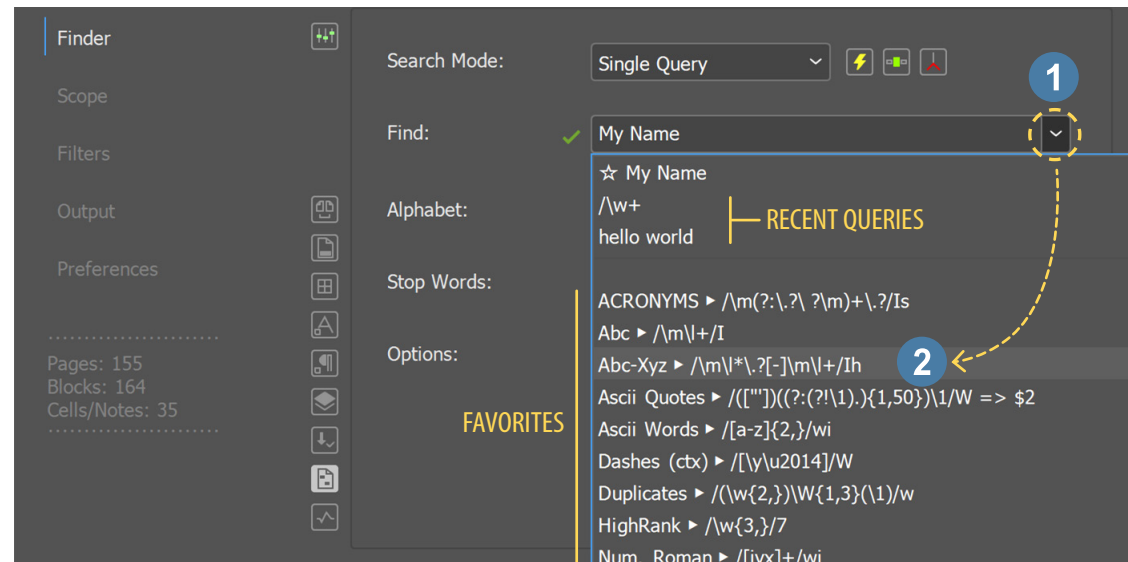
- Click the down arrow ① to the right of the Find edit box.
- In the drop-down list, click the saved query ② that you want to restore: it is loaded in the Find field.

**NOTE** Recent queries are at the beginning of the list, while favorite queries are sorted by name in UTF16 order.

► To save the active query as a favorite: →

- Click the down arrow ① to the right of the Find edit box.
- Click the active query ② at the top of the drop-down list. The star ☆ indicates that the query can be registered as a new favorite.
- In the Add Favorite Query dialog, enter a custom label ③ (naming your query) and click **Add**.

**TIP** Instead of steps ① and ②, you can simply RIGHT CLICK the Find field.



# Finder and Basic Queries

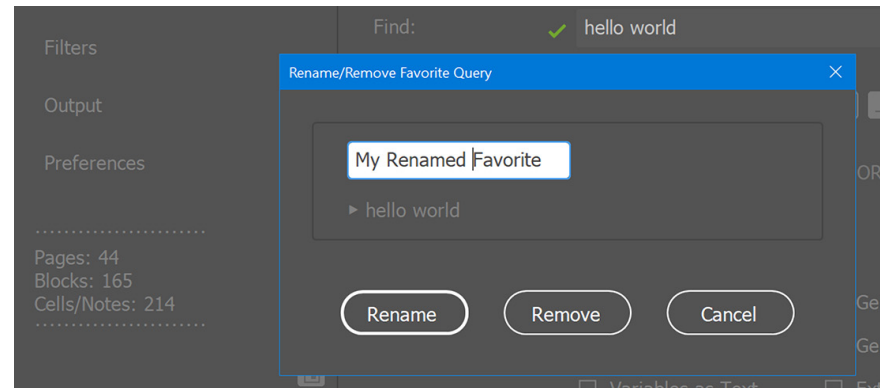


► To rename a favorite or remove it from the list:



- Make it the active query (see the restore stage above).
- Right-click the Find field, or click the first entry in the favorite drop-down list. *Note:* The star ★ indicates that the query is already registered as a favorite.
- In the Rename/Remove Favorite Query dialog, either change the existing label and click **Rename**; or click the **Remove** button.

**NOTE** IndexMatic<sup>3</sup> comes with a set of predefined favorites that capture acronyms, basic and compound proper names, duplicates, Roman numerals, numbers, punctuation marks, etc.



## 10. About Query Lists (QL)

A QUERY LIST (QL) is a simple plain text file, saved on your disk, where each non-empty line specifies a distinct query. Unlike IndexMatic<sup>2</sup>, IndexMatic<sup>3</sup> no longer handles query lists from an internal edit box, so you have total freedom to edit those files using your own, default text editor.

**NOTE** There were serious limitations in using an integrated editor. You can now transparently edit and manage QL files of unlimited length, outside of InDesign.

Unless your system is configured otherwise, query lists will open in TextEdit (macOS) or NotePad (Windows). If you change that, make sure that no formatting or special encoding is applied to the file: IndexMatic<sup>3</sup> only recognizes UTF8 text-encoding files, using the regular, OS-dependent *newline* sequence.


# Finder and Basic Queries



## 11. Loading a Query List


As long as no query list is associated to your project, you cannot send commands from the *Query List* mode.

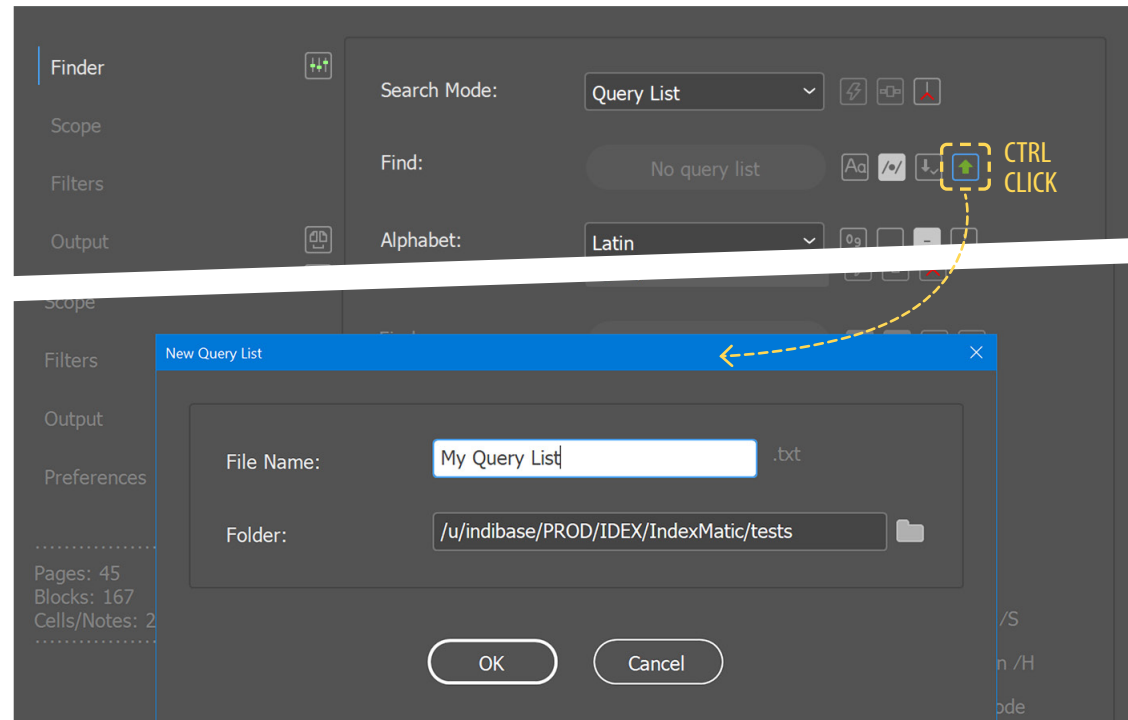
► To create a new QL:

- 1) Activate the **Finder**.
- 2) Select **Search Mode: Query List**.
- 3) Ctrl-Click the Load button .
- 4) In the **New Query List** window, enter a file name and select a destination folder, then click **OK**.

**NOTE** On Windows platforms, you can also create a new text file by simply clicking the Load button and using the system interface, which enables file creation as well.

► To load an existing QL:

- 1) Activate the **Finder**.
- 2) Select **Search Mode: Query List**.
- 3) Click the Load button  and select the file using the system interface.



Once a query list is loaded, it remains linked to your project (unless externally removed). Of course you can load another file whenever needed.

IndexMatic<sup>3</sup> runs your default text editor so you can edit the QL dynamically, append new queries, etc, and save changes. When you go back to the main dialog, any modification (in the file) is detected and the interface updates accordingly.

**NOTE** ⚠ Do not forget to save the QL file from your text editor before returning to IndexMatic<sup>3</sup>.

# Finder and Basic Queries



IndexMatic<sup>3</sup> parses the active QL file and displays the number of valid queries in the COUNT BUTTON (see screenshot).

A green checkmark ✓ indicates that all queries are valid. If syntax errors are detected, a warning symbol ⚠ and a popup message inform you that some queries cannot be validated.

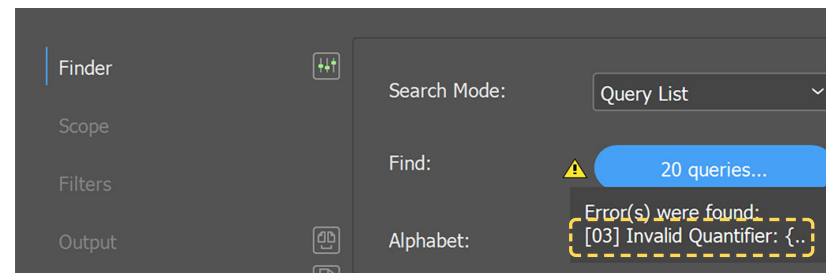
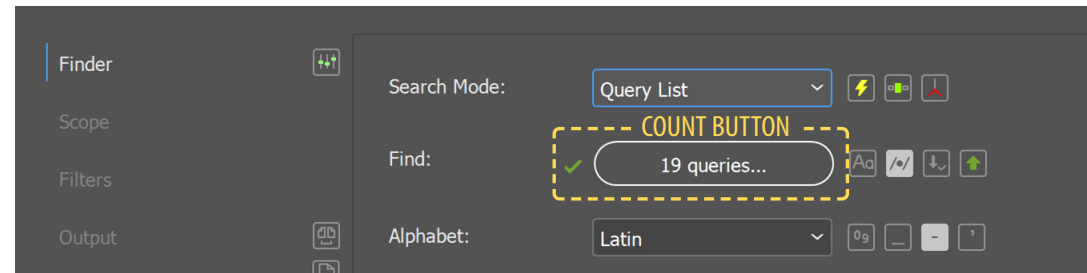
**TIP** If you accidentally closed your text editor, just click the count button to reopen it, with the active QL loaded.

## 12. Submitting a Word List as a Query List

The *Query List* mode allows you to send at once a list of queries. Although it offers powerful extra features (which we will cover in → [Advanced Queries](#)) this mode is already perfect for processing a simple WORD LIST which can be any length.

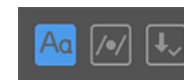
Before you go on, make sure your list has one word or expression per line, with no extraneous characters. Should an expression contain a slash character (/), replace it by the escape sequence \/. Save the list in a text file or have it available in the clipboard.

- 1) Activate the **Finder**.
- 2) Select **Search Mode: Query List**.
- 3) Load or create the QL file as detailed previously. You can either load your word list as is, or paste its content in a fresh query list.



If syntax errors were found in a query list, the popup message shows in brackets the line number of the first error and a short description.

- 4) Adjust other search options to your needs. Typically, **Stop Words: [None]** and **Page Rank: 2** or 3. Configure the specific Query List switches as follows:



- 5) Run the desired command (⚡, =, Hits, Index...)

## 13. Query List Settings

Query lists have special switches visible in the **Find** area, to the right of the count button. These settings only affect QL processing and have no effect in the other search modes:

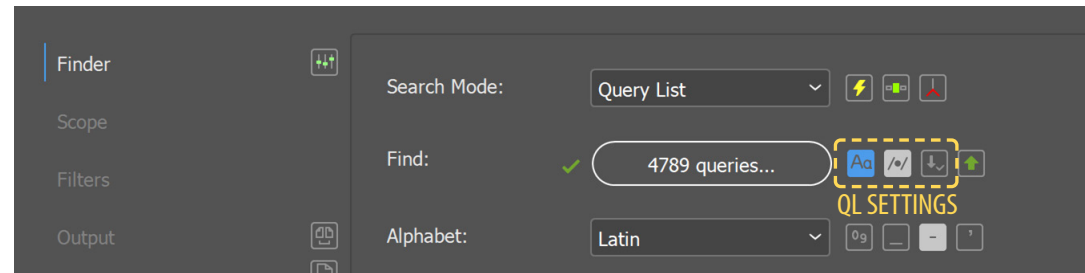
### ► *Magic Case Sensitivity*

Turn on this option to let IndexMatic<sup>3</sup> determine automatically the case sensitivity of **TOKEN QUERIES** (i. e., those based on a literal expression). A query with no local flag **/i** or **/I** is made case sensitive if it contains at least an uppercase letter (e.g. **Buffalo**, **POTUS**); otherwise it is considered case insensitive.

**TIP** When you prepare a word list, write common nouns in lowercase and proper nouns with initial capitals. You can then use Magic Case Sensitivity to enhance your index. If a lowercase form must still be captured case-sensitively, add the suffix **/I**. E.g., **bill/I** will only find lowercase matches (not “Bill”).

### ► *Magic Regex Term*

This option only applies to **REGEX QUERIES** (i. e., those based on a regular expression). When the **OUTPUT** of a query is not specified—like in **/dogs?** or **/Bah[íi]a Blanca/Is**—the engine tries to determine automatically the most natural entry (**dog**, **Bahía Blanca**) by removing optional characters (**s?**) and/or by selecting the first element of a class: **[íi] → í**.




**NOTE** This feature makes sense when you execute advanced queries. As long as you use word lists or token queries, it has no effect.

### ► *Keep Original Order*

With this option turned on, IndexMatic<sup>3</sup> entirely bypasses sorting instructions (**Output** ► **Sorting**) and preserves the original order of your entries in the QL.

*Important:* **Adding Alphabetic Headings** won't apply either.

**NOTE**  If **Keep Original Order** is active, it is assumed that there exists in your query list a one-to-one correspondance between **KEYS** and **TOPICS** (which is the case with a simple word list). IndexMatic<sup>3</sup> will then arrange the entries with respect to the custom order possessed by the incoming queries. But if a particular query produces multiple results, as would do **/\w{5,}**, the index entries within that group will remain unsorted.

# Advanced Queries

So far we've assumed for simplicity that a QUERY is some expression that just captures MATCHES. Well, it can do much more! An IndexMatic<sup>3</sup> query is a full command that not only recognizes certain patterns in the scope, but can also transform or completely rewrite the TOPIC that your readers will see in the final index. Also, a query can deal with subtopics, and sub-subtopics..., and/or specify CROSS-REFERENCES. This chapter will walk you through the detailed syntax of all these commands.

## 1. General Scheme

What makes IndexMatic<sup>3</sup> highly flexible is its query interpreter. At a very first step, the program splits every incoming query in two parts: the INPUT side (to the left) and the OUTPUT side (to the right).

*INPUT => OUTPUT*

In the middle of this scheme is an OPERATOR formed of two characters: the equals sign = and the greater-than sign >. These characters must remain stuck (=>) for the operator to be recognized.

Most of the time you do not use the *INPUT => OUTPUT* scheme, although it still rules your query at an implicit level. *INPUT* represents what is looked for (in the scope), *OUTPUT* represents what has to be printed in the index.

For example, when a query is supplied as a simple token—say `castle`—the system assumes that `castle` is both the searched expression and the final heading, so the query is actually interpreted as:

```
castle => castle
         -----
```

The left side is referred to as the KEY (in this case, a TOKEN); the right side is referred to as the TOPIC structure. So, in the original form of the query (`castle`), the key is explicit and the topic is implicit (that is, implicitly identified to the key).

But on various occasions this default mechanism is unsuitable. Suppose you search a last name, `Gödel`, and need to output a custom topic: "`Gödel, K. (1906-1978)`". Then you will specify that explicit term in your query using the => operator:

```
Gödel => Gödel, K. (1906-1978)
```

**TIP** Spaces around the => operator are optional. They just make your queries easier to read.

Also, you may wish to pool together distinct matches ("`castle`", "`manor`", "`fort`") under a unique topic (`castle`). How to do so? The first solution that comes to mind is to write a query list:

```
castle => castle
manor => castle
fort => castle
```

In the examples below, we underline in dotted line the implicit part of the query as it is finally understood by the interpreter.



# Advanced Queries



The three queries above, based on different keys, will indeed converge into a single topic. Note that a query like `manor => castle` may yield the topic `castle` without that expression ever occurring in the document (since this query only looks for the key `manor`). This illustrates that the general syntax

`KEY => TOPIC`

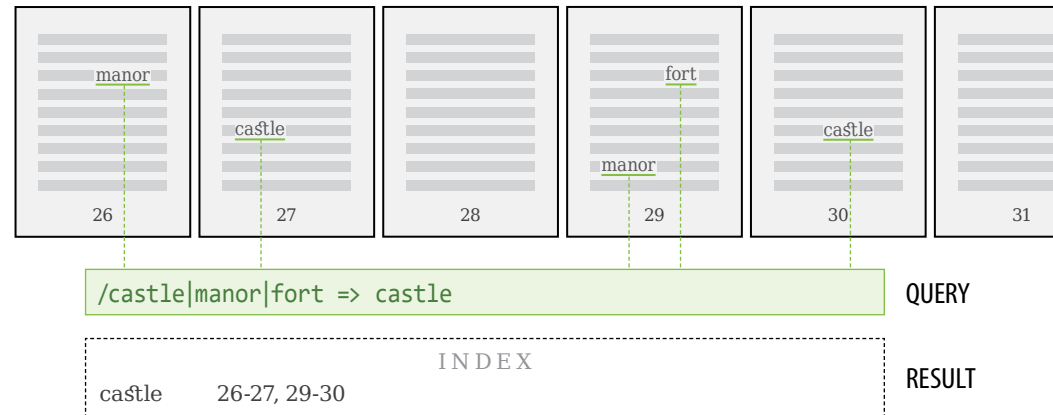
only considers the *KEY* part for searching and puts no condition on the *TOPIC* part. You can tell IndexMatic<sup>3</sup> to rewrite or convert a key into any topic of your choice. This policy is very useful when the target document contains clue words—e.g. “*empiricism*”, “*stoicism*”...—that infer the concept you actually want to index (e.g. *schools of thought*, or *philosophy*).

A shorter way of grouping “*castle*”, “*manor*”, “*fort*” under a single leading topic would be:

```
/castle|manor|fort => castle
```

Here we use a regular expression (a *REGEX* key) that captures any occurrence of either “*castle*”, “*manor*”, or “*fort*”, and link all encountered matches to the topic *castle* (see figure).

**NOTE** Any regex query (i. e., any query based on a regex key) must start with a slash (/). An ending slash may be added to terminate the regex pattern, e.g. `/castle|manor|fort/ => castle`, but that ending slash remains optional as long as the query does not involve local flags.



## 2. Adding Local Flags

**DEFN** A local flag is a setting, encoded by a special character, that overrides a default search option at the query level.

The *KEY* side of a query supports an optional extension, starting with a slash, that specifies a set of *LOCAL FLAGS* (see table below). Here are a few examples:

- The key `castle/I` indicates that the token `castle` must be searched case-sensitively, even if case sensitivity is turned off in **Finder ▶ Options**.
- The key `hello world/iS` both deactivates case sensitivity and generic space for the token `hello world`.
- The key `/colour?r/w4` activates the whole-word option and set the local page rank to 4 for the regex `/colour?r`.

FLAG	EFFECT
1 to 9	Sets the Page Rank (for that particular query).
i	Case Insensitive.
I	Case Sensitive.
w	Whole Word ON.
W	Whole Word OFF.
s	Generic Space ON.
S	Generic Space OFF.
h	Generic Hyphen ON
H	Generic Hyphen OFF
!	Inhibits Stop Word filtering

from version 3.23042

The regex `/colour?r` makes the letter ‘u’ optional, so it both captures “colour” and “color”. → [Introduction to Regular Expressions](#)

# Advanced Queries



from version 3.23042

- The query `/wh\w+!` locally allows the collection of **stop words** (“who”, “which”, “when”...) via a regular expression.

**NOTE** You can mix local flags in whatever order: `/colou?r/w4I` is strictly equivalent to `/colou?r/4Iw`.

Local flags have no impact on determining the implied topic of a query. For example,

`castle/i2`

is interpreted:

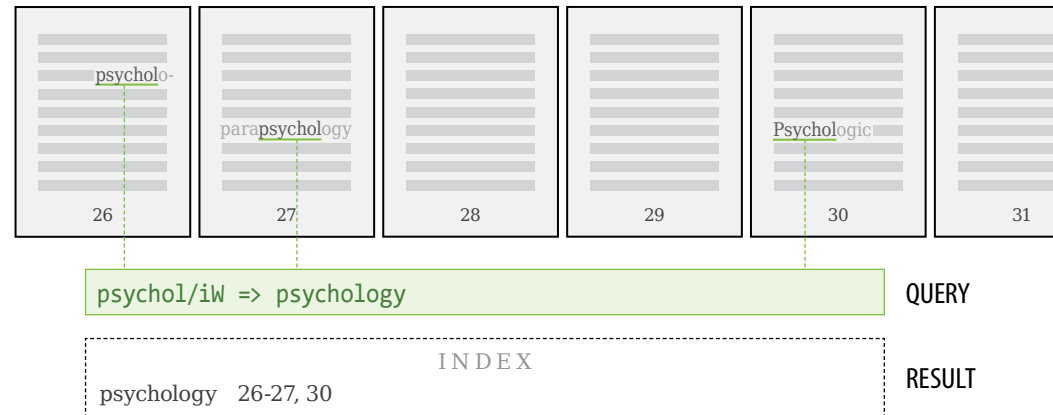
`castle/i2 => castle`

Depending on the default options defined in the **Finder** at a given time, a set of local flags can be redundant. If case sensitivity is globally turned off, the flag `/i` only repeats that very option. But this feature may become a tool in re-usable query lists, as it makes the behavior of fine-tuned commands independent of the default options. For example, suppose you’ve determined that the partial token “*psychol*” is a sufficient basis for the query

`psychol/iW => psychology`

in all the documents you’ve planned to index. Encoding the local flags `/iW` at the query level preserves the effectiveness of this command regardless of IndexMatic<sup>3</sup> settings.

**NOTE** In the above example, providing an explicit topic seems unavoidable. If we were keeping the key `psychol/iW` alone, the engine would attach it to the term ‘*psychol*’.



## 3. Complete Syntax of a Key-Topic Query

Let’s summarize what we learned so far:

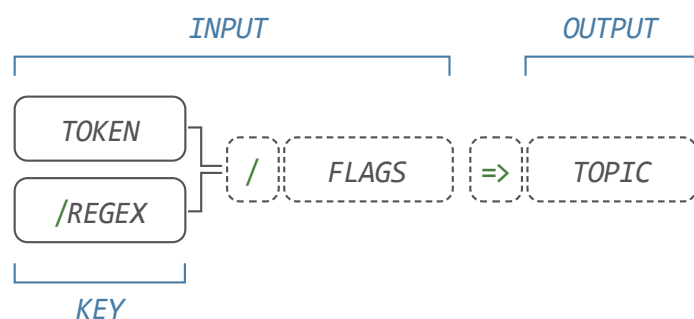
- The key is the only required element in a `KEY => TOPIC` query. If missing, the `=> TOPIC` part is silently added, the implicit topic being made equal to the key.
- In simple cases, the key is just a literal expression, we call it a token. If you don’t use special operators, any string will act as a token and form a complete query.
- You can also supply a regular expression (regex). To do so, insert a slash (/) at the beginning of the pattern, e.g. `/dog|cat|snake` (an optional ending slash is allowed).
- Depending on your settings, a key may be case sensitive, support generic space, whole-word, etc.

IndexMatic<sup>2</sup> users may detect a slight change in terminology compared to the original manual. We now use the word “topic” (sometimes “topic structure”) to designate the output part of a query—where we previously used the word “term”. The two ideas remain roughly equivalent, but we now reserve the word “term” to denote the last node of a hierarchical topic structure (that is, when additional parent topics are involved). With that meaning, a term may also be referred to as the “target topic” of a query. My warm thanks to **Peter Kahrel** who helped me clarify these concepts.

# Advanced Queries

- You can add a */FLAGS* suffix to a key—where *FLAGS* denotes any combination of local flags *i, I, w, W, 1, 2*, etc—in order to override the default search options.

This leads to the general syntax of a key-topic query:



(Dotted blocks indicate optional elements.)

**TIP** Since the symbol */* has a special meaning on the *KEY* side, you might have to use the escape sequence *\/* to specify this character itself. E.g. *\/path\/to* will capture the string “/path/to” in the document.

## 4. Adding a Parent Topic

The rightmost part of the above scheme usually defines the text to be finally rendered, e.g. *stoicism*, as a single topic. But you might want to add a *PARENT TOPIC* over that expression so your index can address hierarchical entries like:

```
schools of thought 33-42 (1)
  empiricism 19, 35-36, 75 (2)
  stoicism 41, 182 (3)
```

In this example, *schools of thought* is the parent topic of the topics *empiricism* and *stoicism*. Those are indented relative to the parent topic and form a compact group.

Note, however, that each line is an *INDEX ENTRY* on its own (as it contains page locators). In line (1), *schools of thought* is a topic without parent topic. In lines (2) and (3), *empiricism* and *stoicism* are children of the topic *schools of thought*.

Thanks to the special operator, *>* (greater-than), you can declare parent topics within any query. Here is the minimal scheme:

```
KEY => PARENT_TOPIC > TARGET_TOPIC
e.g. stoicism/Wi => schools of thought > stoicism
```

where the *TARGET TOPIC* (also known as the *TERM* of the query) determines the actual entry associated to the key, while *PARENT TOPIC* simply declares a hierarchical parent.

# Advanced Queries



- TIP** 1. Spaces around the `>` operator are optional but make your queries easier to read.
2. Since the symbol `>` has a special meaning on the *TOPIC* side, you may have to use the escape sequence `\>` to specify this character itself. E.g., `... => x \> y` will yield the term “*x > y*”.

This new operator just extends the basic syntax of a topic structure. Note that the *PARENT > CHILD* scheme both creates a target topic (*CHILD*) and declares its parent topic, although no particular command is introduced at this parent level. That is, no instruction is provided on capturing specific matches or associating specific pages to the parent topic. The query

```
stoicism/wi => schools of thought > stoicism
```

is all about *stoicism* as an index entry and will not attach extra data to the topic *schools of thought*. It may produce, for example,

```
schools of thought
stoicism 41, 182
```

where the first line (without page numbers) only has the function of a hierarchical heading. If you want to treat that topic as a fully-fledged *TERM*, a specific query is still required for it to be *targeted*, e.g.

```
/schools?/w => schools of thought
```

**NOTE** A parent topic is not necessarily unique relative to a particular child, but you need then to provide a distinct query. E.g.

```
stoicism => schools of thought > stoicism
stoicism => philosophy > stoicism
```

Unless the keys and concepts referred to are radically different, this approach is not recommended though. Better is to declare a cross-reference, as we will explain later.

## 5. Nested Topics

Any topic can itself have a parent topic, which in turn can have a parent topic, and so on. In other words, the topic structure is recursive and allows you to build a precise taxonomy of subjects:

```
KEY => TOPIC
KEY => TOPIC > TOPIC
KEY => TOPIC > TOPIC > TOPIC
etc.
```

For example,

```
/cats? => Mammalia > Carnivora > Felidae > cat
```

declares three nested parent topics for the target topic *cat*. In your index a typical result would be:

**REM.** In this manual, the LAST NODE of the topic structure is also referred to as the “target topic” (or “term”).

# Advanced Queries

```
Mammalia
  Carnivora
    Felidae
      cat 37-41, 72, 105
```

each topic having its specific indentation level. Other queries can then create sibling terms, parallel topics and/or add instructions for the parent topics:

QUERY LIST	<pre>carnivor/Wi =&gt; Mammalia &gt; Carnivora /cats? =&gt; Mammalia &gt; Carnivora &gt; Felidae &gt; cat /dogs? =&gt; Mammalia &gt; Carnivora &gt; Canidae &gt; dog</pre>
RESULT	<pre style="text-align: center;">INDEX</pre> <pre>Mammalia   Carnivora 15-31, 40, 52     Canidae       dog 17, 51-53, 97, 105-106     Felidae       cat 37-41, 72, 105</pre>

**NOTE** Unlike IndexMatic<sup>2</sup>, IndexMatic<sup>3</sup> supports subtopics of arbitrary depth, although it will not create more than 9 indentation levels while rendering the index.

When you create multi-level topics in a query list, it is your responsibility to maintain the overall consistency (spelling, case, levels) of the different headings. IndexMatic<sup>3</sup> cannot guess that two topics that slightly differ should in fact denote the same conceptual element.

## 6. \$-Variables

Inherently, a regex captures different expressions in the text. Hence, when processing a regex query, IndexMatic<sup>3</sup> assumes that each new expression should yield a separate topic, as long as no explicit term is supplied.

Of course, if the query provides a formal topic, as in

```
/dog|cat|lion/w => animal
```

then all matches are aggregated under that term so their distinct forms vanish—all occurrences of the words “dog”, “cat”, “lion” are hidden by the final index entry (*animal*).

But the very same query, without explicit topic,

```
/dog|cat|lion/w
```

will likely detect those three forms at different locations (assuming they are all present in the scope). And more generic regexes, like `/\w+/`, will produce hundreds or thousands of implicit topics at once. Maybe your final index already!

Sometimes that’s exactly what you want. Sometimes you need better control of the matched expressions in order to craft suitable terms, add parent topic(s), etc.

# Advanced Queries

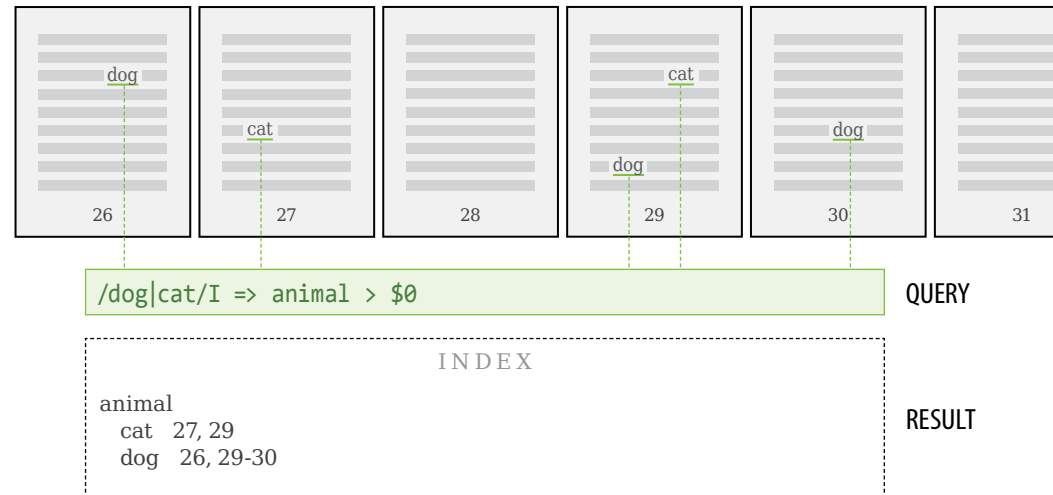


On the *TOPIC* side, the interpreter recognizes special placeholders (called *-\$-VARIABLES*) prefixed with the symbol *\$*. They represent a string with respect to the following table:

VARIABLE	MEANING
<i>\$</i>	Stands for the key as written in the query, ignoring flags and removing escape characters in regex. This variable is both available in regex and token queries. <i>Examples:</i> <code>dog/i =&gt; animal &gt; \$</code> ↔ <code>dog/i =&gt; animal &gt; dog</code> <code>/a\w+ / =&gt; \$</code> ↔ <code>/a\w+ / =&gt; a\w+</code> <code>/a\b\x63 / =&gt; \$</code> ↔ <code>/a\b\x63 / =&gt; a/bc</code>
<i>\$0</i>	Represents the <u>entire match</u> of a regex query, that is, any particular expression it actually captures in the text. (In token queries, <i>\$0</i> is equivalent to <i>\$</i> ). <i>Example:</i> <code>/dog cat/I =&gt; animal &gt; \$0</code> creates the output <code>animal &gt; dog</code> when “dog” is found, and <code>animal &gt; cat</code> when “cat” is found.
<i>\$1 to \$9</i>	Those variables represent any <u>matched substring</u> that results from a capturing parenthesis in the regex. <i>\$n</i> serves as the <i>n</i> th capture in the regular expression (counting left parentheses). <i>Example:</i> <code>/hello (World Bob)/wI =&gt; \$1</code> generates the entries <code>World</code> and <code>Bob</code> for the distinct matches.

In summary, *\$* stands for the key, *\$0* stands for the entire match, and *\$1*, *\$2*, etc, represent parts of that match—keeping in mind that matches and submatches will likely take different forms while the query is running.

**NOTE** Observe that *\$0* still reflects the implicit target topic(s) of a query. So we can now rewrite the auto-completion rule of the interpreter as follows: `KEY => $0`



*-\$-variables* are essential tools for refining topics while optimizing complex query lists:

- They combine naturally with literal parts: `... => hello $2!`
- They can declare parent topics as well: `... => $1 > $0`
- They allow you to rearrange submatches in a different order: `... => $2 $1`
- They support **Topic Formatters**: `... => |$0|`

**NOTE** You will be more comfortable using *-\$-variables* when getting familiar with regex queries.

This simple regex query shows how distinct topics (“cat”, “dog”) are generated from a single command. They emanate from the *\$0* variable, which receives any new captured match. In addition, a parent topic (“animal”) is declared in the output part of the query. It controls how index entries are finally indented.



## 7. Topic Formatters

FORMATTERS are special operators available on the *TOPIC* side. They allow you to select, simplify or change the form or the case of some expressions (mostly variable elements).

### ► Trim Formatter `|expr|`

The *trim* function removes the white spaces that could surround a string: “ *hello world* ” → “*hello world*”. To trim an expression *expr*, use `|expr|`. For example, the code `||$1|-ing` applies the *trim* function to *\$1* and concatenates the result with *-ing*.

**TIP** If the vertical bar `|` is required as is, use the escape sequence `||` (i.e. two vertical bars).

This operator is useful to clean a compound topic structure of undesired spaces that might be captured in a *\$*-variable.

### ► Uppercase Formatter `^expr^`

The *uppercase* function converts any lowercase letter into its uppercase counterpart, leaving other characters unchanged: “*Hello World*” → “*HELLO WORLD*”. To produce the uppercase form of the expression *expr*, use `^expr^`. For example, if *\$1* contains “*abc*” the code `^$1^` will yield *ABC*.

**TIP** If the character `^` is required as is, use the escape sequence `^^`.

### ► Lowercase Formatter `~expr~`

The *lowercase* function converts any uppercase letter into its lowercase counterpart, leaving other characters unchanged: “*Hello World*” → “*hello world*”. To produce the lowercase form of the expression *expr*, use `~expr~`. For example, if *\$1* contains “*ABC*” the code `~$1~` will yield *abc*.

**TIP** If the character `~` is required as is, use the escape sequence `~~`.

### ► Title case Formatter `^expr~`

The *title case* function applies uppercase to the first character and lowercase to the next characters of a string: “*hello World*” → “*Hello world*”. To produce the title case form of the expression *expr*, use `^expr~`. For example, if *\$1* contains “*abc*” the code `^$1~` will yield *Abc*.

### ► Ternary Operator `{expr1?expr2:expr3}`

The ternary operator is a conditional structure that involves three expressions: *expr1*, *expr2*, *expr3*. It yields *expr2* if *expr1* is non-empty, *expr3* otherwise. The syntax is `{expr1?expr2:expr3}` and could read: “if *expr1* then *expr2*, else *expr3*.”

This operator is useful when you have to output different results depending on a variable being set or not. Suppose that a regex captures a sequence of digits in *\$1*, and optionally a sequence of two extra digits in *\$2*. As *\$2* is optional it may be empty. You

# Advanced Queries

want to apply a consistent format, #.##, to all numeric topics. You can then write the command as follows:

```
... => {${2?}$1.$2:$1.00}
```

If \$2 is set (i.e., non-empty), the pattern \$1.\$2 is selected and yields the expected concatenation. Otherwise, the pattern \$1.00 is used so “.00” is added to the number stored in \$1.

**TIP** All symbols required by the ternary operator, { ? : and }, can be escaped using the usual duplicate sequence: {{ ?? :: and }}.

A few recommendations:

- Use the ternary operator wisely. As a recent extension of the query language, it only supports basic *if-then-else* commands.
- Do not embed a ternary operator within another one.
- Do not add cosmetic spaces around the { ? : } symbols, they would be preserved as part of the expression(s).
- If other topic formatters are needed, make sure they don't conflict with the { \_? \_ : \_ } pattern.

Any expression in the pattern can be empty. For example, { \$1?a: } outputs “a” if \$1 is set, otherwise an empty string. This particular structure can be abbreviated { \$1?a } (omitting the

: part). Also, the scheme { \$1? \$1:b }—which yields \$1 if non-empty, “b” otherwise—can be abbreviated { \$1:b } (omitting the ? part).

SHORTCUT	FULL SYNTAX	MEANING
{X?Y}	{X?Y:}	if X then Y else <i>nothing</i>
{X:Y}	{X?X:Y}	if X then X else Y

## 8. Topic Structure Cheatsheet

OPERATOR	FUNCTION	EXAMPLE	ESCAPE SEQ.
=>	Explicit topic.	dog => animal	=\>
>	Topic separator.	dog => animal > dog	\>
\$	Repeats the key.	dog => animal > \$	\>
\$0	Full match.	/dog cat/ => \$0	
\$1, \$2...	Nth submatch.	/(dog cat)s?/ => \$1	
·	Trim.	/(abc *)def/ =>   \$1	
^·^	Uppercase.	/ab./i => ^\$0^	^^
~·~	Lowercase.	/ab./i => ~\$0~	~~
^·~	Title case.	/ab./i => ^\$0~	^^...~~
{·?·:·}	Ternary operator.	/(a)(b?)c/ => { \$2? \$1 : \$0 }	{{ ?? :: }}

## 9. Introduction to Regular Expressions

**DEFN** A regular expression (regex) is a pattern used to match character combinations in strings.

If (and only if) the key of an IndexMatic<sup>3</sup> query starts with a single slash /, it is parsed as a regular expression and the query is referred to as a **REGEX QUERY**. The output of a regex query can be either implicit, as in

```
/dog|cat|lion
```

or explicit, as in

```
/dog|cat|lion => animal > $0
```

The present section specifically focuses on the key side—keeping in mind that the implicit topic of a regex query is **\$0**.

Unlike other interpreters, IndexMatic<sup>3</sup> does not require a regex to end with a slash (unless you need to specify flags), but as the form `/. . ./` is highly conventional you can always add this termination slash without side effects. Hence, the key `/dog|cat|lion` is equivalent to `/dog|cat|lion/`.

Strictly speaking, the actual **PATTERN** of the above regex is `dog|cat|lion`, since the enclosing slashes are just markup characters indicating the very presence of a regular expression. For

that reason, if you were to specify a slash inside the pattern, you would have to **ESCAPE** it using the code `\/` (backslash + slash).

**NOTE** Escaping special characters is one of the first things to digest when learning regular expressions. Luckily, all escape sequences are based on the same rule: insert a backslash before a special character to inhibit its behavior and get it specified literally.

Patterns are mixtures of simple and special characters. Simple characters, like `a é æ §`, only refer to themselves (case-insensitively if the flag `/i` is active). Special characters, like `. + * | ( ) [ ]` and some others, perform a specific task.

► *Alternatives* `X|Y|Z|...`

In `dog|cat|lion`, the pipe `|` is the *alternation* operator. Each element is called an **ALTERNATIVE** and the regex will match any of them (in our example, either “*dog*”, “*cat*”, or “*lion*”).

When IndexMatic<sup>3</sup> runs the query `/dog|cat|lion/`, it looks for any alternative throughout the entire scope and capture all resulting matches. Some are occurrences of “*dog*”, which yields the corresponding index entry—*dog*—and the associated locations; others are occurrences of “*cat*”, etc.

In other words, the single query

```
/dog|cat|lion/
```

# Advanced Queries



does “in one pass” the job of the query list

dog  
cat  
lion

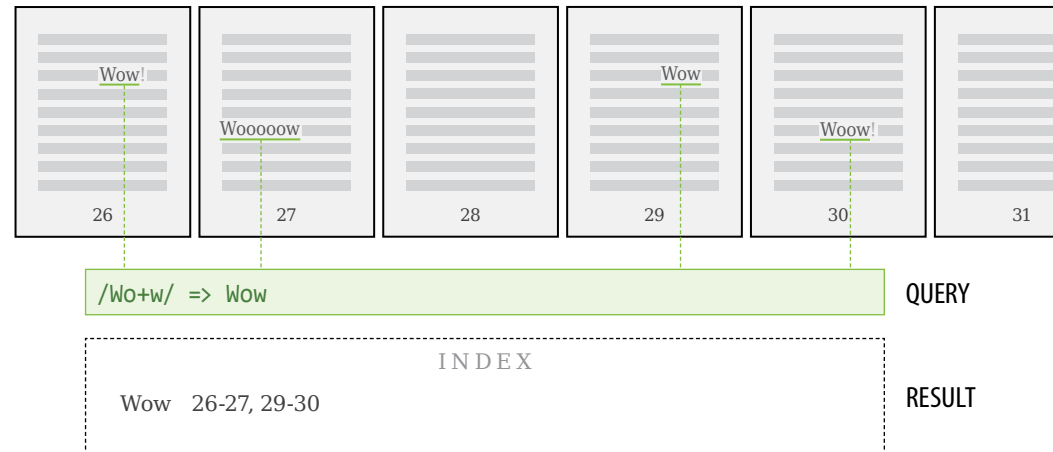
► *Quantifiers*  $X^+ X? X^* X\{m,n\}$

QUANTIFIERS deal with optional and/or repeated elements. Suppose that your document contains occurrences of “Wow”, “Woow”, “Wooww”... You want to capture all these expressions from a single query and reduce them to the topic *Wow*. Use the **+** quantifier to specify one or more “o”:

`/Wo+w/ => Wow`

All quantifiers are postfix operators that define the allowed number of occurrences of the element they follow:

SYNTAX	MEANING
$X^+$	One or more $X$ .
$X?$	Zero or one $X$ . (So $X$ is optional.)
$X^*$	Zero, one, or more $X$ . (So $X$ is optional too.)
$X\{n,p\}$	At least $n$ and at most $p$ $X$ ( $n$ and $p$ being integers).
$X\{n\}$	Exactly $n$ $X$ . Shortcut of $X\{n,n\}$ .
$X\{n,\}$	At least $n$ $X$ .



**NOTE** In the table above,  $X$  denotes a simple character, a character class [...] or a group (...). More on these in the next pages.

Since the quantifier **?** makes an element optional, it is widely used (in English, French, and other languages) to unify the singular and the plural form of a noun whose plural is formed by adding “-s”:

`/dogs?/ => dog`

To extend the optional part to e.g. two consecutive letters, just group the characters in enclosing parentheses:

`/kiss(es)?/ => kiss`

**NOTE** Because ( and ) are special operators, you would have to escape them, \ ( and \ ), if your goal was to detect actual parentheses.

# Advanced Queries



The quantifier `*` (*zero-or-more*) is typically used for controlling trailing spaces and various combinations of punctuation marks whose sequence is not exactly known. In:

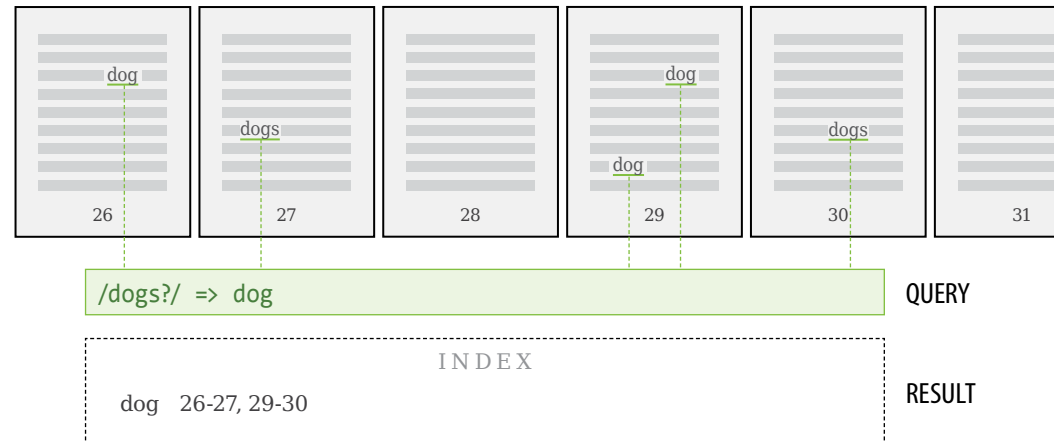
```
/Ingredients *:/sI => menu
```

our basic target is “*Ingredients:*” but random spaces may sometimes precede the colon and you don’t want to miss those occurrences. With the help of the Generic Space flag (`/s`), the sub-pattern  `*` (space followed by `*`) will detect every optional sequence of white spaces, of any kind and any length.

Since quantifiers may eat a given element zero, one, or more times, you could rightfully ask: *how many instances will they actually capture?* And that’s not a rhetorical question! Consider the regex `/dogs?s/`. Will it capture the match “*dogs*”, “*dogss*”, or both? The answer is both.

- In the former case the optional `s?` is not consumed, as it must be ignored to fulfill the match “*dogs*”.
- In the latter case, the optional `s?` is consumed, as it must be captured to fulfill the match “*dogss*”.

The regex processor is therefore forced to consume the right number of quantified elements, as long as there is a strong constraint on the following elements in the pattern. But let’s remove the final `s`. Now, why should the regex `/dogs?/` capture “*dogs*” rather than “*dog*”? After all, since the `s` is optional and ends



the pattern, the processor could be happy with finding the substring “*dog*”—say in the text “*dogs are great*”—without looking any further.

However, when several options exist, the *how-many-instances* problem is subject to a rule: by default, all quantifiers are GREEDY. That is, they consume as many elements as allowed, provided that the entire pattern still matches.

**NOTE** In the `/dogs?s/` example, the processor had to reject the optional `s` in order to match “*dogs*”. But if it had the choice, the processor would have consumed it.

In rare cases that we won’t illustrate here, you might need NON-GREEDY quantifiers instead, i.e. operators that promote the shorter match when multiple solutions are available. To make a quantifier non-greedy, just add a `?` after it (*see table below*).

GREEDY	NON-GREEDY
<code>X+</code>	<code>X+?</code>
<code>X?</code>	<code>X??</code>
<code>X*</code>	<code>X*?</code>
<code>X{n,p}</code>	<code>X{n,p}?</code>
<code>X{n}</code>	n/a
<code>X{n,}</code>	<code>X{n,}?</code>

# Advanced Queries

## ► Grouping (...) (?:...)

GROUPS are parts of a pattern enclosed in parentheses. The processor will treat a group as a single element with respect to quantifiers or other commands. For example,

```
/kiss(es)?/
```

groups the two letters `es` to let the symbol `?` operates on the whole sequence. Another example is

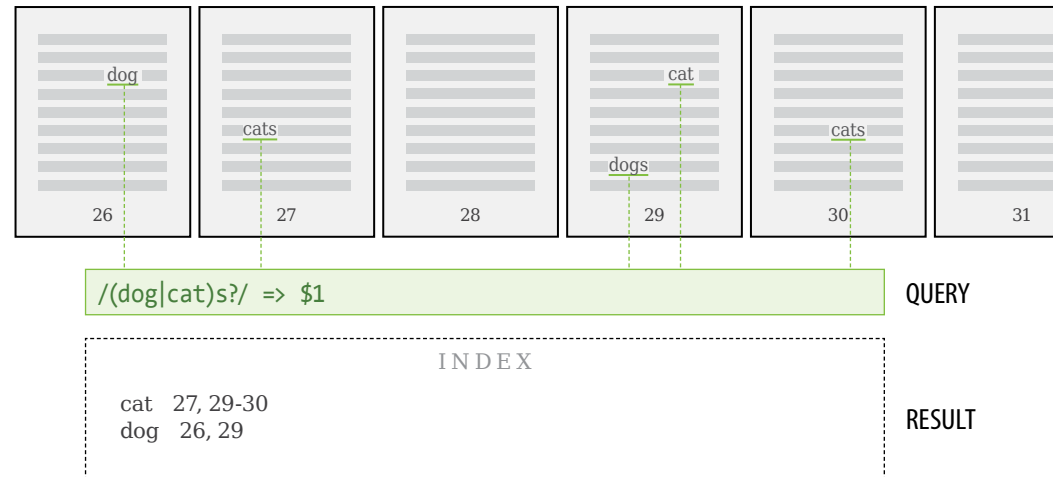
```
/stor(y|ies)/
```

which both captures the forms “*story*” and “*stories*” by somehow factoring the common prefix, `stor`, and adding the alternative `y|ies`.

**NOTE** In the absence of groups, the alternation operator `|` always acts on complete expressions before and after it, including other operators, because it has the “lowest precedence” in the regex syntax. Even concatenation is of higher order, so `ab|cd` is interpreted `(ab)|(cd)`, not `a(b|c)d`.

Groups aren’t just for building variations around a key pattern, they also capture submatches that can be recovered using the `$1`, `$2`... variables. This is the default behavior of a group (also known as a CAPTURING GROUP). For example,

```
/(dog|cat)s?/ => $1
```



is a powerful query that not only captures every possible match (“*dog*”, “*dogs*”, “*cat*”, “*cats*”), but also yields the correct term, singular, for each animal. Indeed, `$1` refers to the submatch emanating from `(dog|cat)`—i. e. “*dog*” or “*cat*”—so it is not affected by the optional `s`.

In this example, `$1` offers a much better solution than `$0`. The latter would have generated four topics: *cat*, *cats*, *dog*, *dogs*.

Capturing groups are left-to-right ordered according to the position of the opening parentheses (also referred to as CAPTURING PARENTHESES):

```
/(abc)xyz(def)uvw(ghi)etc/
```

\$1
\$2
\$3

Nested capturing groups are allowed as well:

```
/(abc)xyz(de(f|g))uvw(hij)etc/
```

\$1
\$2
\$3
\$4

← `$2` contains the submatch `(de(f|g))`, which itself contains `$3 = (f|g)`.



# Advanced Queries

Apart from the full-match variable `$0`, IndexMatic<sup>3</sup> supports nine submatch variables (`$1` to `$9`). You will hardly ever need all these, although you may occasionally build a pattern involving *many* groups that don't require to capture submatches. In order to limit the number of useless capturing groups, you will then use a NON-CAPTURING parenthesis (`?:` instead of `(`). The syntax

```
/stor(?:y|ies)/
```

provides exactly the functionality of a group without making it capturing, so the variable `$1` is not cluttered with uninteresting data and remains free for other uses, as in:

```
/stor(?:y|ies) of (Hope|Job)/
                |
                $1
```

**TIP** Using non-capturing groups wherever submatches are not needed is considered a good practice. Even if this makes the regex a little less readable, this tends to reduce the execution time of your query.

► Character Classes `[...]` `[^...]`

A character CLASS is a set of alternative characters, and specifically this very set enclosed in square brackets when talking about comprehensive classes. The most obvious example of a comprehensive class is `[0123456789]`, which captures any single digit.

As an IndexMatic<sup>3</sup> user you will take advantage of classes on many occasions. A class matches one character among a

predefined set, but since it supports quantifiers, you can generate rich and complex patterns with very little code. For example, `[abc]+` captures every possible string formed of “a”, “b”, and/or “c” in whatever order and of any length—such as “*abacab*”.

Comprehensive classes can often be shortened using character RANGES:

```
[0123456789] → [0-9]
[abcdefghijklmnopqrstuvwxy] → [a-z]
[ABCDEF0123456789] → [A-F0-9]
```

A range specifies the first and the last element, separated by `-` (U+002D HYPHEN-MINUS), of an ordered set of characters. In IndexMatic<sup>3</sup>, all characters of the Unicode Basic Multilingual Plane (BMP) are ordered by codepoint.

**NOTE** Characters beyond the BMP cannot be addressed from simple character classes because they are in fact encoded in UTF16, which requires two string units. → [Codepoints Unicode](#)

Ranges and simple characters can be arbitrarily concatenated, in no particular order. If the character `-` is needed in a class, add it at the beginning of the class specifier, e.g. `[-1-9]`, this way the hyphen is treated literally, not as a range delimiter.

**NOTE** For technical reasons, the escape sequence `\-` is not absolutely safe in IndexMatic<sup>3</sup> class specifiers, so we recommend you stick to the syntax given above.

See also :  
→ [Single-character classes \(Grep emulation\)](#)

# Advanced Queries

**TIP** Although ranges and characters forming a class can be arranged in whatever order, the regex is known to run slightly faster if the most frequent characters are placed at the beginning of the class.

Sometimes it is easier to specify a class negatively, that is, by indicating the characters it prohibits rather than those it allows. Such class, referred to as a **NEGATED CHARACTER CLASS**, is formed on inserting a `^` character after the opening bracket, e. g. `^[^0-9]`. It then matches any character that is not enclosed in the brackets (in our example, “any non-digit character”).

**NOTE** The special character `^` has a different meaning outside of a class specifier. See **Assertions**.

Except for the `^` symbol, the syntax is the same as for positively defined classes. However, because a negated class corresponds to a complementary set, we need to clarify what “any character” means to IndexMatic<sup>3</sup>.

Some codepoints are never encountered in InDesign, others are only used internally by the interpreter. This set of exotic characters, referred to as **FORBIDDEN CODEPOINTS**, defines an outside world that no regex can reach.

**!** Forbidden codepoints are (U+) 0000, 0001, 0002, 0005, 0006, 000B, 000C, 000E, 000F, 0010, 0011, 0012, 0013, 0014, 0015, 001B, 001C, 001D, 001E, 001F, 00AD, FFFF. In IndexMatic<sup>3</sup>, the expression “any character” always excludes these codepoints.

► *Metacharacters* . SPACE - \...

Some classes are so common that we no longer declare them as a comprehensive set of characters. Instead, a shorter code is made available, known as a **METACHARACTER**. The most straightforward example is `\d`, strictly equivalent to `[0-9]`.

**NOTE** Semantically, a metacharacter is equivalent to a class.

IndexMatic<sup>3</sup> provides the following non-standard set of metacharacters (*the NEG column shows codes for the negated classes*):

METACHARACTER	CLASS	NEG
<code>\.</code>	(DOT) Any character (excl. forbidden codepoints). Escape: <code>\.</code>	
<code>\s</code>	(SPACE) Any generic space <b>if GS is on</b> . Any generic space (even if GS is off). Escape: <code>\s</code>	<code>\S</code>
<code>\y</code>	(HYPHEN) Any generic hyphen <b>if GH is on</b> . Any generic hyphen (even if GH is off). Escape: <code>\y</code>	<code>\Y</code>
<code>\w</code>	Any character of the current alphabet, including optional add-ons (digits, underscore. . .) if selected.	<code>\W</code>
<code>\l</code>	Any lowercase letter of the current alphabet. <b>!</b>	<code>\L</code>
<code>\m</code>	Any uppercase letter of the current alphabet. <b>!</b>	<code>\M</code>
<code>\d</code>	Any digit. Equivalent to <code>[0-9]</code> .	<code>\D</code>
<code>\h</code>	Any “horizontal space” as implemented in GREP. Equivalent to <code>[\t\xA0\u2001-\u200A\u202F]</code> .	<code>\H</code>
<code>\t</code>	Only matches the TAB character (U+0009).	
<code>\p{...}</code>	Classes based on Unicode properties ( <i>see next page</i> ).	<code>\P{...}</code>

**!** Also remember that the escape sequence `\~` is required to capture a tilde (~).

**!** When using a metacharacter that discriminates case, make sure your query is case sensitive!

# Advanced Queries

The special scheme `\p{...}`, and its complementary scheme `\P{...}`, addresses classes derived from Unicode character properties. The complete syntax is detailed in the table below.

**NOTE** More on Unicode Character Properties:

[www.unicode.org/versions/latest/ch04.pdf#G39](http://www.unicode.org/versions/latest/ch04.pdf#G39)

Metacharacters based on Unicode properties often specify huge regex classes, so they run slower on average than basic metacharacters. They are useful for scanning a document *agnostically*, that is, without any prior information about the writing system, punctuation marks, symbols, etc. For example, the regex `/\p{L}+/` may discover more words than `/\w+/`, because

⚠ When using a metacharacter that discriminates case, make sure your query is case sensitive!

METACHAR.	CLASS & SAMPLES	METACHAR.	CLASS & SAMPLES	METACHAR.	CLASS & SAMPLES
<b>LETTERS</b>		<b>SYMBOLS</b>		<b>PUNCTUATION</b>	
<code>\p{Ll}</code>	Any <b>Lowercase</b> letter in whatever alphabet: b, é, α, ā, œ, ç, ɥ, ø, ħ... ⚠	<code>\p{Sc}</code>	Any <b>Currency</b> symbol: \$, €, €, ¥, \, ₤, ₮, €, ₩...	<code>\p{Pi}</code>	Any <b>Initial</b> punctuation mark: «, ‘, ’, “, ”, <...
<code>\p{Lu}</code>	Any <b>Uppercase</b> letter in whatever alphabet: À, Ĥ, Dž, Æ, Nj, Ÿ, Ж, Ф, ℋ, N... ⚠	<code>\p{Sk}</code>	Any <b>Modifier</b> symbol: ½, ¼, ⅓, <sup>◌̂</sup>... ⚠	<code>\p{Pf}</code>	Any <b>Final</b> punctuation mark: », ’, ”, >...
<code>\p{Lt}</code>	Any <b>Titlecase</b> letter in whatever alphabet: Dž, Lj, Nj, Dz, Ā, Œ... ⚠	<code>\p{Sm}</code>	Any <b>Math</b> symbol: <,  , ~, ¬, ±, ×, ÷, ∑, ⇒, #...	<code>\p{Ps}</code>	Any <b>Open</b> punctuation mark: (, [, {, [, (, {, ~...
<code>\p{LC}</code>	Superset of <b>Cased</b> letters: Ll + Lu + Lt	<code>\p{So}</code>	Any <b>Other</b> symbol: !, ©, °, ✕, ∞, ∞, %, №, ↓, ☒, ☈, ☉, ™... ⚠	<code>\p{Pe}</code>	Any <b>Close</b> punctuation mark: )], ], }, ], >), ~...
<code>\p{Lm}</code>	Any <b>Modifier</b> letter in whatever alphabet: ḥ, ḥ, ḥ, ḡ, ḡ, ḡ... ⚠	<code>\p{S}</code>	Superset of symbols: Sc + Sk + Sm + So	<code>\p{Pc}</code>	Any <b>Connector</b> punctuation mark: —, ~,  ,  , ---, ~, ~...
<code>\p{Lo}</code>	Any <b>Other</b> letter in whatever alphabet: ª, ¼, ½, ¾, ⅓, ⅓, ⅓, ⅓, ⅓... ⚠	<b>SPECIAL</b>		<code>\p{Pd}</code>	Any <b>Dash</b> punctuation mark: -, -, ~, ~, -... ⚠
<code>\p{L}</code>	Superset of Unicode letters: LC + Lm + Lo	<code>\p{Cc}</code>	Any <b>Control</b> character up to U+009F, excluding forbidden codepoints.	<code>\p{Po}</code>	Any <b>Other</b> punctuation mark: ;, !, #, %, ', :;, @, \$, ¶, ;, ¶, ©, ¶, †, ×...
<b>NUMBERS</b>		<code>\p{Cf}</code>	Any <b>Format</b> character (e.g. ARABIC NUMBER SIGN), excluding SOFT HYPHEN.	<code>\p{P}</code>	Superset: Pi + Pf + Ps + Pe + Pc + Pd + Po
<code>\p{Nd}</code>	Any <b>Decimal</b> number: 0, 9, ¼, ¾, ⅓, ⅓, ⅓, ⅓... ⚠	<code>\p{Cn}</code>	Any <b>Unassigned</b> character (e.g. U+0378).	<b>MARKS</b>	
<code>\p{Nl}</code>	Any <b>Letter</b> number: I, viii, ①, ②, ③, ④... ⚠	<code>\p{Co}</code>	Any character of the <b>Private Use</b> Area (e.g. U+E000).	<code>\p{Mc}</code>	Any <b>Spacing</b> mark (e.g. DEVANAGARI SIGN)
<code>\p{No}</code>	Any <b>Other</b> number: ², ³, ¼, ¾, ⅓, ⅓, ⅓, ⅓, ⅓... ⚠	<code>\p{Cs}</code>	Any <b>Surrogate</b> code, i.e. [�D800-�DFFF].	<code>\p{Me}</code>	Any <b>Enclosing</b> mark
<code>\p{N}</code>	Superset of Unicode numbers: Nd + Nl + No	<code>\p{C}</code>	Superset of special characters: Cc + Cf + Cn + Co + Cs	<code>\p{Mn}</code>	Any <b>Nonspacing</b> mark
				<code>\p{M}</code>	Superset of combining marks: Mc + Me + Mn

# Advanced Queries

it actually captures every sequence of *letters* in the sense of the Unicode standard, while the metacharacter `\w` specifically identifies a letter in the active alphabet (as set in [Finder ▶ Alphabet](#)).

**NOTE** Do not confuse `\w` according to IndexMatic<sup>3</sup> with its usual definition in other regex languages.

Except the dot (`.`), all metacharacters can be added to comprehensive classes as well. E. g., `[_:'\w\s]`, `[\da-f]`, `[\h\y\m]`, or `[\p{P}\p{S}]` are all valid, bringing together the underlying character sets.

**TIP** Inside a class specifier, the dot character `.` simply refers to itself, so you do not have to escape it, e.g. `[.,;:]`.

▶ *Codepoints (Unicode)* `\xHH \uHHHH \u{H...}`

It is usually safe to load exotic characters in a regex, as in `/êâç./` or `/[℄Δἆ3Ψ♥%Ⓢ]+/`. The latter pattern is a random mixture of Latin, Coptic, Cyrillic, Armenian, and Greek characters followed by some symbols. Excluding the set of forbidden characters (see **Character Classes**), all Unicode characters of the Basic Multilingual Plane are supported by the regex engine. More precisely, all UTF16-encoded characters—since UTF16 is internally used by InDesign for manipulating text strings.

Characters beyond the BMP, such as U+1F302 CLOSED UMBRELLA (🛂), are of course available in InDesign documents and can

still be detected by IndexMatic<sup>3</sup>, but they are in fact encoded in UTF16, using two BMP units forming a SURROGATE PAIR:

U+1F302 → U+D83C U+DF02

Thus, both InDesign and IndexMatic<sup>3</sup> treat the Unicode character U+1F302 as a sequence of two special characters. This has important consequences on processing regular expressions. Consider the class `[^aââ]`. Against all odds, it does not contain the character `â`! Indeed, the elements actually loaded in that class are U+D83C, U+DF02, then `a`, `à`, and `â` (five characters in total).

As long as you need to capture one extra-BMP character—in fact, the underlying surrogate pair—outside of any class or quantified expressions, your regex will work. But UTF16 creates serious obstacles to more advanced processing of Unicode features.

To summarize: remember that all regular expressions are basically expressed in UTF16.

**NOTE** *At the time of publishing this manual, some workarounds are still being worked on. They might be implemented in future updates.*

The regex syntax allows you to enter a codepoint (the hexadecimal number behind an Unicode character) instead of the character itself. For example, the letter `a`, which is the character U+0061, can be specified `\x61` as well. There's hardly any reason to do so for such a simple character. However, special

# Advanced Queries

and invisible characters (such as U+00A0 NO-BREAK SPACE) are much easier to handle using their code. Also, your text editor (and IndexMatic<sup>3</sup> interface) will not properly render the glyph of *any* character—say in Arabic, Hebrew, Chinese or Anatolian!—since the fonts presently available on your system do not cover the whole Unicode universe.

For these reasons, you may prefer the codepoint syntax. Two main forms are available, `\xHH` and `\uHHHH` (where *H* stands for any hexadecimal digit):

- `\xHH` encodes the character U+00HH. E.g.:

À → `\xC0`

ß → `\xDF`

This syntax requires exactly two hexadecimal digits, so it cannot deal with codepoints greater than FF.

- `\uHHHH` encodes the character U+HHHH. E.g.:

À → `\u00C0`

Ł → `\u0141`

ᄀ → `\u266B`

This syntax requires exactly four hexadecimal digits, so it can address any character of the BMP—including Hangul syllables, CJK compatibility ideographs, etc.

**NOTE** The prefixes `\x` and `\u` must be lowercase. It is recommended (but not mandatory) to write the hexadecimal digits A to F in uppercase form.

`\xHH` and `\uHHHH` are equivalent to the characters they stand for, so each code is parsed as a single element: `/a\xDF?/` has the same meaning as `/aß?/`, the range `[\u2669-\u266C]` is the same as `[ᄀ-ᄁ]`, and so on.

In IndexMatic<sup>3</sup>, a third scheme is also supported, `\u{H...}`, with a variable number of hexadecimal digits:

À → `\u{C0}`

Ł → `\u{141}`

ᄀ → `\u{266B}`

and even

ᄁ → `\u{1F302}`

The `\u{H...}` syntax can indeed eat codepoints beyond U+FFFF (up to U+10FFFFD). How is that possible? The interpreter simply generates the corresponding surrogate pair. Thus `\u{1F302}` is translated into `(?:\uD83C\uDF02)` under the hood. As you can see, a non-capturing group is created to make basic operations and quantifiers, like `\u{1F302}+`, still work as expected.

⚠ However, the above trick does not work in classes, so any `\u{H...}` code that would go beyond FFFF is forbidden in class specifiers. The regex interpreter will purely ignore it!

**NOTE** Rather than `\u{` you can equivalently use `\x{`. The two schemes are supported to comply with both modern ECMAScript RegExp syntax `\u{...}` and InDesign GREG syntax `\x{...}`.

# Advanced Queries

► Assertions      `^... ...$ ...(?=...) ...(?!...)`

ASSERTIONS are special codes that do not consume any character but rather specify conditions on pattern boundaries, beginnings and/or endings. IndexMatic<sup>3</sup> basically supports four assertions:

- 1) The starting assertion, using the scheme `/^.../`, requires the entire match to be found at the beginning of a paragraph or style range (depending on whether a character style filter is active). E.g.

```
/^Hello/I
```

will only consider instances of “Hello” that start a paragraph (assuming no filter is used).

- 2) The ending assertion, using the scheme `/...$/`, requires the entire match to be found at the very end of a paragraph or style range (depending on whether a character style filter is active). E.g.

```
/world!$/I
```

will only consider instances of “world!” that terminate a paragraph (assuming no filter is used).

**NOTE** When a character style filter splits the text into ranges (i.e. targets) which are subparts of a paragraph, each of those ranges is then considered a whole unit and the `^...` and `...$` assertions can identify any of them.

- 3) The positive lookahead, using the scheme `...(?=...)`, matches the first part only if it is followed by the second part of the pattern. E.g.

```
/Hello(?= world)/I
```

will only consider instances of “Hello” followed by “world”. This regex is similar to `/Hello world/I` but it only captures the “Hello” part; hence the “world” part won’t be included in the match (`$0`).

- 4) The negative lookahead, using the scheme `...(?!...)`, matches the first part only if it is not followed by the second part of the pattern. E.g.

```
/Hello(?! world)/I
```

will only consider instances of “Hello” *not* followed by “world”.

⚠ Unlike GREP and other regex languages, IndexMatic<sup>3</sup> does not natively support “lookbehind” assertions. However, you can get the effect of a positive lookbehind using the scheme

```
/Hello (world)/I => $1
```

and mimic a negative lookbehind using the scheme

```
/Hello world|(world)/I => $1
```

More on the “fake lookbehind” trick:  
<https://indiscripts.com/post/2022/01/indexmatic-how-to-emulate-lookbehind>



# Advanced Queries



For syntactical reasons, IndexMatic<sup>3</sup> also supports the usual “word boundary” assertion `\b` (and its negative counterpart `\B`) as implemented in ExtendScript. However, as they remain insensitive to non-ASCII letters, these codes make little sense with extended alphabets. Better is to use either the Whole Word option (if relevant), or the metacharacters `\w` and `\W` (according to your needs).

⚠ Most GREP-specific assertions like `\<`, `\>`, `\A`, `\z`, `\K`, are not implemented in IndexMatic<sup>3</sup>.

## 10. Emulated GREP Features

**DEFN** GREP is a regular expression language inherited from Unix. InDesign provides its own implementation (referred to as “InDesign GREP”) in the Find/Change dialog of the application.

Keep in mind that IndexMatic<sup>3</sup> does not invoke, nor rely on, InDesign GREP commands. Although this technology is available to the scripting subsystem, it has a high performance cost, so querying InDesign GREP would have considerably increased the response time of the program.

Instead, IndexMatic<sup>3</sup> implements its own syntax (→ **IndexMatic Regex Syntax**), mostly based on common ExtendScript/JavaScript RegExp specification, as defined in ECMA-262, 5th edition.

Some GREP commands or metacharacters are emulated in IndexMatic<sup>3</sup> anyway, in order to maximize the consistency between the two dialects.

### ► GREP-like Metacharacters

CODE	CHARACTER
<code>~y</code>	Right Indent Tab
<code>~S</code>	Non-Breaking Space
<code>~S</code>	Fixed Width NBSP
<code>~f</code>	Flush Space
<code>~&gt;</code>	En Space
<code>~m</code>	Em Space
<code>~3</code>	Third Space
<code>~4</code>	Quarter Space
<code>~%</code>	Sixth Space
<code>~/</code>	Figure Space
<code>~.</code>	Punctuation Space
<code>~&lt;</code>	Thin Space
<code>~ </code>	Hair Space
<code>~k</code>	Discr. Line Break
<code>~j</code>	Non Joiner (ZWNJ)
<code>~~</code>	Non-Breaking Hyphen
<code>~=</code>	En Dash
<code>~_</code>	Em Dash

CODE	CHARACTER
<code>~"</code>	Straight Double Quotes
<code>~'</code>	Straight Single Quote
<code>~[</code>	Left Single Quote
<code>~]</code>	Right Single Quote
<code>~{</code>	Left Double Quotes
<code>~}</code>	Right Double Quotes
<code>~2</code>	© Copyright sign
<code>~6</code>	§ Section sign
<code>~7</code>	¶ Pilcrow
<code>~8</code>	• Bullet
<code>~r</code>	® Registered sign
<code>~e</code>	… Ellipsis
<code>~d</code>	™ Trademark Symbol
<code>~h</code>	End Nested Style Here
<code>~i</code>	Indent To Here
<code>~v</code>	Any text variable
<code>~x</code>	Section marker
<code>~I</code>	General Marker
<code>~a</code>	Anchored Object Marker

List of ~ metacharacters supported in IndexMatic<sup>3</sup> regexes.

- Spaces
- Hyphens and dashes
- Quotation marks
- Symbols
- Markers

bugfix from version 3.24012

⚠ If you need to capture the tilde character (`~`) literally from a regex always use the escape sequence `\\~`. This syntax was neither documented nor specified in earlier versions of the program, which caused critical issues especially when declaring character classes.

## ► GREP-like Operators

Most GREP operators and/or commands are universally shared by regex languages: classes [...], groups (...), non-capturing groups (?:...), quantifiers ? + \* {,} etc. Here are some notable exceptions and more specific rules:

- IndexMatic<sup>3</sup> does not emulate the specific GREP switches that control case-sensitivity, space-sensitivity, comments, subroutines, multiline or single-line modes.
- It implements positive and negative *lookahead* assertions, not *lookbehind* assertions.
- It supports the `^...` and `...$` assertions, not the GREP-specific `\<` and `\>` codes.
- It does not support POSIX classes of the form `[ [ . . . ] ]` but most of them can be easily translated into **Unicode Properties** `\p{...}` and `\P{...}`. In addition, IndexMatic<sup>3</sup> automatically converts single-character classes like `[a]` or `[é]` into the set of diacritical variants based on the root letter, e.g. `[aââ...]`
- With respect to the current alphabet,  
`\w` (in IndexMatic<sup>3</sup>) is equivalent to `[\l\u]` (GREP), and  
`\m` (in IndexMatic<sup>3</sup>) is equivalent to `\u` (GREP).

**NOTE** The code `\l` is consistent (denoting a lowercase letter), but `\u` had to be used for codepoints specifiers of the form `\uHHHH`.

- The `\Q...E` scheme is supported in IndexMatic<sup>3</sup>: in a regular expression, all characters between the `\Q` and `E` marks are then parsed literally (as in InDesign GREP) so you don't need to escape special symbols within that scope. E.g., the query

```
/\w+ \Q(a+b/c)\E.+/
```

temporarily deactivates the regex interpreter between `\Q` and `E` so the enclosed expression `(a+b/c)` is searched as typed.

**NOTE** ⚠ All characters inside the `\Q...E` region are read literally, including the slash symbol `/` which then no longer acts as a terminator. Also, the space and hyphen characters do not work in a “generic” way even if Generic Space (resp. Generic Hyphen) is turned on. Use the `\Q...E` syntax only if some part(s) of your regular expression must be isolated from all regex effects (operators, metacharacters, etc.)

## 11. Query List: Comments and Cross-References

Since a QUERY LIST (QL) is just a sequence of queries, any feature already available at the individual query level can be processed from within a QL. Empty lines are ignored, those beginning with a slash `/` are assumed to declare a REGEX query, and the others are TOKEN queries.

**NOTE** Remember that the slash character `/` must still be escaped `\/` in token queries, as it is expected to introduce a flag sequence.

⚠ Finally, let's point out that certain unobservable bugs in GREP affect ExtendScript (hence IndexMatic<sup>3</sup>) ... and vice versa!

One of the most sensitive is the frequent malfunction of quantified alternatives of the form `(a|b|c)?`, which can lead to an infinite loop and InDesign to crash. In this specific case, the equivalent pattern `(a|b|c|)` will often help you out.

from  
version  
3.23041

The class `[']` both captures straight and typographic apostrophe (`'`).

# Advanced Queries

However, additional INSTRUCTIONS are specifically available to query lists, based on the prefix `//` (double slash) which then introduces either a COMMENT, a CROSS-REFERENCE or a DIRECTIVE.

**NOTE** None of these QL instructions are allowed in *Single Query* mode since they do not denote actual queries. If you need to capture the particular string `“//”` (formed of two slashes), use the escape sequence `\\` in your query.

## ► Comments `// ...`

By default, any line that begins with `//` is considered a COMMENT in your QL file, unless it matches one of the patterns discussed in the next sections.

Comments are ignored by the interpreter. They help you structure your list and make it more readable, especially if it contains a large number of items.

**NOTE** A comment line is necessarily distinct from a query line. For the time being, IndexMatic<sup>3</sup> does not allow you to append a comment at the end of a query line (as you would do in standard programming languages).

```
//=====
// MyQueryList 12/20/2023
// Author: Bob rev. 2.3
//=====

// Main Concepts
//-----
philosophy
beauty
nature
/facts?
being

// Philosophers
//-----
Aristotle
Confucius
Plato
Descartes => $, René
Hume => $, David
Kant => $, Immanuel
Nietzsche => $, Friedrich
Wittgenstein => $, Ludwig

// etc
//-----
```

## ► Cross-References `// ... => ...`

A line that begins with `//` and contains the `=>` operator is assumed to declare a CROSS-REFERENCE:

```
// TOPIC => REFERENCE
```

where *TOPIC* represents some topic structure and *REFERENCE* an arbitrary expression associated with it. Such instruction will automatically generate, in the final index, a line of the form

```
topic See reference
```

or, in case *topic* is the target topic of a distinct query,

```
topic 12, 34-35, 67 See also reference
           page locators (query)           cross-ref.
```

- In the first case above, the *topic* did not arise from any existing query so the index entry simply results from the cross-reference instruction. This is a SEE REFERENCE.
- In the second case, a standard query yielded that term (and the corresponding pages). The reference is then added after the locators; this is a SEE-ALSO REFERENCE.

**NOTE** More on formatting cross-references and customizing index separators in → [Setting Up Separators and Delimiters](#).

# Advanced Queries

An easy way to memorize how the `// TOPIC => REFERENCE` scheme works is to read the operator `=>` as meaning “see” or “see also”, depending on whether the left-hand *topic* is targeted by a query.

That aside, IndexMatic<sup>3</sup> makes no assumption about the *reference* (right-hand part); it could be an existing heading as well as any other source—external document, author, or bibliographical indicator—outside of the indexed area.

Therefore, if you want to manage only *internal* cross-references, it is your responsibility to check that these expressions actually point to an existing element among the index entries.

The below example shows how different options may combine. Instruction ① links *Plato* to *Socrates* (SEE-ALSO reference, since *Plato* exists separately); ② creates a SEE reference from *hemlock* to

*Socrates: hemlock* is not requested (and maybe even not present in the document), but the reader is led to *Socrates* in case s/he looks up that word in the index. Both ① and ② link to a topic which happens to be actually indexed (*Socrates*). Finally, ③ links *Aristotle* to an external label, “[DocA:5]”. Here is declared a SEE-ALSO reference whose target is not part of the indexed terms.

**NOTE** ⚠ Do not confuse the `=>` operator in a `// TOPIC => REF` instruction with the `=>` operator in ordinary `KEY=>TOPIC` queries (then connecting a key to a topic).

If relevant, PARENT TOPIC(s) can be specified on the *TOPIC* side of any cross-reference,

```
// People > Philosophers > Plato => Socrates
```

declaring a tree structure as discussed in → **Nested Topics**:

```
People
  Philosophers
    Plato See also Socrates
```

Here again, it is up to you to make this hierarchy coincide with that resulting from a separate query:

```
// Query:
Plato => People > Philosophers > $
// Consistent XRef:
// People > Philosophers > Plato => Socrates
```

QUERY LIST	RESULT
<pre>// Main Concepts philosophy/3 beauty nature  // Philosophers Aristotle Plato Socrates  // X-refs ① // Plato =&gt; Socrates ② // hemlock =&gt; Socrates ③ // Aristotle =&gt; [DocA:5]</pre>	<pre>INDEX  Aristotle 27, 39-40; See also [DocA:5] ③ beauty 13, 40, 67 hemlock See Socrates ② nature 17, 25, 71 philosophy 5, 12-23, 38-51, 62 Plato 23-27, 42-44; See also Socrates ① Socrates 8, 11, 25-31, 59-63</pre>

# Advanced Queries

Space characters between the syntactic operators `//`, `>` and `=>` are optional, although recommended for readability.

On rare occasions, it might happen that a simple comment requires the characters “=>” without expecting a cross-reference. In such case, use the syntax `//! . . .=>. . .` to prevent IndexMatic<sup>3</sup> from interpreting the line as declaring a cross-reference.

**TIP** The prefix `//!` turns any line into a forced comment, no matter the special symbols it may contain.

## 12. Query List: Directives

`//~ . . .`

DIRECTIVES are special instructions introduced in IndexMatic<sup>3</sup>, based on the prefix `//~`. Unlike comments and cross-references, they can interact with other elements of your query list and transform the queries before being sent to the interpreter.

Directives are powerful but require a little concentration to be used properly. The key idea is simple: many query lists have repeated patterns, or reusable items, that you would like to control in a more global way. For example, instead of

```
/(dog)s?/i => animal > ~$1~
/(cat)s?/i => animal > ~$1~
/(rabbit)s?/i => animal > ~$1~
// etc
```

wouldn't it be nice to simply provide a basic word list,

```
dog
cat
rabbit
. . .
```

and to tell IndexMatic<sup>3</sup> how to transform each line so it works as a full, well-configured regex query? As we shall see in a moment, you can now automate this procedure.

**NOTE** Although directives appear as specific lines in a query list file, they are not IndexMatic<sup>3</sup> queries in themselves (rather meta-controllers operating on lines). In this manual, directives are printed in **purple** to prevent any confusion with the normal query syntax.

### ► *General Rules*

---

Any directive has one of the three canonical forms:

```
//~name param
```

```
//~name :: output
```

```
//~name param :: output
```

where *name* denotes the RESERVED NAME of the directive, *param* a PARAMETER (sometimes in quotes, sometimes optional) and *output* a PATTERN only needed in directives that transform

# Advanced Queries



existing lines. These elements are separated by a space character wherever needed, each directive being entirely formatted on a single line which imperatively begins with the sequence `//~` (slash slash tilde). The `::` operator, formed of two colons, is only required when an *output* element is expected.

In the declarative part (`//~name`) you can detach the name of the directive by a space character. For example,

```
//~include test/myQueryList.txt
```

and

```
//~ include test/myQueryList.txt
```

are equivalent.

With the exception of `include`, all IndexMatic<sup>3</sup> directives act on the lines following them in the query list, until they encounter either a blank line or a new directive which terminates the action of the present one. During this preprocessing stage, comments and cross-reference instructions are ignored.

**TIP** You can also add an “empty directive”

```
//~
```

at any location to forcibly interrupt the present directive.

This syntax is therefore equivalent to a blank line.

## ► *Output Pattern*

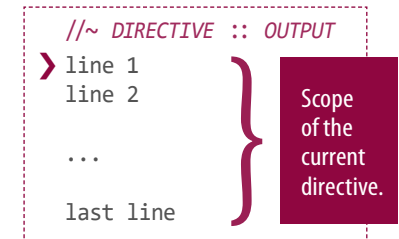
Directives that expect an *output* pattern (like `split` and `format`) will use it to build the final query, the one actually supplied to the interpreter. The scenario is shown below: each line in the scope of the present directive is processed and changed into a new one as dictated by the *output* pattern.

How input lines are parsed depends on the directive in use, but the output line is always generated the same way, based on PLACEHOLDERS `^0`, `^1`, `^2`... identified in the *output* pattern:

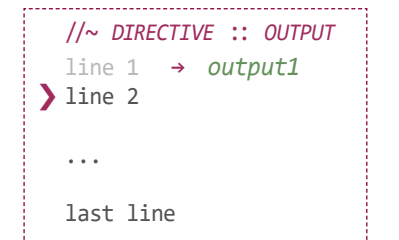
- `^0` represents the original input string, that is, the whole line as it was before any treatment.
- `^1` represents the first match extracted by the directive, `^2` the second match, and so on.

**NOTE** As most directives are based on regular expressions, you find here mechanisms close to those already defined for regex queries. In particular, directive placeholders `^0`, `^1`... play a role similar to that of `$0`, `$1`... variables.

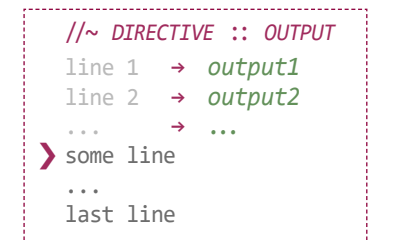
- Given a placeholder `^n` (e.g. `^2`), the syntax `^<n` represents the first character of the string contained in `^n`. E.g., if `^2` contains “hello”, `^<2` denotes the character ‘h’.



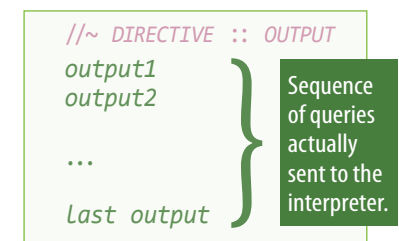
↓ replacement step



↓ replacement step



↓ ...





# Advanced Queries

- Given a placeholder `^n`, the syntax `^.n` represents the expression `(?:S|C\.)` where `S` denotes the string contained in `^n` and `C` the very first character of that string. E.g., if `^3` contains “Albert”, `^.3` stands for `(?:Albert|A\.)`, which captures “Albert” as well as “A.”
- Given a placeholder `^n`, the syntax `^:n` represents the expression `[Cc]T` where `C` (resp. `c`) denotes the first character uppercase (resp. lowercase) of the string contained in `^n` and `T` the next characters of that string. E.g., if `^1` contains “De Niro”, `^:1` stands for `[Dd]e Niro`, which captures “de Niro” as well as “De Niro” in case-sensitive queries.

Apart from placeholders (`^0`, `^1`...) and their special modifiers, all characters in the *output* pattern are interpreted literally. So, for example, the pattern `/^1/I => $1 > ^0` only regards the highlighted parts as variable elements; other symbols are reproduced as is without transformation. Entities like `/`, `=>`, `$1`, `>` have no *meaning* at the directive level—although they will mean a lot to the interpreter!

► *Split Directive* `//~split separator :: output`

The `split` directive splits an input line into substrings (`^1`, `^2`...) according to a *separator*, then generates the output line with respect to the *output* pattern. The separator must be specified either as a string enclosed in double or simple quotes, e.g.

```
//~split ", " :: ^1 => $
```

or as a valid JavaScript RegExp enclosed in slashes, e.g.

```
//~split /[,;] ?/ :: ^1 => $
```

**NOTE** In the second example above, either a comma or a semicolon, optionally followed by a space, will act as a separator, so it can split lines of the form “*abc,efg,xyz*”, “*abc;efg;xyz*”, or even mixed formats like “*abc,efg; tuv;xyz*”.

The placeholders `^1`, `^2`... (up to `^9`) then denote the consecutive parts (excl. the separator) of the original string.

This directive is useful for processing lists of the form

```
Astaire, Fred
Bacall, Lauren
Bardot, Brigitte
Bellucci, Monica
Bogarde, Dirk
Bogart, Humphrey
```

The last and first names can be separated to form a more elaborate query, like `/Astaire/I => Astaire (Fred)`. To do so, prepend the directive

```
//~split ", " :: /^1/I => ^1 (^2)
```

Note that `^1` reflects the last name, `^2` the first name. You could also use `^0` to recover the original line (e.g. “Astaire, Fred”).

from  
version  
3.23041

The `~split` directive will keep the input line (`^0`) unchanged if the separator is not found. (The output pattern is then purely ignored.)

# Advanced Queries

```
//~split ", " :: /^1/I => ^0
Astaire, Fred
Bacall, Lauren
Bardot, Brigitte
Bellucci, Monica
Bogarde, Dirk
Bogart, Humphrey
```

This example shows you that a simple list of people—which is not originally formatted in a way that provides relevant IndexMatic<sup>3</sup> queries—can be easily preprocessed and changed into a form that now fits our needs. In book indexing, surnames (*Astaire*, *Bacall*, *Bardot*..) are usually a sufficient key. At least, most of them are particular enough—assumed case-sensitive—in a specific field such as cinema, philosophy, politics. That’s why our key is represented by `/^1/I` in the above directive, while the index entry, the full term, is `^0`.

**TIP** By addressing such consistent patterns, directives offer a kind of batch process that may cover 90%, if not 99%, of your final index. Of course, there are still exceptions that you will have to deal with more locally, e.g. “Gene” vs. “Grace” Kelly, or “Isabella” vs. “Roberto” Rossellini. A good practice is to structure your query lists into different sections according to the formal criteria to be applied, then to associate with each sub-list the right directive.

► *Format Directive*      `//~format regex :: output`  
    `//~format :: output`

The `format` directive extracts substrings (`^1`, `^2`..) from the input line according to a regular expression (if supplied), then it generates the output line with respect to the *output* pattern. Specify a valid JS regex enclosed in slashes, with optional `/g` and/or `/i` flags, e.g.

```
//~format /[a-z]+/g :: ^0 => ^1
```

or, use an equivalent string form (in quotes), e.g.

```
//~format "[a-z]+" :: ^0 => ^1
```

With the latter syntax, however, the regex can no longer be declared as *global* (`/g` flag) so it will only capture the first match.

**NOTE** ⚠ The regex dialect supported in such directives is restricted to that of ExtendScript Regular Expressions. Do not confuse it with the IndexMatic<sup>3</sup> query language described in this manual.

`format` also supports a variant which does not require the *regex* parameter at all, e.g.

```
//~format :: /^0/i => $
```

Only the `^0` placeholder is then available, standing for the whole input string. This simple scheme is anyway very powerful, as it

# Advanced Queries

allows you to reformat basic word lists in much more efficient commands:

```
//~format :: /^0s?/iw => ^0
cat
dog
rabbit
```

The above code will transform each word (“cat”, “dog”...) into the corresponding fine-tuned query (`/cats?/iw=>cat`, ...) capturing both singular and plural forms and producing the expected index entry.

The `format` directive is also useful for processing a multi-level index. Instead of rewriting the `=>topic > ...` part for every item of the same category, just write it once in the directive, and group your queries accordingly:

```
//~format :: ^0 => Philosophers > $
Aristotle/I
Confucius/I
Plato/I
Socrates/I
```

**NOTE** In the above example, the meta-syntax `^0` represents the input line while `$` is so far transparent. Then, once parsed by the interpreter, the resulting code just behaves as usual and `$` denotes the key of the query, e.g. “Aristotle” in `Aristotle/I=>Philosophers > $`.

## ► *Include Directive* `//~include file`

The `include` directive gets an external QL file included in the present one, as if all the lines of the external file were written at the very location of the directive. The *file* specifier is relative to the current folder (that is, the one that contains the query list file from where `include` is invoked). You can specify relative paths, e.g. `sub1/sub2/myQueries.txt`, and access parent folders using the conventional `../` operator.

Absolute paths are allowed too:

```
//~include c:/MyDocs/IX/queries/myQueries.txt
```

This directive opens up endless possibilities for reusing resources in different projects. For example, if you have designed a QL specializing in movie stars, you can just include it while indexing any new book dedicated to cinema.

**NOTE** ⚠ Unlike other directives, `include` does not break the action of an upstream directive. See it as a pure copy-paste mechanism. The `//~include...` line is, so to speak, “replaced” by those coming from the external file. The active directive (if any) is always determined regardless of `include`. Of course the included lines may in turn introduce their own directives.

# Output and Reports



The **Output** panel centralizes all the settings governing the production of the final index (output file, formatting, sorting and many other options) based on your present scope, filters and queries. It exposes three subpanels: **Destination**, **Layout** and **Sorting**. The **Destination** subpanel brings together scattered features related to the processing of index entries, while **Layout** and **Sorting** control more specific options that you will usually set once and for all in a project.

## 1. Output File Formats

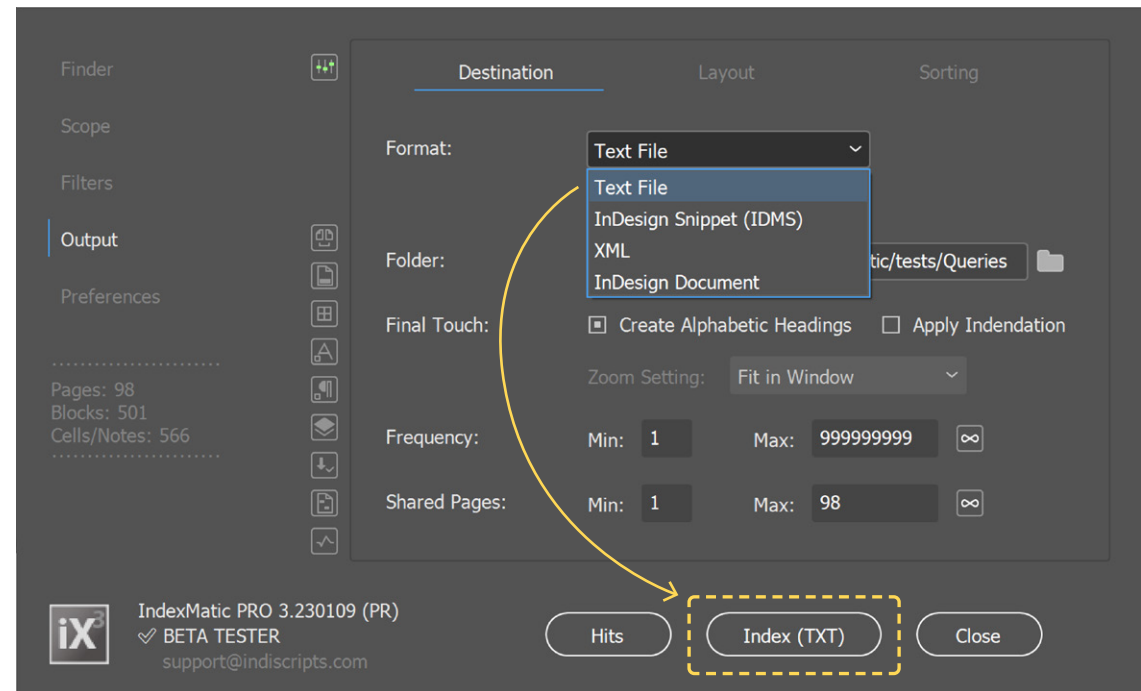
IndexMatic<sup>3</sup> can generate either a basic plain text file (taking on the .TXT extension), an IDMS snippet (placeable in InDesign), a pure XML file (which you can then process from a dedicated program or database), or an InDesign document.

Go to **Output** ► **Destination** ► **Format** to select the desired file format.

**TIP** The **Index** button at the bottom of the dialog box then reflects the corresponding extension: (*TXT*), (*IDMS*), (*XML*), or (*INDD*). This way you know what kind of file IndexMatic<sup>3</sup> is about to produce even if the **Output** panel is not visible.

► *Text File* .txt

The *Text File* format is the most universal in the sense that it can be read from any simple text editor. It doesn't support style



enrichments. All characters are UTF8 encoded. Subtopics are indented using either tab or space characters (as defined in the **Layout** subpanel). Here's what a typical *text-file* index looks like:

```
attack 159, 171-172, 174-175, 177-178, 189, 192, 212-213, 215, 225, 232,
development 238-239, 241, 245
exchange 157-158, 162, 166, 174, 176, 188-189, 196, 201-203, 205-206, 2
opponent 169, 221, 230, 235, 247, 251, 257
pieces 180, 183, 186, 195, 207, 232, 235, 254
Players
  Alekhine 163; See also Sabadell
  Anand 184, 188, 191, 208, 240, 242-243, 265, 268
  Aronian 167, 185, 191, 195-196, 198, 208, 256
  Bacrot 265
  Beliavsky 200, 203, 231
  Carlsen 155, 157, 161-164, 166-167, 169, 171, 173-174, 176-188, 190-19
  Dominguez 205, 214
```

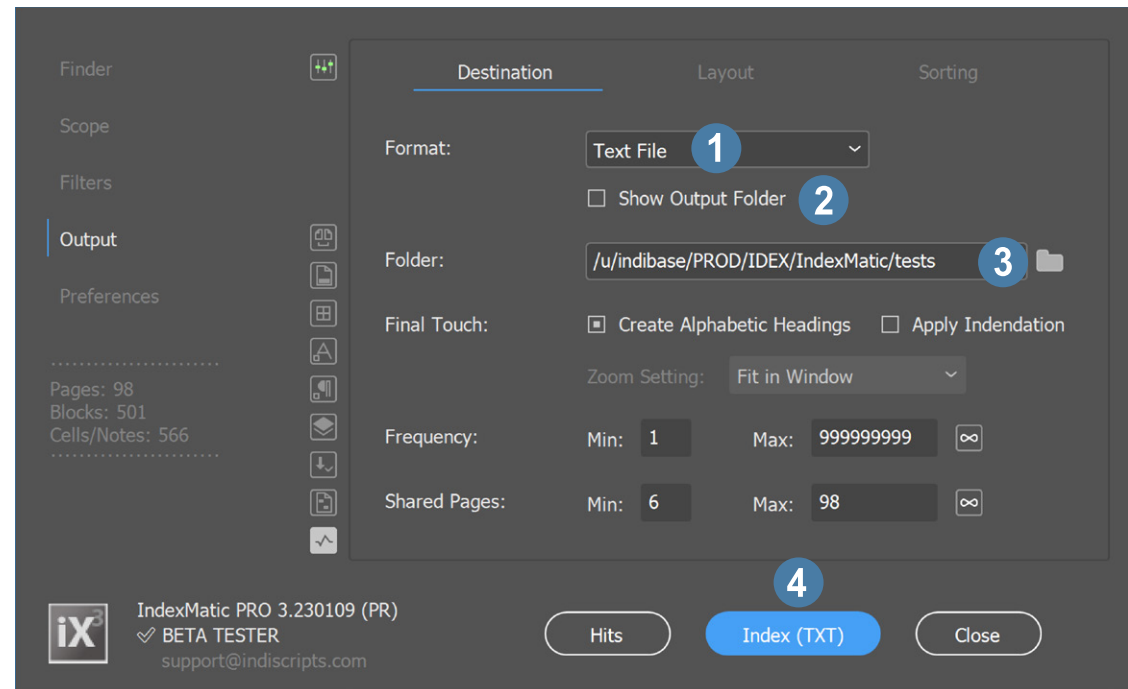
The **Output** ► **Destination** subpanel exposes all important settings in relation with the final index structure. Select the desired file format in the drop-down list to access more specific options.

# Output and Reports



To generate your index as a plain text file:

- 1) Activate the **Output ▶ Destination** subpanel and select **Format: Text File**.
- 2) Click the **Show Output Folder** checkbox to instruct IndexMatic<sup>3</sup> to reveal the parent folder instead of opening the TXT file. (If this option is turned off, the program will directly attempt to open the index in your default text editor).
- 3) In the **Folder** field, type or paste the path of the output folder. Alternatively, click the icon button to the right of the input field to select a particular folder.
- 4) Click **Index (TXT)**.



## ▶ InDesign Snippet .idms

**DEFN** IDML (InDesign Markup Language) is an XML-based format for representing InDesign content. A “snippet” is a file that holds and encodes InDesign objects using a subset of the IDML syntax (IDMS). These “.idms” files are specially convenient for storing individual components (shapes, frames. . .) that you want to re-use in your project.

IndexMatic<sup>3</sup> is able to generate a fine-tuned InDesign SNIPPET (IDMS file) that both reflects your index content and pre-assigns consistent styles to all interesting elements. For example,

HEADINGS at different levels will receive a distinct paragraph style (*H0, H1, H2...*) so you can adjust separately text attributes like indent, size, paragraph spacing, or keep options. Also, dedicated character styles are applied to terms, page locators, cross-references, separators, “See” and “See also” marks.

**NOTE** Output as a snippet, your index can then be placed (manually or automatically) into the last section of your InDesign document or book. If you need to redo the operation when rebuilding the index, existing style attributes are not lost because the snippet never overrides settings you have already defined.

# Output and Reports



The .IDMS file generated by IndexMatic<sup>3</sup> can be stored temporarily or permanently depending on your preferences—see **Preferences ► General ► IDMS**. If you decide to place the snippet directly into your document *via* IndexMatic<sup>3</sup>, you don't necessarily need to keep the file in memory.

The IDMS component basically encodes a text frame container, a story and various object, paragraph and character styles arranged in their respective group:

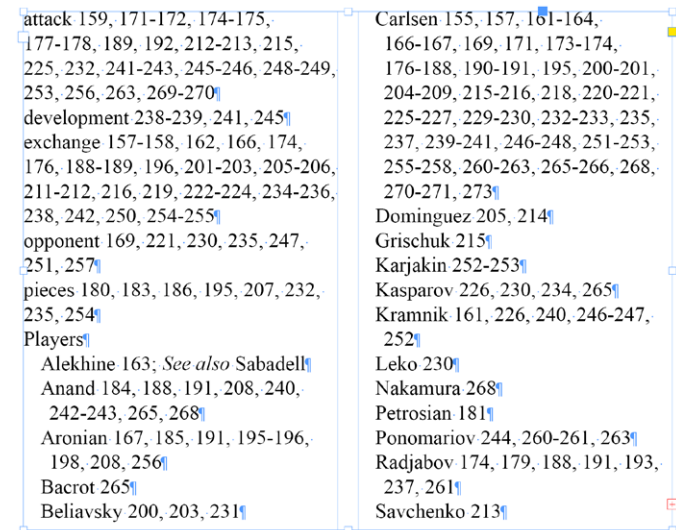
STYLES	BASE	DESCRIPTION
<b>[INDEX]</b>		
<b>IndexFrame</b>	<i>[None]</i>	Style of the index text frame.
<b>[INDEX]</b>		
<b>Base</b>	<i>[No Parag. Style]</i>	Base style for all headings.
<b>H0</b>	<i>Base</i>	Zero-indent heading.
<b>H1</b>	<i>Base</i>	First sublevel heading.
<b>H2</b>	<i>Base</i>	Second sublevel heading.
<b>...</b>	<i>Base</i>	Third, fourth... sublevels.
<b>[INDEX]</b>		
<b>Separator</b>	<i>[None]</i>	Base style for all separators.
<b>Locator</b>	<i>[None]</i>	Base style for all locators.
<b>Locator-Pages</b>	<i>Locator</i>	Specific style for pages.
<b>Locator-Xref</b>	<i>Locator</i>	Specific style for cross-references.
<b>See-Marker</b>	<i>Separator</i>	Specific style for "See".
<b>See-Also-Marker</b>	<i>See-Marker</i>	Specific style for "See also".
<b>T0</b>	<i>[None]</i>	Topic in H0-styled paragraph.
<b>T1</b>	<i>[None]</i>	Topic in H1-styled paragraph.
<b>T2</b>	<i>[None]</i>	Topic in H2-styled paragraph.
<b>...</b>	<i>[None]</i>	Topic in H3, H4, etc.

■ Object Styles   ■ Paragraph Styles   ■ Character Styles

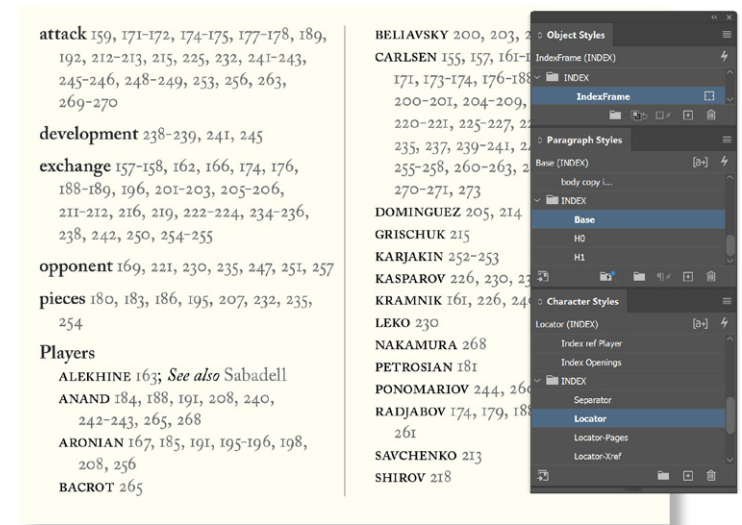
A typical index snippet is shown below in its fresh (placed) state. As you can see, it involves a two-column text frame (as instructed by the *IndexFrame* object style). By default, most paragraph and character styles are made *transparent*.

From there you can control and re-style the presentation of your index: just go into the desired Styles panel and change the settings of any particular target style under the *[INDEX]* group. It only takes a few operations to completely redefine the formatting of your index entries in a way that fits your layout.

As long as your custom settings persist in your InDesign document, you can re-run IndexMatic<sup>3</sup> and place a new index snippet without losing your presentation.



DEFAULT (NEUTRAL) LAYOUT



CUSTOM STYLES

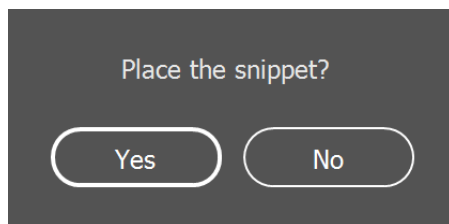


# Output and Reports



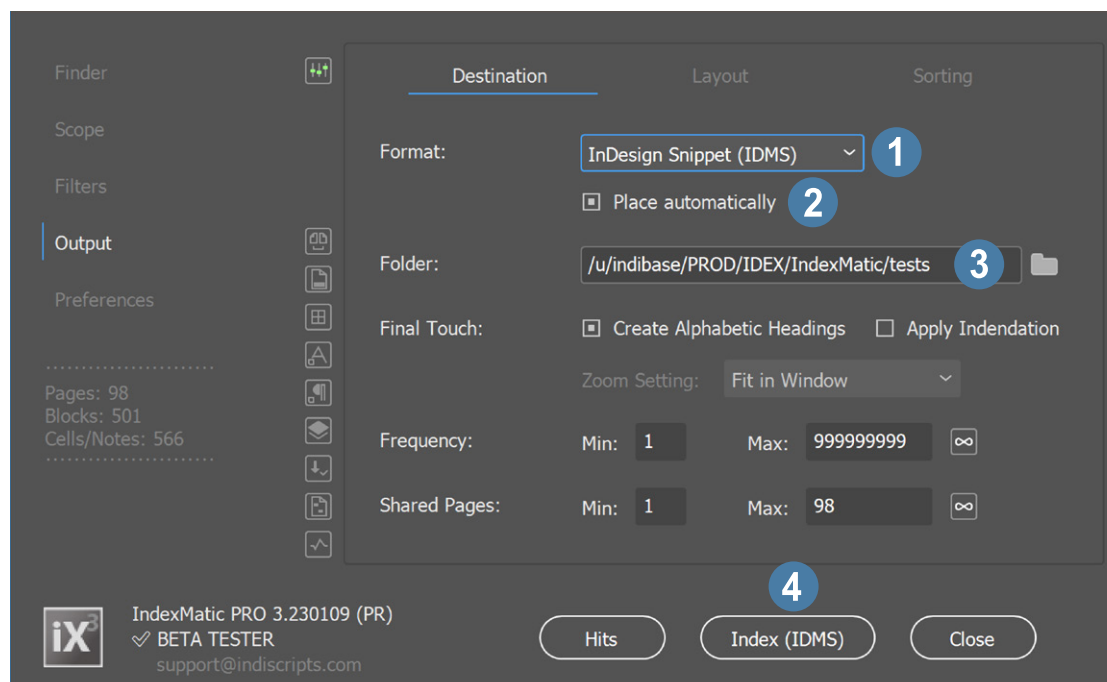
To generate your index as an InDesign snippet:

- 1) Activate the **Output ► Destination** subpanel and select **Format: InDesign Snippet (IDMS)**.
- 2) Click the **Place automatically** checkbox to let IndexMatic<sup>3</sup> try to place the index in your document or book.
  - When this option is active, the program attempts to place the snippet on an appropriate page or to update a previously positioned snippet. If no satisfactory solution is found, IndexMatic<sup>3</sup> populates the placegun with the IDMS file so you can still place the snippet manually.
  - When **Place automatically** is turned off, IndexMatic<sup>3</sup> creates the IDMS file and lets you decide:



Click **No** to only reveal the IDMS file in the output folder.

- 3) In the **Folder** field, type or paste the path of the output folder. Alternatively, click the icon button to the right of the input field to select a particular folder.
- 4) Click **Index (IDMS)**.



**NOTE** Saving your index on disk as a standalone IDMS file is the best option if the final section of your book—the one containing bibliographic elements, indexes, colophon, etc.—is not visible or available at the time you invoke IndexMatic<sup>3</sup>.

**NOTE** ⚠ InDesign snippets cannot embed hyperlinks. If you want IndexMatic<sup>3</sup> to also create hyperlinks from locators to book pages, you need to export the index as an “InDesign Document” (as we shall see below).

# Output and Reports



## ► XML .xml

```
<index scope="Chess Life CXX" date="Saturday, December 31 2022 13:08:11" pageCount="122" creator="
<entries>
  <entry depth="0" kind="term" value="attack">
    <locators>
      <locator kind="pages" label="pg">159, 163-164, 166-167, 170-172, 174-175, 177-180, 187-190
    </locators>
  </entry>
  <entry depth="0" kind="term" value="development">
    <locators>
      <locator kind="pages" label="pg">155-156, 160, 163-165, 167, 174, 185, 195, 198-199, 226-2
    </locators>
  </entry>
  <entry depth="0" kind="topic" value="Players"> TOPIC: "Players"
    <entries>
      <entry depth="1" kind="term" value="Alekhine">
        <locators>
          <locator kind="pages" label="pg">155, 163, 183, 197, 230</locator> PAGES
          <locator kind="xref" label="seeAlso">Sabadell</locator> & XREFS
        </locators>
      </entry>
      <entry depth="1" kind="term" value="Anand">
        <locators>
          <locator kind="pages" label="pg">162, 173, 184, 188, 191, 198, 204, 208-209, 212, 215,

```

If your index has to be post-processed through third-party software, the XML format might be a convenient export mode. The index entries are then structured according to this DTD:

```
<!DOCTYPE index [
  <!ELEMENT index (entries)>
  <!ATTLIST index scope CDATA #IMPLIED>
  <!ATTLIST index date CDATA #IMPLIED>
  <!ATTLIST index pageCount CDATA #IMPLIED>
  <!ATTLIST index creator CDATA "IndexMatic PRO 3">
  <!ELEMENT entries (entry*)>
  <!ELEMENT entry (locators?,entries?)>
  <!ATTLIST entry depth (0|1|2|3|4|5|6|7|8|9) "0">
  <!ATTLIST entry kind (term|topic) "term">
  <!ATTLIST entry value CDATA #REQUIRED>
  <!ELEMENT locators (locator*)>
  <!ELEMENT locator (#PCDATA)>
  <!ATTLIST locator kind (pages|xref) "pages">
  <!ATTLIST locator label (pg|see|seeAlso) "pg">
]>
```

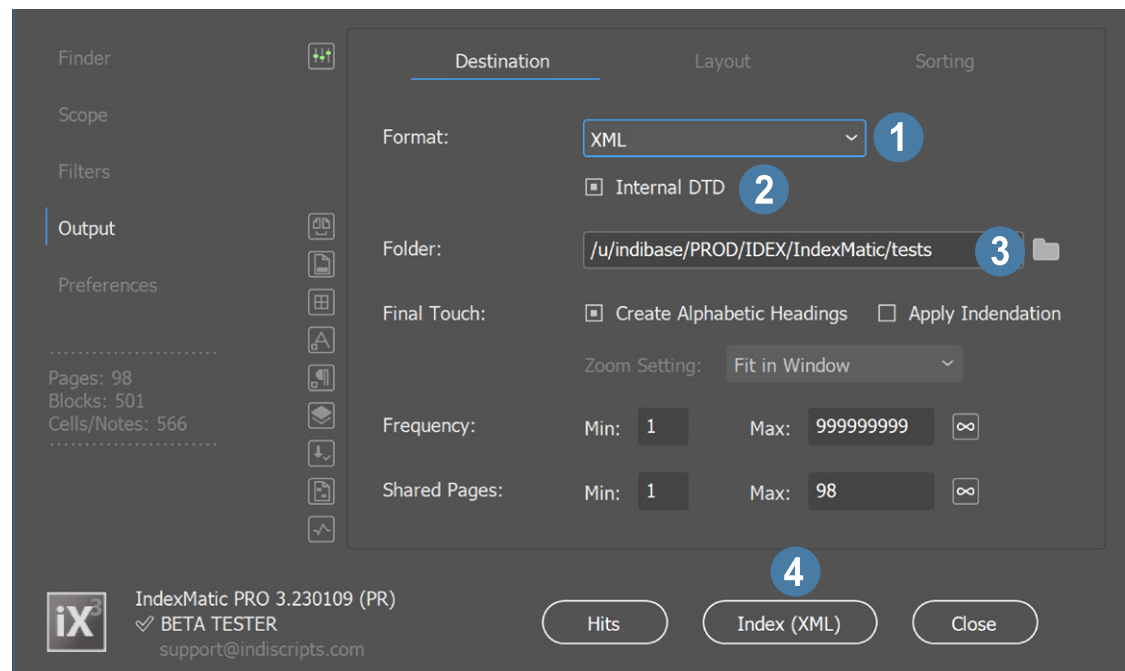
where each <entry> element represents either a terminal topic (TERM) or a parent TOPIC, while each <locator> element describes either a sequence of page numbers (kind="pages")

or a cross-reference (kind="xref"). The example below shows how these elements are arranged.

To generate your index as an XML file:

- 1) Activate **Output ► Destination** and select Format: **XML**.
- 2) Click **Internal DTD** to have the DTD included in the XML file.
- 3) In the **Folder** field, type or paste the path of the output folder. Alternatively, click the icon button to select a particular folder.
- 4) Click **Index (XML)**.

If the option *Create Alphabetic Headings* is active while exporting your index in XML, each base letter ('A', 'B', 'C...') will form a top-level topic (depth=0) having its own subtopics.



# Output and Reports

## ► *InDesign Document* .indd

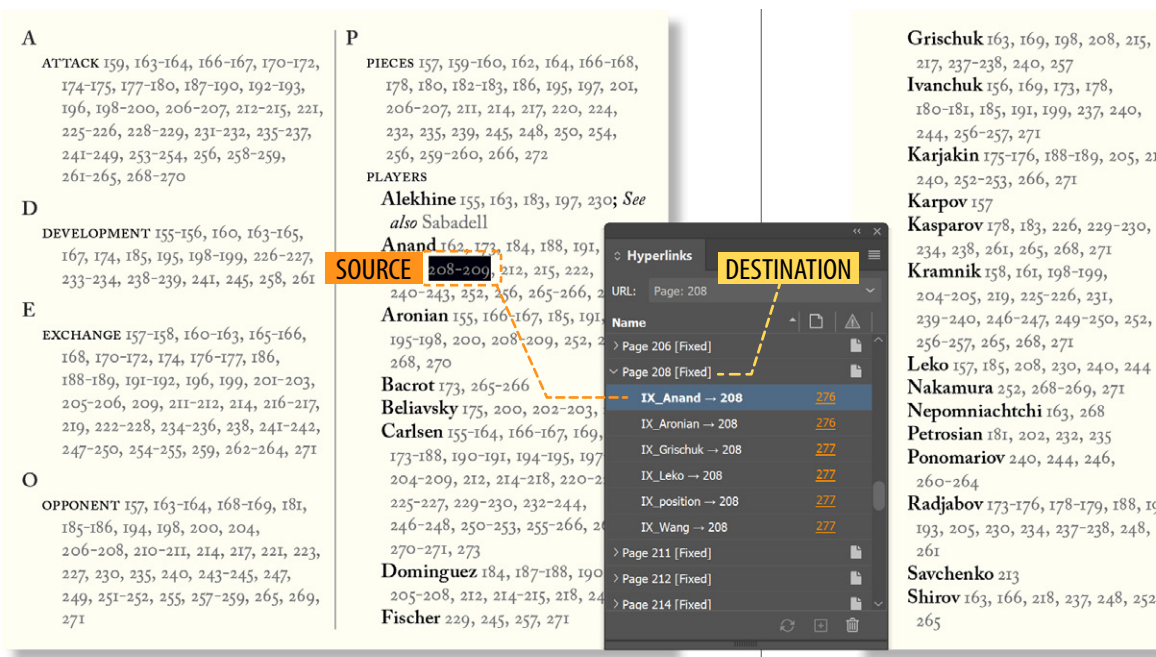
*InDesign Document* is the most advanced output format, it allows you to produce a full-featured, standalone document—or to add the index pages to the active book. Behind the scenes, IndexMatic<sup>3</sup> still relies on a transient IDMS snippet that contains the whole index, but it then builds the required InDesign components and links them to your project.

This mode is specially designed for creating **HYPERLINKS** before you export your document to an interactive PDF or ePUB. In the final index, each **PAGE LOCATOR** is translated into an hyperlink (source) and associated to the corresponding page (destination). As shown in the screenshot, IndexMatic<sup>3</sup> generates clean hyperlink names of the form

“IX\_<topic> → <page>”

so that you can easily check their consistency in the Hyperlinks panel. Shared page destinations are declared as long as the index points out to pages of the same document, external page destinations are declared otherwise.

**NOTE** When a locator contains a range (e.g. “123-129”) or a note reference (e.g. “123 n.7”), the hyperlink is made to point out to the first associated page (123).



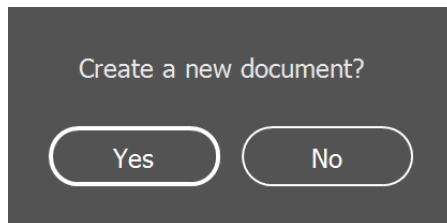
To generate your index as a separate InDesign document, or to include it in your current document/book with the ability to append hyperlinks:

- 1) Activate the **Output ► Destination** subpanel and select **Format: InDesign Document**.
- 2) Activate the **Create Hyperlinks** option if InDesign hyperlinks have to be associated to page locators at the end of the process.
- 3) In the **Folder** field, type or paste the path of the output folder. Alternatively, click the icon button to select a particular folder.

# Output and Reports



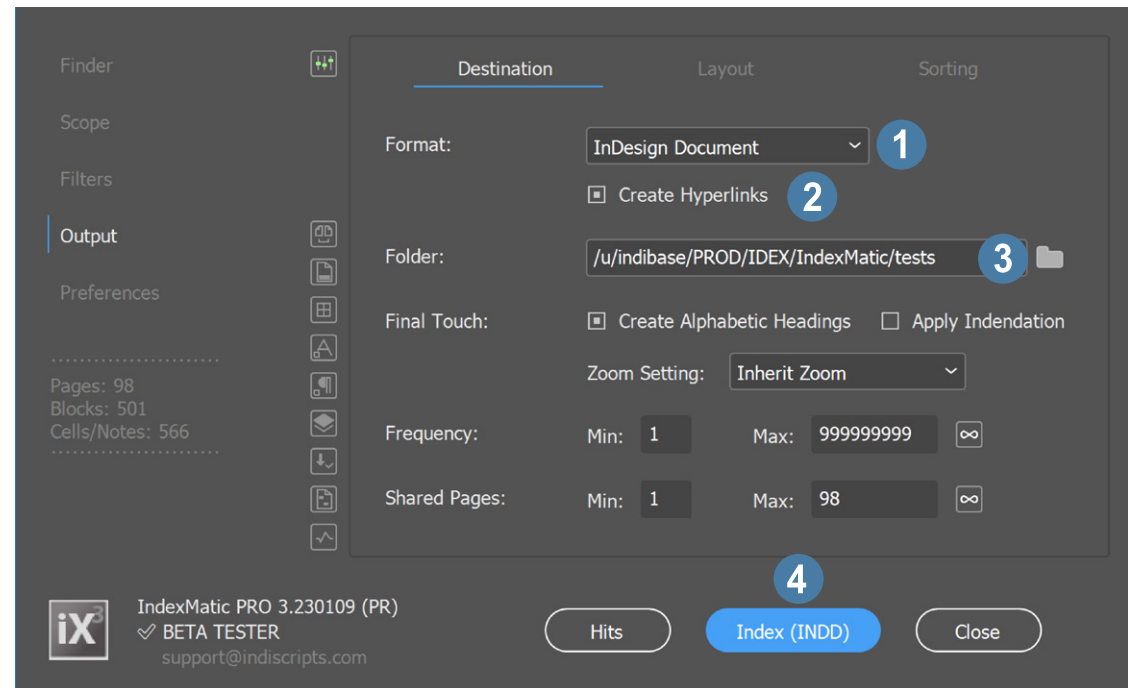
4) Click **Index (INDD)**. When the index is ready, IndexMatic<sup>3</sup> offers you the option to create a new document:



- Click **Yes** to generate a fresh InDesign document. If hyperlinks are required, all will be based on external page destinations.
- Click **No** to let the program import the index in your current project, that is, either the active document (if the scope has a single document) or the last chapter of the book (multi-document scope).

Whenever IndexMatic<sup>3</sup> is instructed to import the index in your project, it tries to determine the best document and page for that purpose. For example, if you are simply updating an existing index, the program will automatically recover the existing location and inject new data in place. In other cases, it may add a blank page at the end of the target document in order to create a free text frame for placing the index.

If the algorithm fails to determine a satisfactory solution, IndexMatic<sup>3</sup> populates the placegun with the index snippet so you can still place it manually.

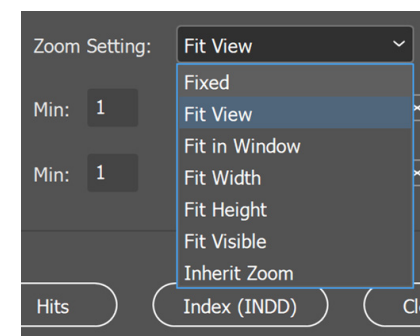


**NOTE** ⚠ Generating hyperlinks is a time-consuming process. If your index has thousands of page locators, expect the task to complete in several minutes.

## ► *Hyperlinks Zoom Setting* →

When the **Format** is set to *InDesign Document* and **Create Hyperlinks** turned on, the option **Final Touch**

► **Zoom Setting** is enabled. You can then select the view state of the page being jumped to (*Fixed, Fit View, Fit in Window...*) as defined in InDesign's Hyperlink dialog box.



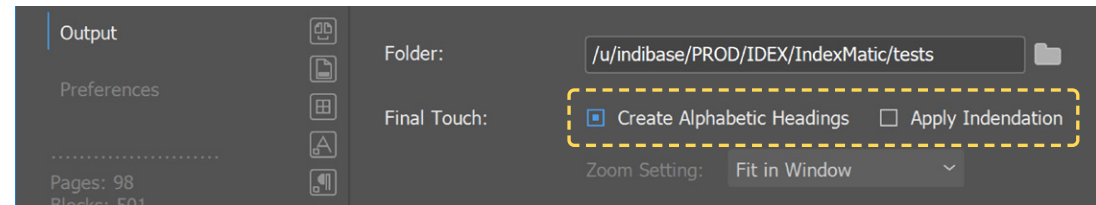
## 2. Adding Alphabetic Headings

In order to visually detach alphabetic group of entries and make your index easier to navigate, you might want to insert initial letters, or ALPHABETIC HEADINGS, at the beginning of each set thus formed:

A  
Ackerley ...  
Anson ...  
Archie ...  
B  
Bacchus ...  
Balzac ...  
C  
Caesar ...  
etc.

Although this could be (tediously) done by adding fake parent topics to your query list: `Ackerley => A > $`, `Bacchus => B > $`, etc., IndexMatic<sup>3</sup> offers the option to create those surrounding topics automatically.

- 1) Activate **Output** ► **Destination** and select the desired format.
- 2) In **Final Touch**, click *Create Alphabetic Headings*. Those headings are treated as ancestor topics, so all existing topics declared in your query list will be grouped under these elements.



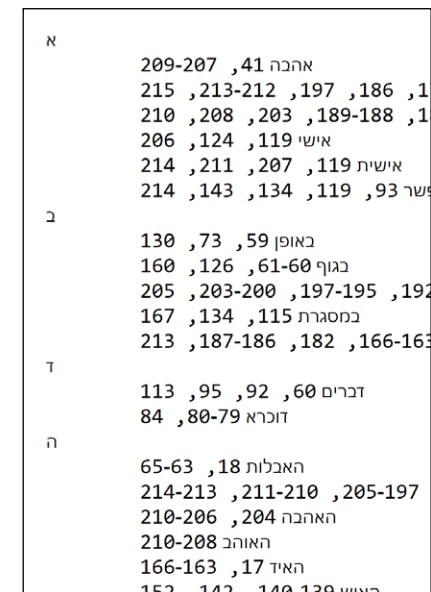
- 3) Optionally, click *Apply Indentation* to get regular topics shifted one level deeper. →

If the active alphabet contains ligatures (œ, Æ, Nj...) or diacritical elements (ç, é, ü...), such graphemes are attached to a BASE LETTER according to the selected sorting language (see **Output** ► **Sorting**).

In most Latin languages, œ is associated to the base letter O, ç to C, etc. But IndexMatic<sup>3</sup> properly deals with specific languages that may regard a combined form as a distinct letter, e.g. æ in Danish, ll and ñ in Spanish, c'h in Breton, r' in Ukrainian, lj in Bosnian, etc.

**NOTE** IndexMatic<sup>3</sup> also detects digits and symbols outside of the regular alphabet. If such characters appear at the beginning of a term and are not bypassed due to sorting options, the corresponding expressions are grouped under the heading #.

**NOTE** ⚠ Alphabetic headings are not generated if sorting is inhibited due to the option *Keep Original Order* ↓.



Example of alphabetic headings with *Apply Indentation* enabled. (Note that in Hebrew, text and index entries are right-to-left directed.)



## 3. Filtering Index Data by Statistical Criteria

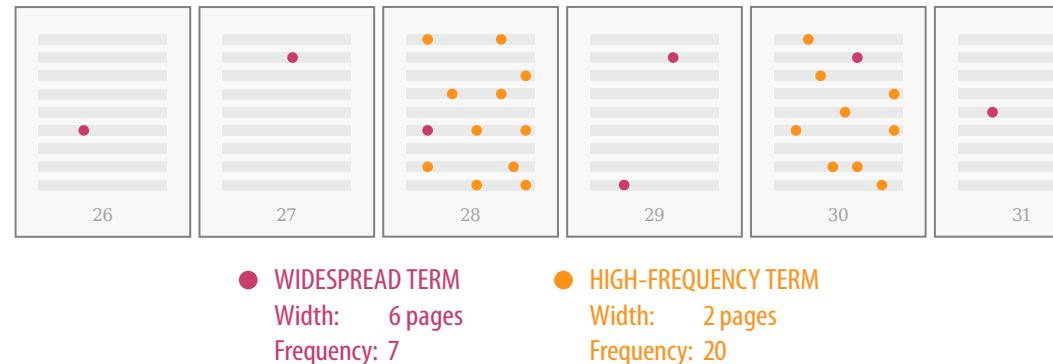
A very common problem in automatic or semi-automatic indexing is balancing the volume of information. An index with too many entries, or too many locators per entry, is unusable in practice because the reader derives no benefit from exploring such a large number of references.

Increasing the page rank (**Finder ▶ Options**) is the most effective and immediate way to limit the inflation of page locators around a particular keyword. This procedure is carried out upstream by the query engine. It works very well but cannot act *globally* on the set of indexed topics.

**NOTE** A Page Rank condition only decides whether a given expression is relevant *locally* (that is, on a page scale).

To supplement this tool, IndexMatic<sup>3</sup> introduces two statistical filters, independent of the search engine, which determine whether a TERM (i. e., a target topic) should be kept in the final index:

- The FREQUENCY of a term refers to the overall number of its occurrences (also referred to as HITS). This count reflects exactly how many times the program has encountered or produced this particular expression while running the queries.
- The SHARED PAGES number (also referred to as the term WIDTH) indicates how many different pages are associated with it.



The figure above illustrates an important point: a term widely used in the book (●) does not necessarily have a high frequency. Conversely, a high frequency term (●) may be concentrated on only a few pages.

Of course there is on average a correlation between frequency and width: the most frequent words tend to appear on more pages, and the words sharing more pages tend to appear more frequently! However, when extreme values are reached (very high or very low), the combination of these two numbers can become significant.

For example, a term of maximum width (that is, present on almost all pages) whose frequency is still five to ten times higher, is most likely irrelevant to your index. Such ubiquity makes it meaningless as it reduces it to a noise word, like “the” or “for”. Now, if its frequency is of the same order of magnitude as its width, say in a ratio of 1 to 2, then such term is probably an essential keyword, and you will need to adjust its Page Rank to make your index focus on page locations where it really matters.

Obviously, the frequency is always greater than or equal to the width.



# Output and Reports



On the other hand, an extremely low-frequency term is generally irrelevant—unless you are looking to identify *hapaxes*, which often reveal typos or spelling inconsistencies. But a term identified on quite a few pages (i.e., of small width) which nevertheless has a decent or relatively high frequency, certainly represents an important topic deserving of being reported, especially since it is concentrated in just certain places in the book.

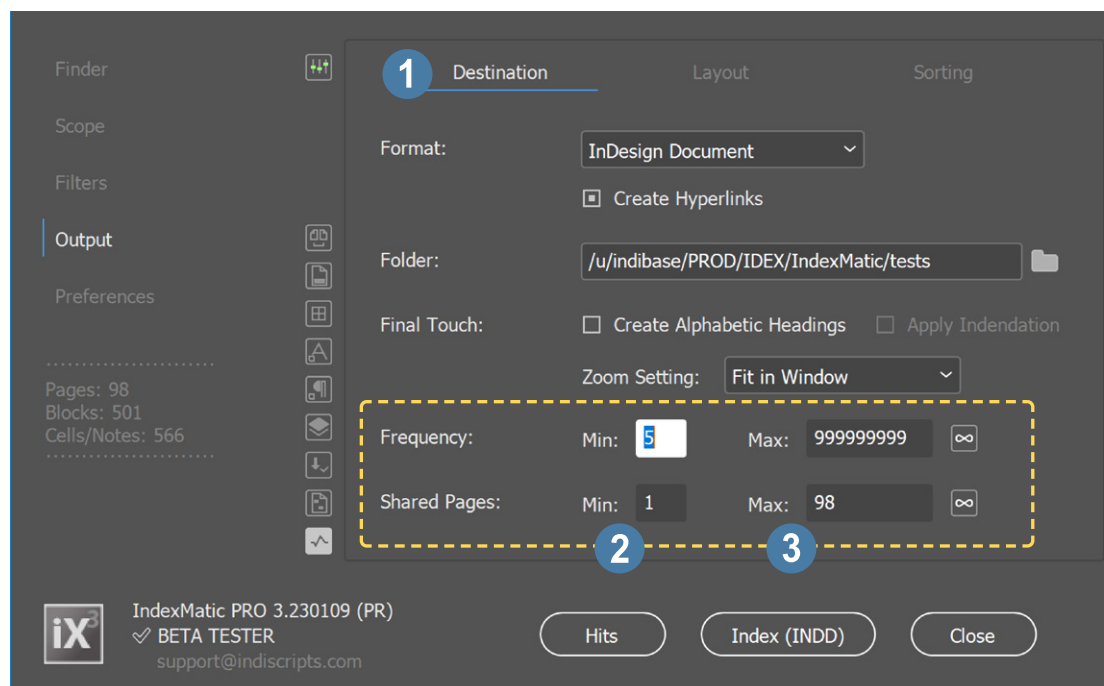
In summary, FREQUENCY and SHARED PAGES can be great criteria for improving an overcrowded index. Depending on the number of pages of your book, its particular structure, its semantic field, it is up to you to determine how best to exploit those features.

IndexMatic<sup>3</sup> allows you to set a minimum and maximum value for each parameter. Below and above these limits, a term will be rejected from the index.

**NOTE** Both the **Frequency** and the **Shared Pages** conditions must be satisfied. To inhibit one and/or the other, simply restore the minimum and maximum values.

To filter out terms with respect to FREQUENCY or WIDTH:

- 1) Activate the **Output** ► **Destination** subpanel and go into the **Frequency** (resp. **Shared Pages**) area.
- 2) In the *Min* field, enter the minimum frequency (resp. width) that any indexed term must have. You can use the arrow keys ↑ or ↓ to increase/reduce the number. The default value is 1 for both



parameters. 0 is specially allowed as a frequency, meaning that IndexMatic<sup>3</sup> will also report NOT FOUND ENTRIES. →

- 3) In the *Max* field, enter the maximum frequency (resp. width) that any indexed term can have. You can use the arrow keys ↑ or ↓ to increase/reduce the value. The default maximum value is determined automatically and can be reset using the ∞ button.

**NOTE** Obviously, the maximum number of Shared Pages cannot be greater than the total number of pages in the entire scope.

**A NOT FOUND ENTRY** is a term without match. Some query is expected to identify it, e.g. `/sophis/w=>sophism` but no corresponding key was found in the scope. In such case the topic “sophism” should not appear in the index, unless you set the minimum Frequency to zero. (No actual page locator can be associated with it though.)

## 4. Setting Up Separators and Delimiters


The **Layout** subpanel allows you to precisely define the formatting characters used in the final index. It controls how pages will be grouped under an index entry, how to represent locators, ranges, cross-references, etc.





The **Separators** area contains all essential elements you may like to customize. Simply click the central button (SELECTOR) of an element to redefine the corresponding character(s).

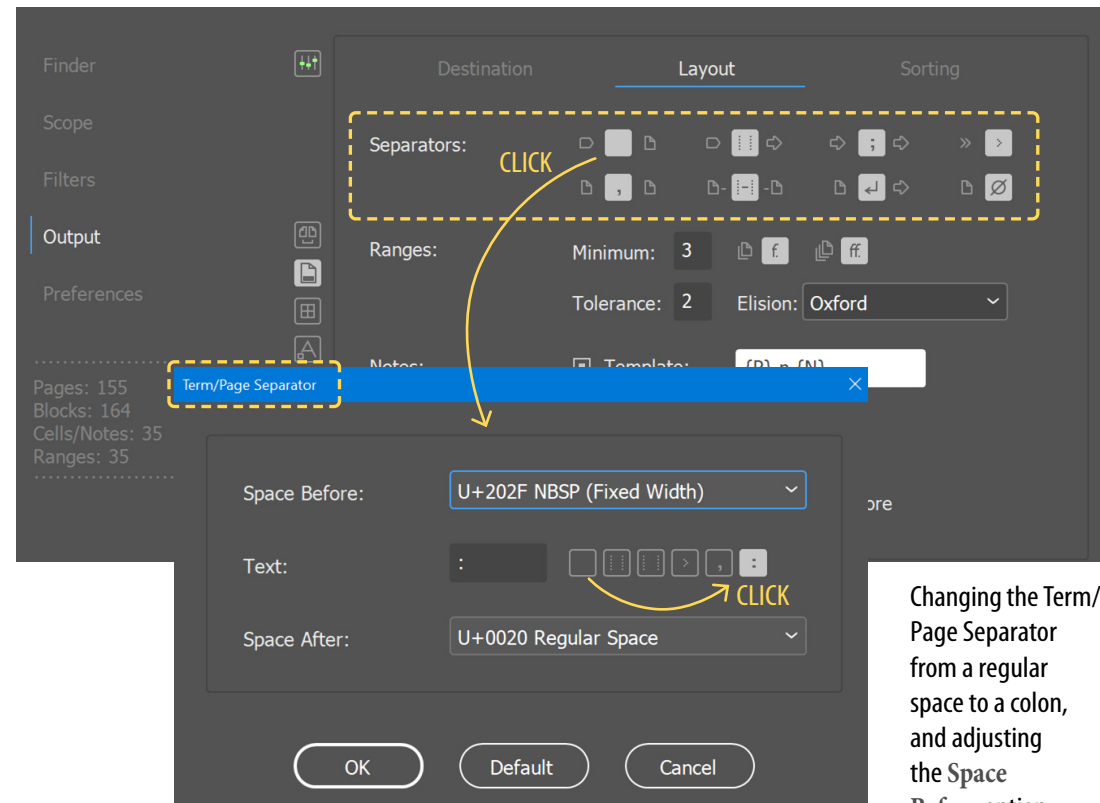
► **Term / Page Separator** e.g. SPACE 

By default a basic index entry consists of a term (i.e. target topic), a regular space character, and a sequence of page numbers:

Plato , 31-35, 48

In IndexMatic<sup>3</sup> interface, the space separator is symbolized by a solid blank square . To change it, click the selector and choose a different symbol in the popup dialog (see screenshot).

- The **Text** area shows the active separator. A few alternative symbols are available to the right of the input field, e.g. en space , em space , tab , colon .
- You can either click any of them or enter custom characters in the edit box. At most three characters are supported here (not counting the *space before* prefix and the *space after* suffix).



Changing the Term/ Page Separator from a regular space to a colon, and adjusting the Space Before option. (All selectors of the Separators area work the same way.)

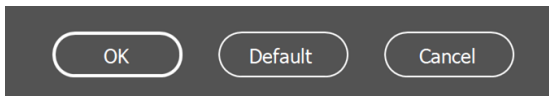
- The **Space Before** (resp. **Space After**) drop-down list controls which specific space must be inserted before (resp. after) the separator. Many options are available, from *[No Space]* to any special space character that InDesign supports.

**NOTE** **Space Before** and **Space After** are automatically preset depending on the active symbol. In the figure above, the prefix was manually changed into U+202F (although the colon expects no space before in most languages).

# Output and Reports



- In the selector dialog, click **OK** to confirm your choice, **Default** to restore the default settings, **Cancel** to discard your input.



**TIP** Non-printable characters like U+000A (forced line break), U+2003 (em space), etc., can be specified by entering their *uHHHH* code in the Text field.

► *Term / Cross-Reference Separator* *e.g. EN SPACE*

This selector controls which separator must be used between a term and a “See” reference in entries like

Plato See Socrates

Typical options are regular space, tab, em space, en space (similar to the Term / Page Separator). A special character such as right indent tab (U+0008) might be specified as well.

► *Cross-Reference Separator* *e.g. SEMICOLON*

A query list may attach multiple “See” or “See also” references to a particular term, e.g.

Plato *See* Idealism; Problem of universals; Socrates



The Cross-Reference Separator specifies how to format these consecutive elements. Typical options are semicolon, comma, slash, bullet, middle dot...

► *Indent Character* *e.g. TAB*


Multi-level indexes (involving alphabetic headings and/or regular topics) require a consistent indentation scheme; this greatly helps readers find their way around the topic tree.

- If you generate the index as an InDesign snippet or document, the question can be entirely addressed from the special paragraph styles *H0*, *H1*, *H2*... created by IndexMatic<sup>3</sup>. All you need to do is adjust the Indents and Spacing attributes to your layout. In such a case, you


Some built-in punctuation symbols available in the Layout panel, and their associated code point in *uHHHH* form.

# Output and Reports

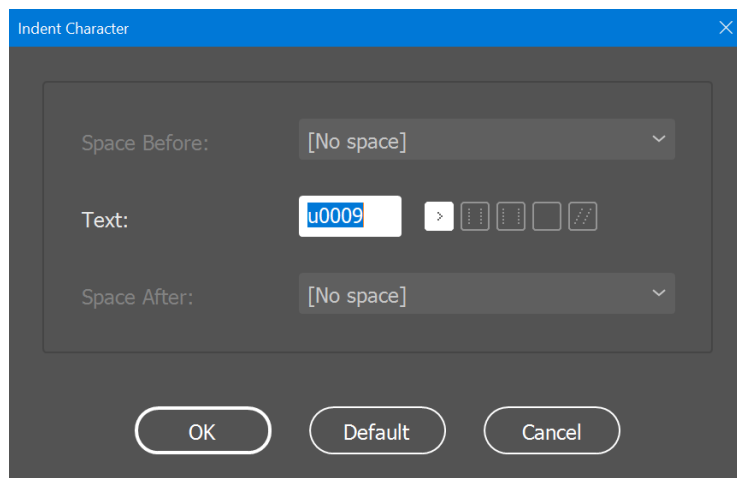


probably don't need an explicit indent character at all, so you will set the selector to , which represents no character.

- Otherwise, specify a non-empty indent character (or string) that clearly marks level separations. A tab character is generally well rendered in simple text editors. Alternatively, enter a sequence of two or three regular spaces, or any *fixed-width* space (en space, em space...)

**NOTE**  In XML output mode, all “structural” separators are ignored since their role is fulfilled by the markup language.

As shown below, the **Space Before/After** options are disabled in the **Indent Character** dialog—because they would not make sense—but you can still fill in the **Text** field as already discussed in the previous pages.



## ► Page Separator e.g. COMMA

This selector controls which separator must be used for grouping PAGE LOCATORS (i. e. page numbers and ranges).

Plato 12, 31-35, 48, 53-57

The most common option is the comma (with no space before and a regular space after) but you can specify any other character.





**NOTE** The page separator is visible in all output modes: plain text, InDesign snippet and even XML (the `<locator>` element then contains the sequence of page numbers formatted with respect to your custom separator.

```
<locator kind="pages" label="pg">97</locator>
/locators>
ntry>
try depth="0" kind="term" value="déplacement">
locators>
<locator kind="pages" label="pg">145 ; 146 ; 149</locator>
/locators>
ntry>
try depth="0" kind="term" value="description">
locators>
```

## ► Page Range Delimiter e.g. NON-BREAKING HYPHEN

This selector is specifically used for formatting page ranges:

Plato 12, 31-35, 48, 53-57

The recommended option is the non-breaking hyphen, symbolized by , as it maintains every page range on a single line. In plain text output format, a basic hyphen-minus  (U+002D) might be used instead, although the Unicode hyphen  (U+2010) is often considered more *typographically correct*. Another common option is the en dash . This delimiter will be visible in XML export too.

XML export of a French index using a custom page separator.

# Output and Reports



► **Page/Cross-Reference Sep.** *e.g. END OF PARAGRAPH*

This selector controls which separator must be used between a sequence of page locators and a “See also” reference in entries like

Plato 12, 31-35, 48, 53-57  
See also Socrates

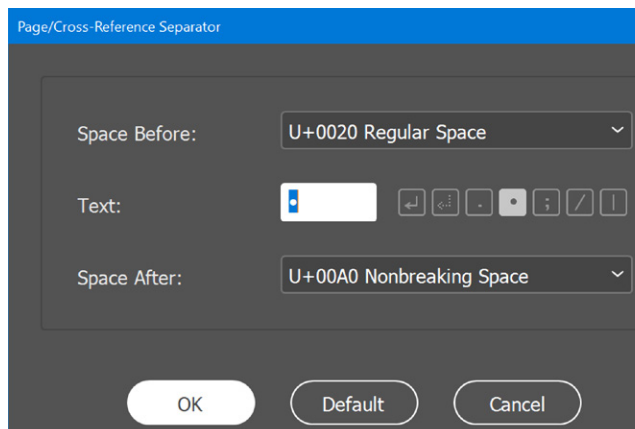
In the above example, a new line is created (with some extra indent) due to the separator (end of paragraph), the option *Indent more* being applied to “See also” references. We could have specified a forced line break as well.

**NOTE** Whether a cross-reference is regarded as a “See” or a “See also” element entirely depends on the attachment of page locators with the considered topic. Hence there is no confusion between the Page/Cross-Ref selector in question here, and the Term/Cross-Ref selector previously described.

A more compact layout is still possible, e.g. →

Plato 12, 31-35, 48, 53-57 • See also Socrates

by specifying a separator (bullet, em dash, or any suitable symbol) that does not consume a new line. The below screenshot shows settings for getting inline “See also” references introduced by a bullet.



► **Final Character** *e.g. EMPTY*

A full index entry—whether it contains page locators, cross-references, or a mixture of the two—usually ends without any termination character. But you might want some explicit markup in your layout, e.g.

Philosophers  
Plato 12, 31-35, 48, 53-57 ; See also Socrates.  
Socrates 17, 29-30, 66.

The Final Character selector, if not set to the empty element , may specify any punctuation mark: full stop, pilcrow sign, bullet, etc.

Note that such custom character only regards lines that contain locators or references. Alphabetic headings or parent topics won't be affected by this setting.



**P**  
**PIECES** 157, 159-160, 162, 248, 250, 254, 256, 259-260, 266, 272.  
**PLAYERS**  
**Alekhine** 155, 163, 183, 197, 230; *See also* Sabadell.  
**Anand** 162, 173, 184, 188, 191, 198, 204, 208-209, 212, 265-266, 268.  
**Aronian** 155, 166-167, 185, 191, 195-198, 200, 208-209, 252, 256.

## 5. Fine-Tuning Page Ranges

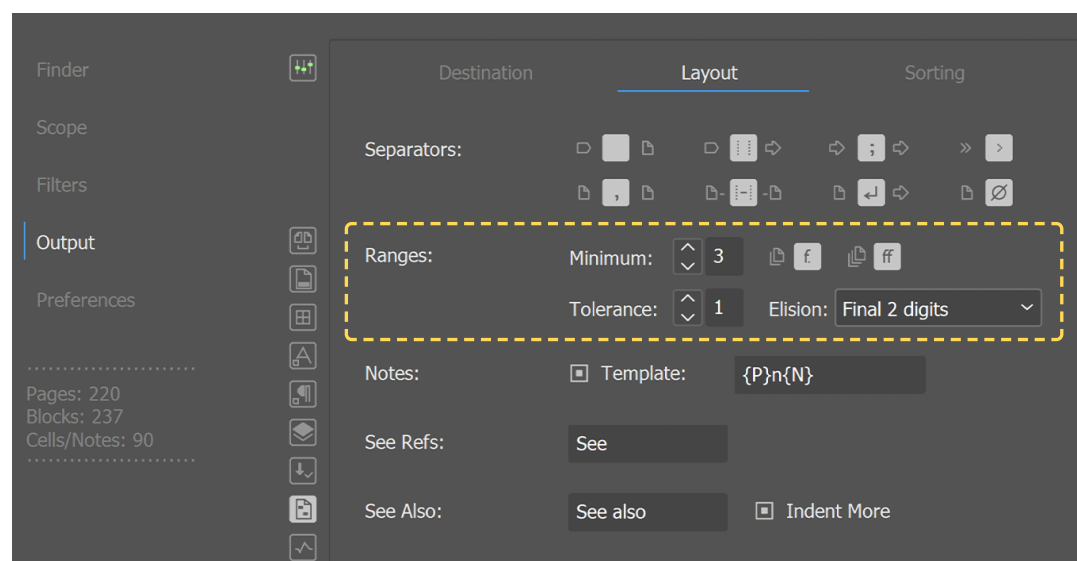
**DEFN** From an indexing point of view, a range is strictly equivalent to the comprehensive sequence of pages it denotes; we only use it to condense index entries that may involve many page numbers.

The way we make up and format numeric intervals is surprisingly varied across cultures. Basically, a PAGE RANGE consists of two locators (mostly numbers) joined by a punctuation mark, e.g. 123-127, and this expression refers to the closed interval associated to these min-max boundaries. In our example, the range 123-127 concisely denotes a sequence of five pages: 123, 124, 125, 126, 127.

**NOTE** The separator involved in formatting a range (usually a hyphen or dash) can be freely modified using the Page Range selector, in the Separators area.

Since pages are numbered in ascending order—including in prefaces or annexes having a special numbering like *i*, *ii*, *iii*, *iv*...—we must make sure that both ranges and individual pages are assembled in ascending order and in a coherent fashion for all index entries.

When distinct systems coexist, as in “viii, ix, x, 3, 4, 5”, we are not allowed to reduce the set to a single range (*viii-5*). It doesn't matter that all these pages actually follow each other in the book, two ranges are still needed: *viii-x, 3-5*. But are they even legit?



Some publisher guidelines or typographical rules advise against forming ranges involving only two or three pages, arguing that the gain in space is negligible between, say, 11-13 and 11, 12, 13. Other conventions call for specific formatting of pairs and triples: the sequence 34-35 is just written 34f., and 34-36 becomes 34fff.

Conversely, one can save space by declaring ranges that do not *exist* completely. For example, “10, 12, 13, 15, 16” is almost 10-16 if we neglect the missing numbers (11 and 14). IndexMatic<sup>3</sup> offers you the option to fill in these tiny gaps when more compactness is desired.

Finally, a range may be *elided* with respect to its most significant digits or according to more subtle rules. For example, English



# Output and Reports

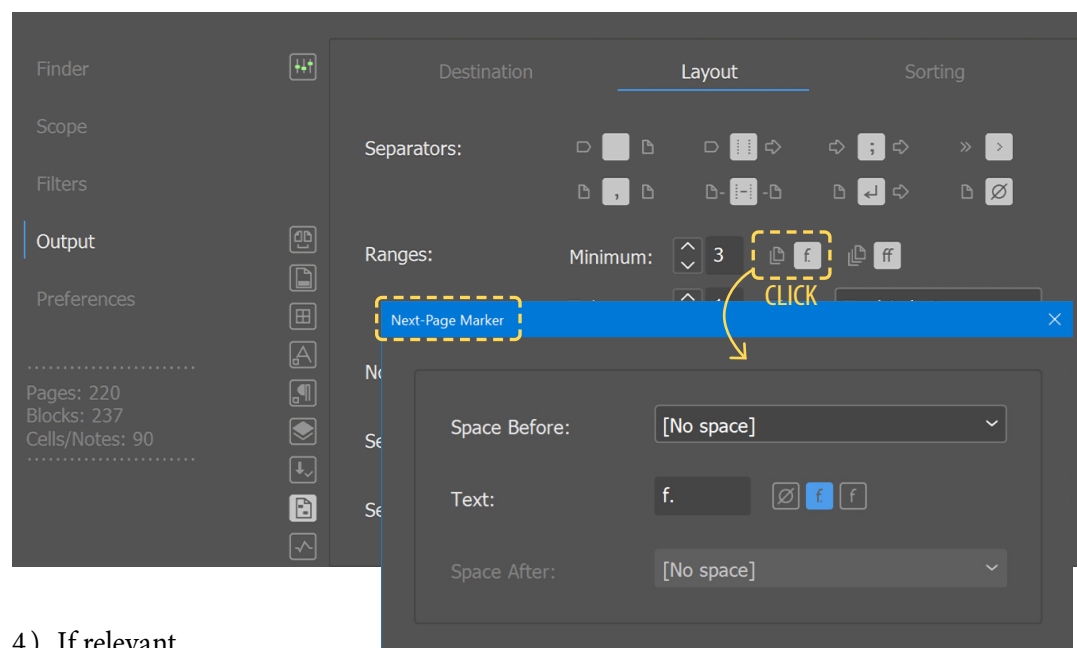


readers understand 137-9 as denoting the range 137-139. This abbreviated form is not generally accepted, however, in European books.





IndexMatic<sup>3</sup> provides all the options for rendering page ranges according to your own rules.

- 1) Activate the **Output ▶ Layout** subpanel and go into the **Ranges** area.
- 2) In the *Minimum* edit box, enter the minimum distance between the first and the last number in a range. The default value is 1, meaning that two consecutive page numbers are enough to constitute a range. If instead you want to allow ranges from three consecutive numbers, set the option to 2. (In that case the sequence “12, 13” will be printed as it is, while “12, 13, 14” will reduce to 12-14).
- 3) In the *Tolerance* edit box (whose default value is 0), indicate how many missing numbers are allowed when the program creates a range. By default, IndexMatic<sup>3</sup> strictly reflects page locations as they have been collected, for each entry, by the query engine. If a term was found only on pages 21, 23, and 25, no range is supposed to come out. But if one (1) missing number is allowed, then “21, 23, 25” will be shortened to 21-25.

**NOTE** In case of conflict, the *Minimum* option supersedes the *Tolerance* option. For example, if any range must target at least six numbers, the fake range 21-25 won't replace 21, 23, 25.



4) If relevant,

- Click the Next-Page Marker selector  to define a suffix indicating a sequence of two pages, e. g. 31f. instead of 31, 32. When turned on, this option forcibly increases the *Minimum* to prevent the program from creating regular ranges of two elements. To cancel this special formatting, reset the selector to .
- Click the More-Pages Marker selector  to define a suffix indicating a sequence of three pages, e. g. 31ff. instead of 31, 32, 33. When turned on, this option forcibly increases the *Minimum* to prevent the program from creating regular ranges of three elements. To cancel this special formatting, reset the selector to .

The abbreviations **f.** (next page) and **ff.** (several consecutive pages) are commonly used in German and Nordic books. (My thanks to my colleague **Roland Dreger** for drawing my attention to this feature.)

# Output and Reports



5) Click the *Elision* drop-down list to select the rule applied to elide page ranges:



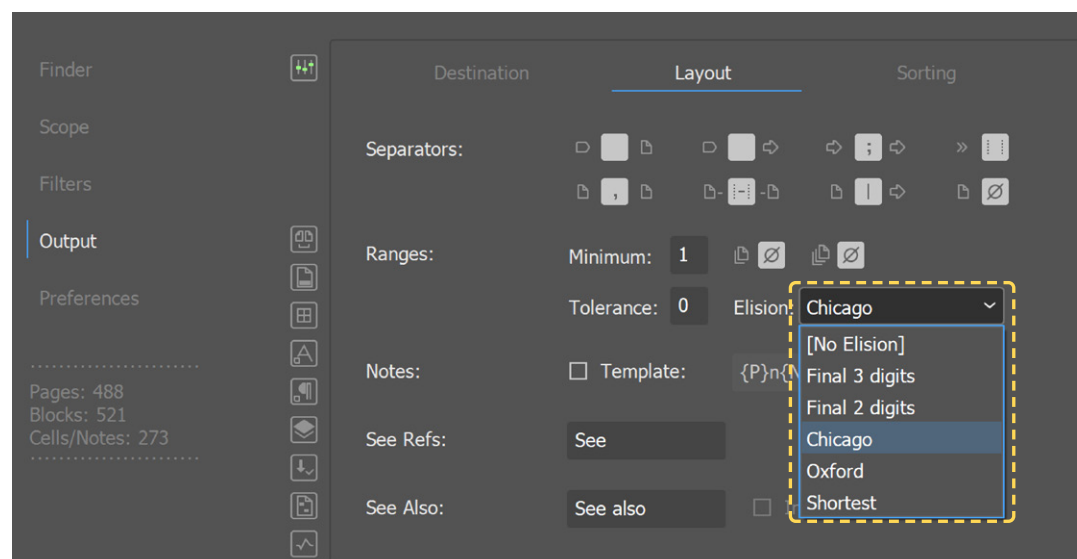
- *[No Elision]* Never shorten a range.  
E.g. 10-12, 23-25, 345-349, 1234-1237, etc.
- *Final 3 digits* Preserve at least three final digits.  
E.g. 42-45, 100-102, 1021-1023, 1491-560, 10150-302

**NOTE** 1021-1023 cannot reduce to 1021-023 because 0 cannot introduce an elision.

- *Final 2 digits* Preserve at least two final digits.  
E.g. 42-45, 100-102, 210-11, 352-65, 1021-23

**NOTE** 100-102 cannot reduce to 100-02 because 0 cannot introduce an elision.

- *Chicago* Comply with the *Chicago Manual of Style* rules.  
E.g. 10-12, 42-45, 100-102, 103-7, 132-36, 1021-23  
For more detail see → “[Number-span elision](#)”, by Peter Kahrel
- *Oxford* Minimal form except if final digits range in 10..19.  
E.g. 10-12, 42-5, 100-2, 132-6, 210-11, 1021-3
- *Shortest* Strict minimal form.  
E.g. 10-2, 42-5, 100-2, 210-1, 352-65, 1021-3  
⚠ This scheme is to be used only in special circumstances since 10-2 is usually an invalid elision.



## 6. Including Note Markers in the Index

If your SCOPE is allowed to span footnotes and/or endnotes, IndexMatic<sup>3</sup> can identify index entries that emanate only from these particular regions.

For example, suppose the key */holism/* has a match on page 89, note 17, and doesn't appear anywhere else on that page. It might then be nicer to render the locator as follows,

holism 89n17

or by any similar convention: “89n. 17”, “89 [17]”, “89:17”. The *Template* checkbox lets you activate and specify this option:

# Output and Reports



- 1) Activate the **Output** ► **Layout** subpanel and go into the Notes area.
- 2) Click the *Template* checkbox (if unchecked).
- 3) Enter your custom page/note template: the code “{P}” represents the page number and “{N}” represents the note number. Other characters (including inner spaces) will be rendered as typed. Here are some examples:

{P} n.{N} (default template)	→	89 n.17
{P}n{N}	→	89n17
{P}:{N}	→	89:17
{P}[[{N}]]	→	89[17]

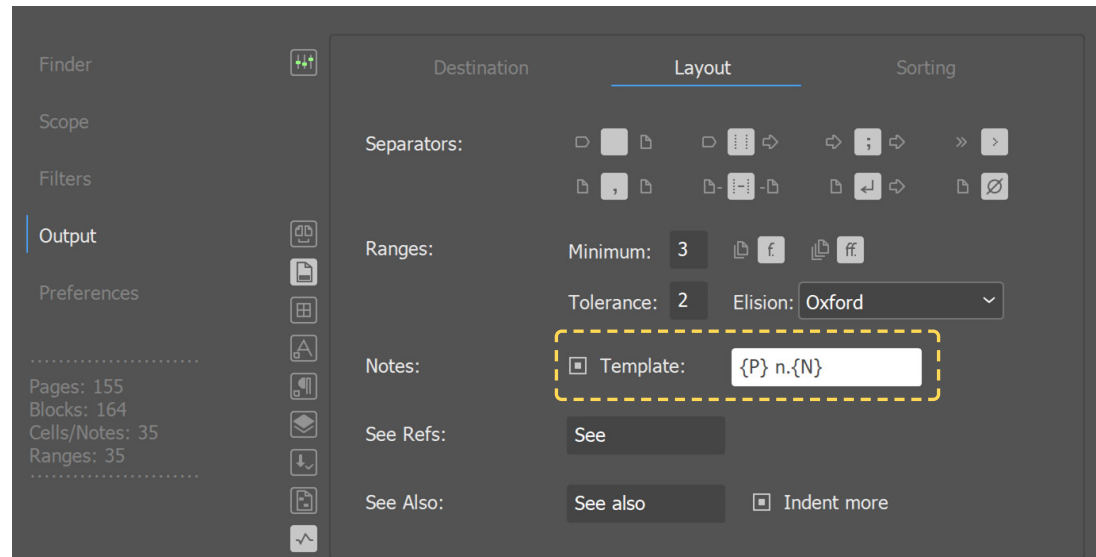
**TIP** When templating page-note locators, keep in mind that they should fit into larger sequences of page numbers and ranges, as in

holism 31, 45-47, **89n17**, 102 — *See also* Jan Smuts

So make sure your pattern combines with other separators without creating confusion.

Just as IndexMatic<sup>3</sup> respects the various modes of page numbering while forming simple page locators (4, 004, iv, d), it also outputs note references as defined in InDesign’s Document Footnote (resp. Endnote) Options dialog.

To inhibit all note markers, simply uncheck the *Template* box.



## 7. “See” and “See also” Markers

By default, See and See-Also cross-references are introduced by the expressions “*See*” and “*See also*”, respectively (in the English locale). You can of course specify different expressions or even symbols, e.g. a rightwards arrow “→” for both.

- Just go into the **See Refs** (resp. **See Also**) edit field and enter the desired, case-sensitive text. (A regular space after is assumed if not entered.)
- In **See Also**, click the *Indent more* checkbox to impose an extra level of indentation on these cross-references (relative to page sequences).

The “*Indent more*” option is only available when the Page / Cross-Reference separator introduces a new line.

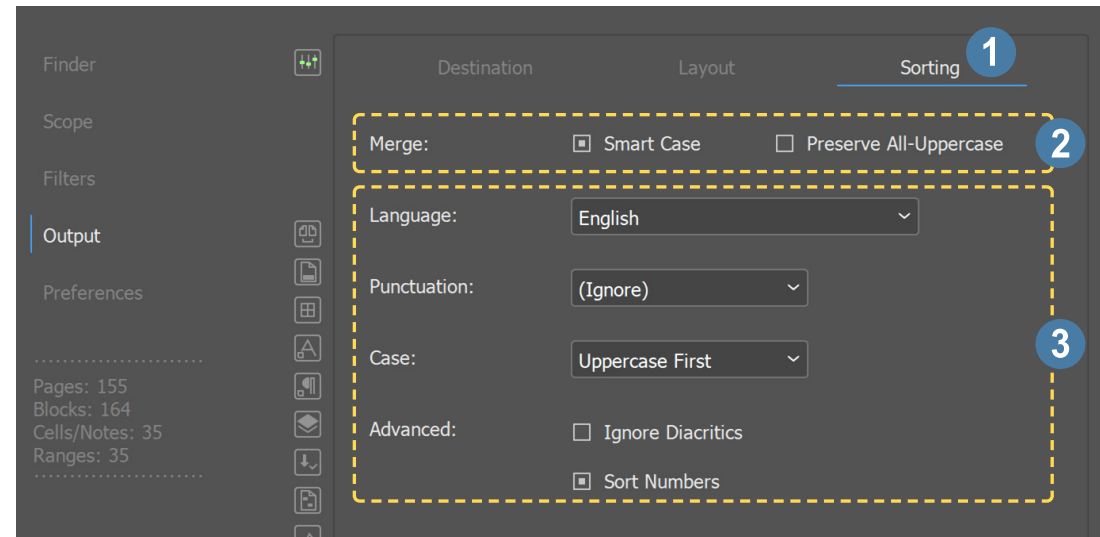
## 8. Sorting Index Entries

The **Sorting** subpanel is all about grouping and alphabetizing your entries. This is the last-but-not-least step in producing a clean index. Headings and topics must be consistently ordered so that readers can easily explore the content.

⚠ First, remember that you can completely bypass sorting if you supply a QUERY LIST that happens to be already ordered as expected. In that particular case, activate the **Finder** panel in Query List mode and switch on the Keep Original Order button . All sorting preferences are then ignored (and alphabetic headings won't be generated).

In most cases, however, you will let IndexMatic<sup>3</sup> do the work for you, either because the indexed terms are not known in advance (if they result from global regex queries, transformations and directives...) or simply because manual sorting becomes tedious when you have to manage lexical subtleties, sublevels, language-related conventions, and so on.

- 1) Activate the **Output ▶ Sorting** subpanel.
- 2) In the **Merge** field, turn on the *Smart Case* option if your queries are case-insensitive, and case variants (“recipe” vs. “Recipe”) have to be merged under a single term. This is usually what you want. When Smart Case is active, IndexMatic<sup>3</sup> detects expressions that could be merged and select the *predominant form*, that is, the one that appears most often in the book. The sub-option



*Preserve All-Uppercase* prevents the Smart Case feature from merging acronymic forms (“AIDS”, “TED”, “POTUS”) with possible homonyms (“aids”, “Ted”, “potus”).

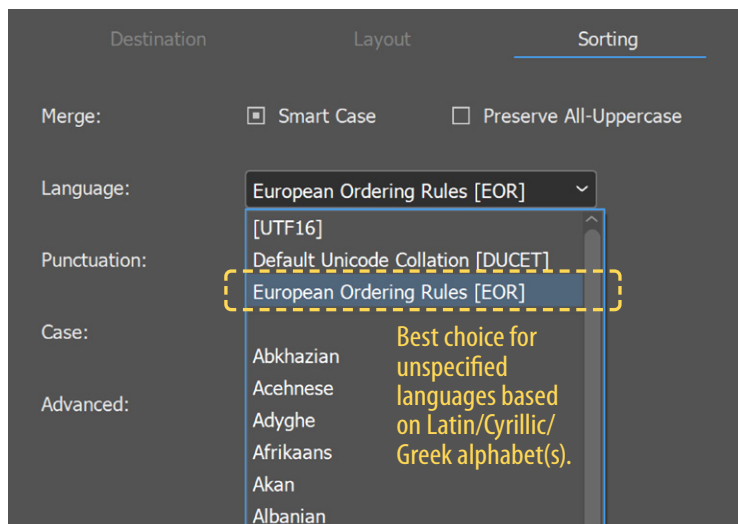
- 3) The next options (**Language, Punctuation, Case, Advanced**) define the input parameters of the COLLATION ALGORITHM. They finely control the rules applied while *alphabetizing* index entries.

- In the **Language** drop-down list, select the language or system that fits best the expressions to be sorted. About 230 languages are available here, including special systems like *Dutch (phonebook)*, *German (phonebook)*, *Norwegian Bokmål* vs. *Norwegian Nynorsk*, etc. In principle, the sorting language should be made consistent with the **Alphabet** and **Stop Words** selected in the **Finder** panel.

You can also control the case upstream by using term FORMATTERS:  
... => ^\$1~

⚠ In some environments, InDesign does not fully display the drop-down list and you cannot reach the last items. In such an event, click the **Language** caption and type the initial letter of the item until it appears.

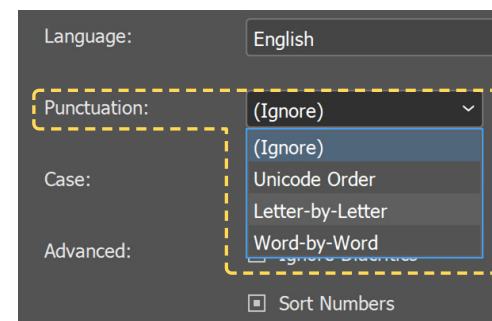
# Output and Reports



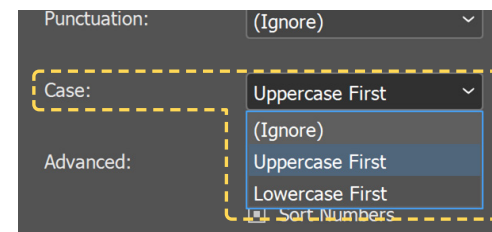
However, if your index has multilingual elements (or if no language fits your needs), *European Ordering Rules [EOR]* is usually the optimal solution for topics that only include Latin/Cyrillic/Greek letters. Choose *Default Unicode Collation [DUCET]* if foreign alphabets are in use and should be treated as specified in the [Default Unicode Collation Element Table](#). Finally, if your index entries do not depend on any specific language (ASCII labels, alphanumeric codes, etc.), the *[UTF16]* ordering system may be chosen.

**NOTE** *[UTF16]* is the fastest ordering mechanism, as it just rely on character code points and make no assumption about letters, words, languages. When it is selected, other sorting preferences (*Punctuation*, *Case*...) are disabled.

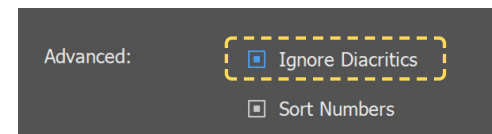
- Punctuation marks—more generally referred to as *variable collation elements*—can be entirely or partially disregarded while sorting entries. Go into the **Punctuation** drop-down list and select the desired policy: (*Ignore*), *Unicode Order*, *Letter-by-Letter*, or *Word-by-Word*. How these options work is discussed in depth in the [SmartSort Manual](#).



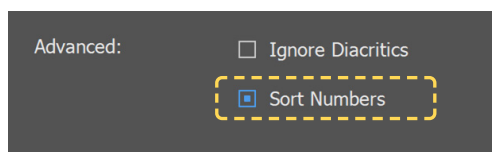
- The **Case** option strictly determines how to order case variants of a given expression (“Bill” vs. “bill”). Select *Uppercase First* to make uppercase letters come first (ball, **B**ill, **b**ill, bull), *Lowercase First* to make lowercase letters come first (ball, **b**ill, **B**ill, bull). You can also select (*Ignore*), which slightly speeds up the sort if no case variants are expected at all—for example, when the *Smart Case* feature is active.



- Check **Advanced** ► *Ignore Diacritics* to prevent the collation algorithm from ordering diacritical variants of a given expression (“Èrès” vs. “Ères”). Most languages don’t require such refinement so here again you can lighten IndexMatic<sup>3</sup> workload. For more details on this parameter, please refer to the [SmartSort Manual](#).

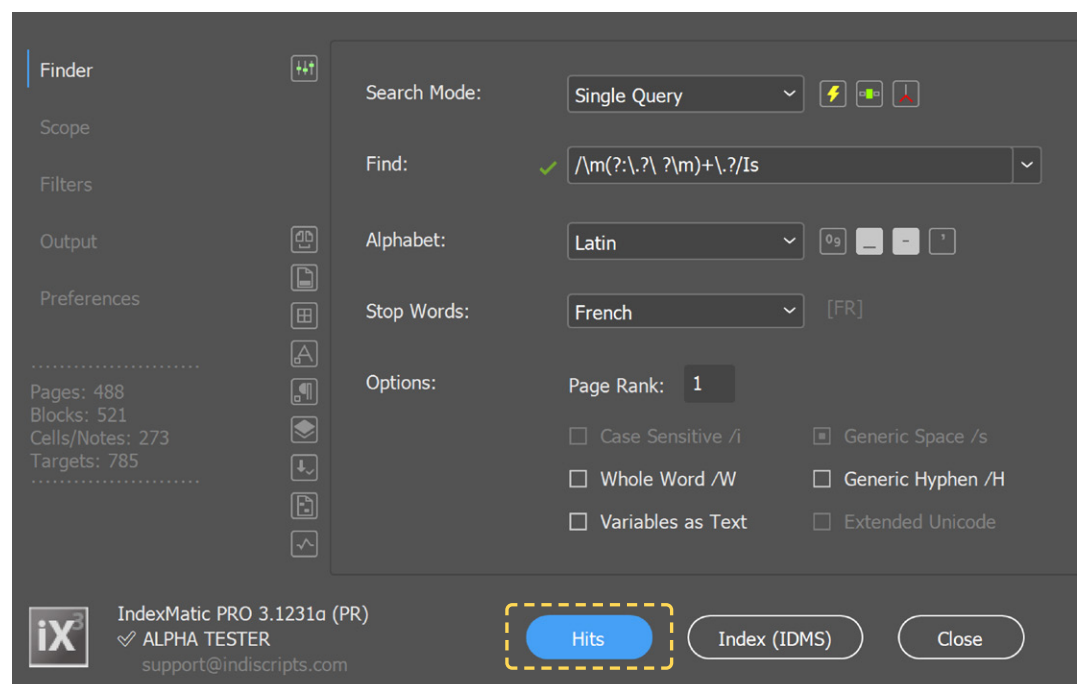


# Output and Reports



- Check **Advanced** ► *Sort Numbers* to apply a numeric sort to sub-expressions entirely formed of digits. This option is useful if your index contains titles like “1984”, “2001, a *Space Odyssey*” that should be arranged numerically.

**NOTE** ⚠ *At the time of releasing this manual, IndexMatic<sup>3</sup> does not deal with special ordering rules that may be required in certain sequences, e.g. “Louis XIV”, “Louis XV”, “Louis XVI”. A solution is being worked on that should be implemented in a future update.*



## 9. Processing Hits Reports

In addition to its distinct export modes (all performed by the **Index...** button), IndexMatic<sup>3</sup> comes with a **Hits** button that allows you to generate useful reports without even leaving the program.

There is a key difference between the Quick Matches feature ⚡ (which you use to preview matches) and a **Hits** report. The latter not only submits your queries to the search engine, it also computes the resulting terms or entries with respect to formatters,



# Output and Reports



Page Rank, Smart Case and output conditions, that is, as they would actually appear in the finished index.

This leads us to draw a clear distinction between a MATCH and a HIT: a match is an expression actually found in the document(s), a hit is the corresponding term that either derives from it or results from additional transformations and conditions.

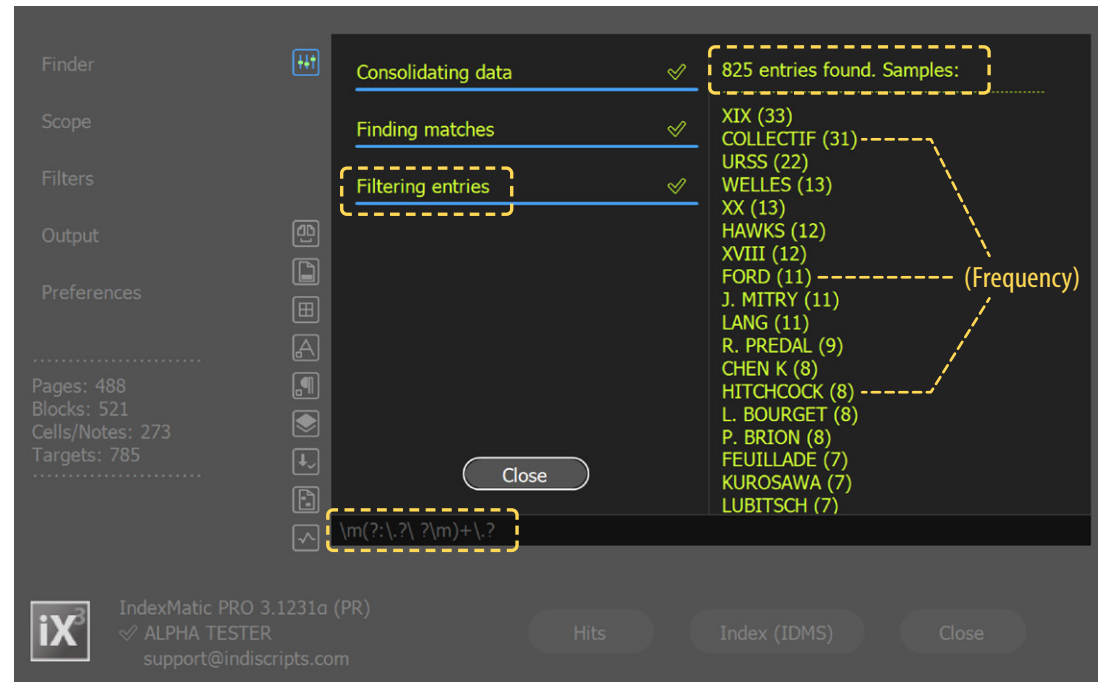
**NOTE** Remember that many events may happen: grouping of multiple search keys under a single topic; applying of a formatter; merging of case variants; filtering by page rank, frequency and/or shared pages; addition of parent topics. . .

Hits represent *countable* elements. IndexMatic<sup>3</sup> knows how many times a particular term has been generated or encountered (that's its *frequency*) and which page locators are to be associated with it (this determines its *width*). → [Filtering Index Data by Statistical Criteria](#).

The **Hits** feature takes all this into account and presents you with data that reflects what your settings do actually produce. You can also use this tool to study various aspects of the text before building the index.

For example, let's click the **Hits** button with the following single query (assuming page rank = 1).

```
/\m(?:\.\.? \ ?\m)+\.\.?/Is
```

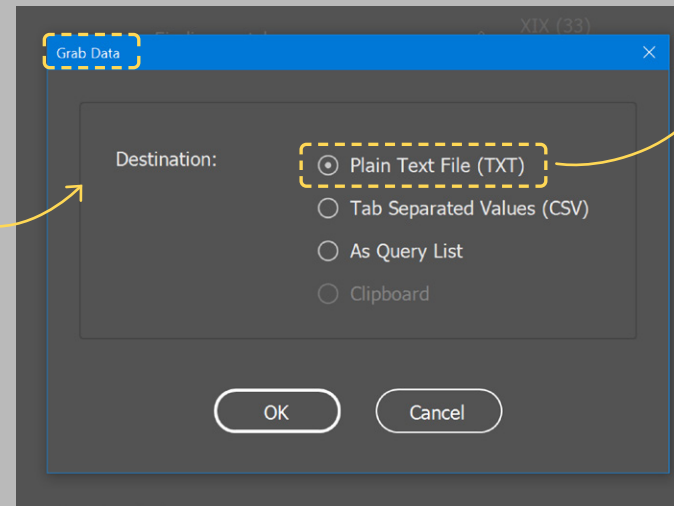
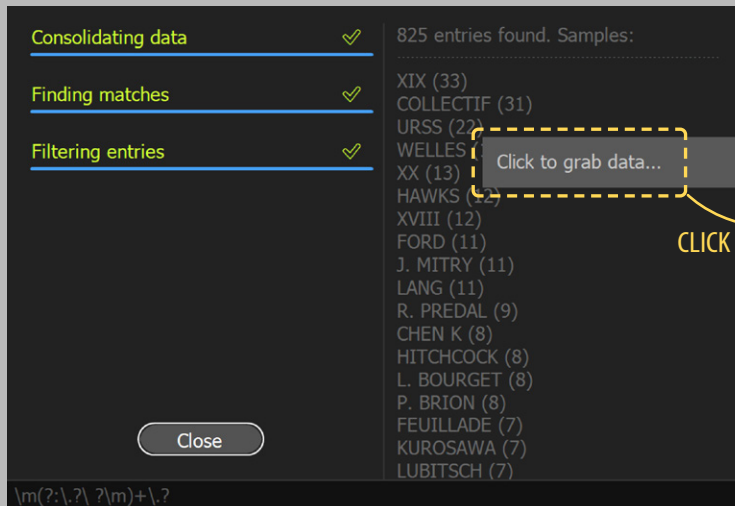


This query is intended to detect acronyms or similar forms. The console shows up and reports on three tasks: “Consolidating data”, “Finding matches” and “Filtering entries”. The latter is specific to the **Hits** feature.

As shown in the above screenshot, our sample document (and our current settings) can produce 825 entries in accordance with our query. A few of them are shown in the console. The number in parentheses indicates the frequency of the term under consideration, e.g. “HITCHCOCK (8)”. Such information wouldn't be available from a Quick Matches ⚡ scan.

What IndexMatic<sup>3</sup> reports as an “entry” is either a simple term (excluding parent topics) or the full topic structure if required. This depends on the **Hits** preferences. → [Customizing Hits and Stats](#)

# Output and Reports



Term	Freq.
XIX	33
COLLECTIF	31
URSS	22
WELLES	13
XX	13
HAWKS	12
XVIII	12
FORD	11
J. MITRY	11
LANG	11
R. PREDAL	9
CHEN K	8
HITCHCOCK	8
L. BOURGET	8
P. BRION	8
FEULLADE	7
KUROSAWA	7
LUBITSCH	7
MANN	7
MINNELLI	7
MURNAU	7
OPHULS	7
RAY	7
RENOIR	7
VIDOR	7
XVII	7
A. BAZIN	6
CUKOR	6
FBI	6
HUSTON	6
DEMENTEE	6

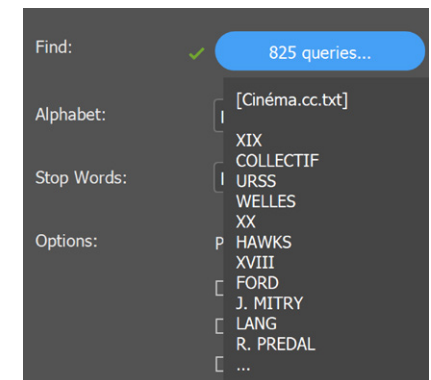
## ► Grab Hits Data

Of course you can get more than an overview. With the Hits console visible, hover the mouse over the sample area and click. The **Grab Data** dialog shows up. Select a **Destination** format among the following choices:

- *Plain Text File (TXT)*. Renders the hits in a .TXT file and opens your default text editor.
- *Tab Separated Values (CSV)*. Renders the hits in a .csv file (actually based on the tab separator) and loads it in your default spreadsheet editor (if available).

	A	B
1	Term	Frequency
2	XIX	33
3	COLLECTIF	31
4	URSS	22
5	WELLES	13
6	XX	13
7	HAWKS	12
8	XVIII	12
9	FORD	11
10	J. MITRY	11
11	LANG	11
12	R. PREDAL	9

- *As Query List*. Translates all collected topics into a new query list and makes it available right now in IndexMatic<sup>3</sup>. This option allows you to very quickly create a draft list based on the filtered terms. A typical use is to extract high page-rank expressions. You then have at hand a ready-to-use—or at least ready-to-refine—QL.



- *As Stop Words*. This new option—introduced in v. 3.23052—allows you to instantly convert the collected elements into a **Custom Stop Word list**.

from version 3.23052

**NOTE** ⚠ The "Clipboard" destination is not implemented yet.

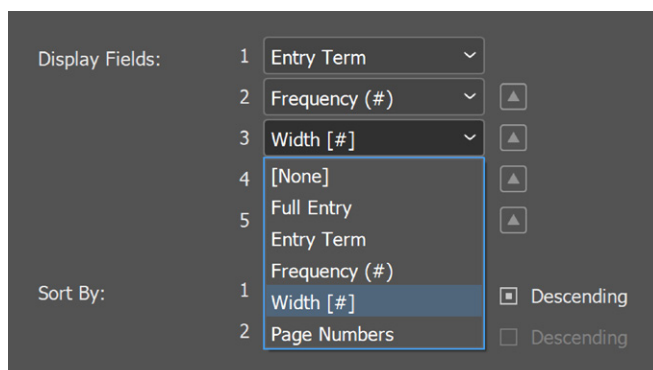
# Advanced Preferences and Features



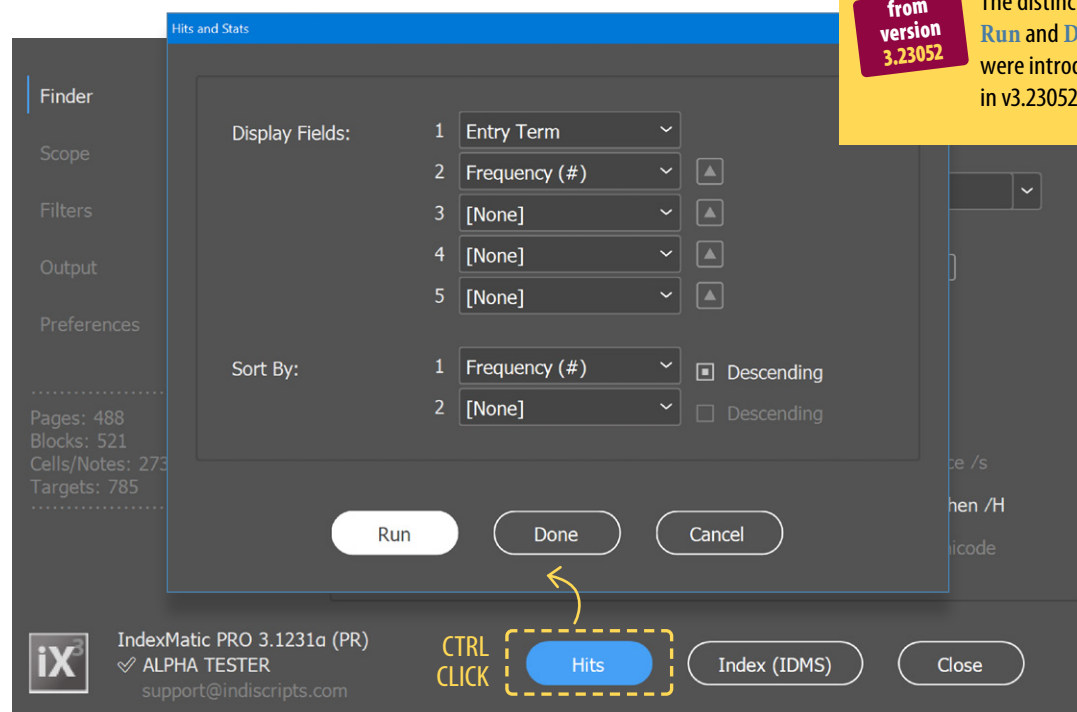
## 1. Customizing Hits and Stats

IndexMatic<sup>3</sup> lets you control which fields are to be shown in a **Hits** report. You can make the program not only displays terms and frequencies, but also the *width* (number of found pages), the sequence of page numbers, or the full structure of an index entry (that is, including parent topics).


- 1) From whatever active panel, **Ctrl-Click** the **Hits** button. The **Hits and Stats** dialog shows up.
- 2) In the **Display Fields** area, select and order the fields that need to be shown. Each line, numbered from 1 to 5, contains a drop-down list with six options:



- *[None]* Skip that line.
- *Full Entry* The entire topic structure, e.g. “Philosophy ► Authors ► Plato”.
- *Entry Term* The term only (i. e. the target topic).

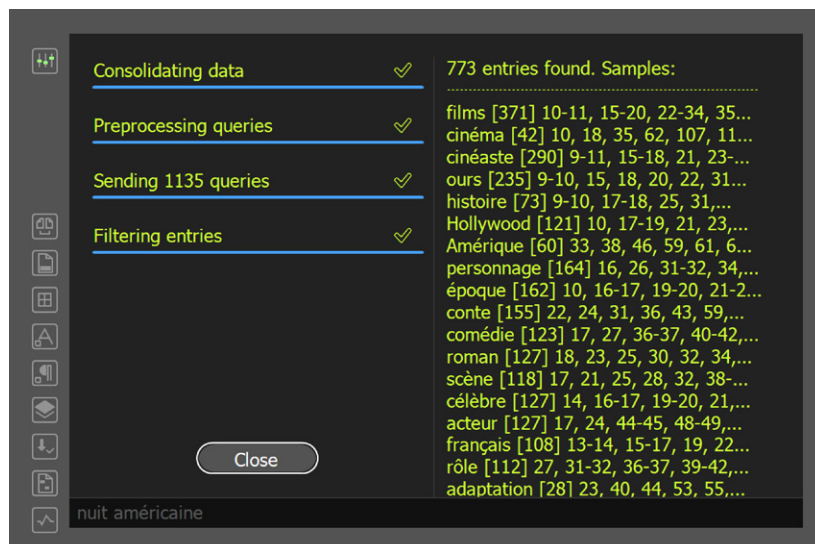


- *Frequency (#)* Term frequency (by convention, this number is formatted in parentheses).
- *Width [#]* Term width (by convention, this number is formatted in square brackets).
- *Page Numbers* Sequence of locators associated to the term.

**TIP** The  buttons allow you to push an item up, so you can quickly reorder the display fields according to your needs.

- 3) In the **Sort By** area, select a primary key (first line) and optionally a secondary one (second line). Outgoing data will then be sorted by these fields. If selected, the *Frequency* and/or *Width* fields can

# Advanced Preferences and Features



be ordered from highest to lowest value by clicking the *Descending* checkbox.

**TIP** You can sort data by a hidden field (that is, not selected in the display fields). For example, you might choose to report terms and page numbers only, and those items ordered by descending frequency.

4) Click **Run** to apply your preferences and generate a report. (Or, just click **Done** to save your preferences — *available in v. 3.23052.*)

Your **Hits and Stats** preferences are retained until you change them.

Preview of a custom Hits report in the console.

Display fields:

1. *Entry term*,
2. *Width [#]*,
3. *Page numbers*.

Order By: *Frequency* (descending).

## 2. About IndexMatic<sup>3</sup> Preferences and Settings

IndexMatic<sup>3</sup> will remember not only the settings you have chosen for the active scope (if you ask it), but also your long-term preferences.

**NOTE** Backup processes are mostly imperceptible and adapt to your session. When working on several indexes in parallel, the program ensures that settings get restored (queries, filters, layout, output folder, etc.) with respect to each particular project.

Global preferences address options considered independent of the scope. They determine the default behavior of the program. Here are the main ones:

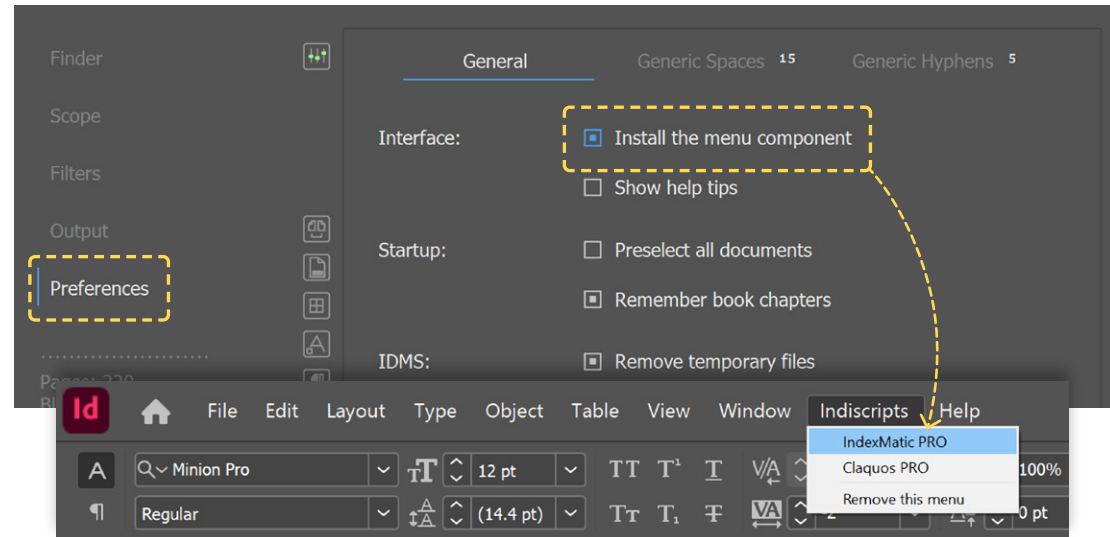
- **Hits and Stats** settings (display and sort fields) as detailed in → [Customizing Hits and Stats](#).
- **Favorite queries**.
- Your preferred *Grab* format (TXT, CSV, Query List) when you click the data area in the console.
- Whether to show the parent folder when an output text file is generated—rather than opening the file.
- In IDMS export mode, whether to place the snippet automatically if possible.

# Advanced Preferences and Features



- Whether to install or remove the Indiscripts menu.
- Whether to show help tips in IndexMatic<sup>3</sup> user interface.
- Whether to remove temporary IDMS files when successfully placed.
- Length of the context used in Matches-in-Context feature, and whether stop words should be ignored here (*see below*).

**NOTE** The last settings listed above are controlled from the **Preferences** ► **General** subpanel discussed below.



## 3. Changing Basic Preferences

Scattered global options are grouped in the **Preferences** ► **General** subpanel.

### ► *Install the menu component*

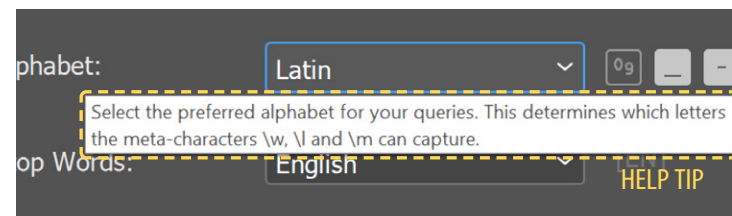
The checkbox **Interface** ► *Install the menu component* controls the visibility of the item “IndexMatic PRO” (or “TRY”) in the Indiscripts menu, which is automatically loaded when InDesign starts up.

We recommend enabling this option if you frequently use IndexMatic<sup>3</sup>: it will save you from opening the Scripts panel whenever you have to run the script.

### ► *Show help tips*

In all IndexMatic<sup>3</sup> dialogs and panels, every hovered element triggers the display of a tooltip that summarizes its function. This behavior is controlled by the option **Interface** ► *Show help tips* (enabled by default).

Now that you are an expert user of the program, you may no longer need to see this information. If so, uncheck the option!



**from version 3.23041** The option **Startup** ► *Preselect all documents* (added in v.3.23041) changes the default behavior of the **Target Documents** dialog when multiple documents are available as IndexMatic<sup>3</sup> is starting up.

**from version 3.23081** The option **Startup** ► *Remember book chapters* (added in v.3.23081) makes the **Target Documents** dialog remember your previous selection in a book. (Due to a bug, a fix has also been applied to this feature in v.3.24012.)



# Advanced Preferences and Features




## ► IDMS: Remove temporary files →

IndexMatic<sup>3</sup> needs to create an .IDMS file when exporting your index as either an InDesign document, or snippet. In any case where the element is immediately placed, there is probably no need to keep the file in memory.

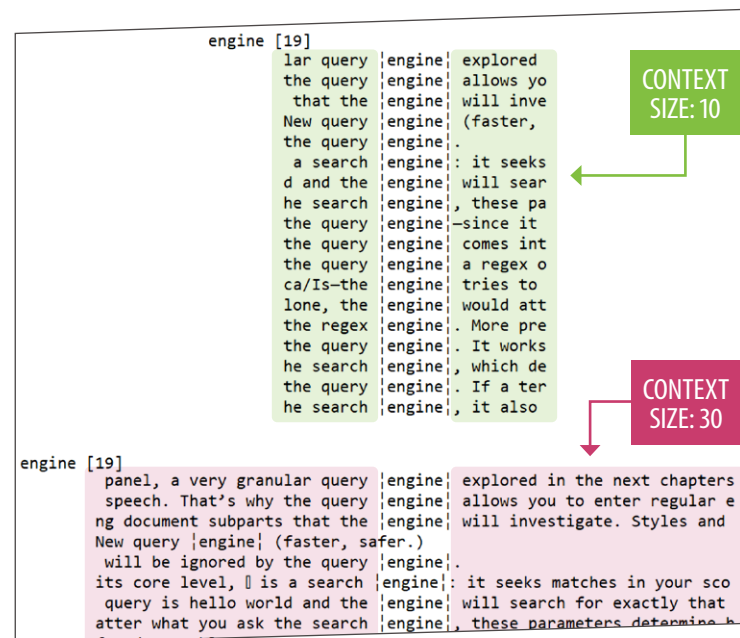
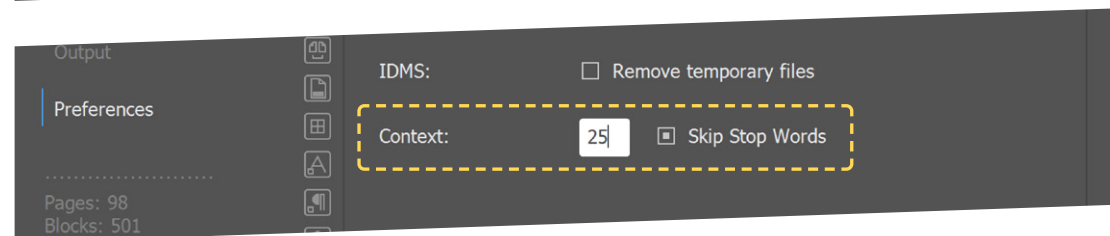
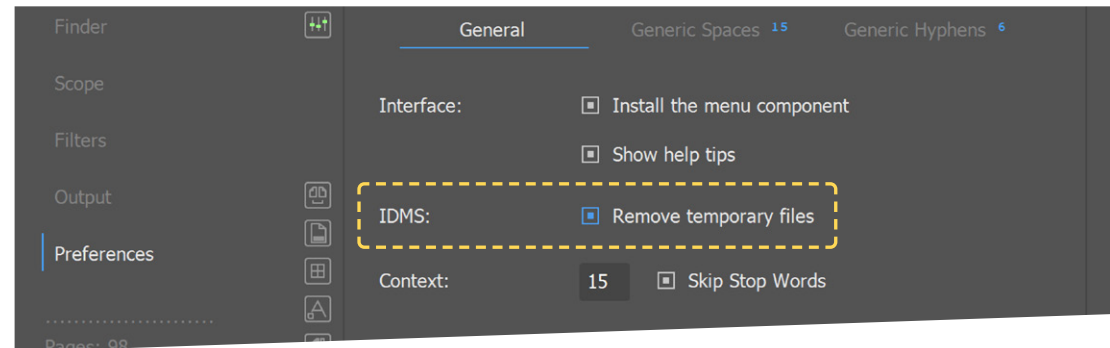
Click IDMS ► *Remove temporary files* to have the program clean up these temporary files as it goes.

## ► Context Preferences →

The **Context** area controls the behavior of the Matches-in-Context feature . The number entered in the edit box (between 1 and 32) determines how many characters can be extracted to the left and right of a context node. The default value is 15.

Click *Skip Stop Words* to prevent STOP WORDS from taking the place of a context node. This is of course the recommended option, unless you want to examine the context of noise words. This option has no effect if **Finder** ► **Stop Words** is set to *[None]*.

**NOTE** Keep in mind that the Matches-in-Context feature operates on raw matches, not hits. Hence it does not reflect how matches are to be transformed or filtered out. The purpose of *Skip Stop Words* is then to lighten the concordance report. (On the distinction between matches and hits, see → [Processing Hits Reports](#))





# Advanced Preferences and Features



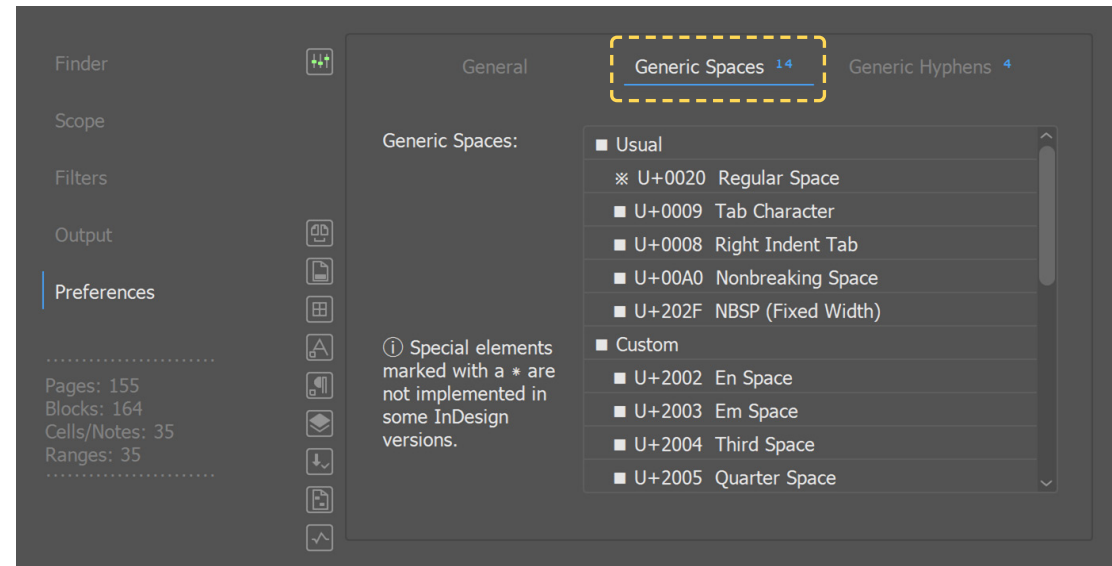
## 4. Fine-Tuning Generic Spaces

The **Generic Spaces** subpanel offers a checklist to precisely define the working range of the `GENERIC SPACE` wildcard. The regular space `U+0020` is necessarily included, and by default many *white spaces* are specified too.

- 1) Activate **Preferences** ► **Generic Spaces**.
- 2) Scroll down the list and check all the characters to be recognized by the space wildcard when generic space is turned on (`/s`). This applies to both regex and token queries. Three categories are available: *Usual*, *Custom*, and *Special*.

- *Usual* and *Custom* contain InDesign characters ordinarily regarded as spacing between words (including tabs, Figure Space, Hair Space...).
- The *Special* branch lists more exotic characters such as Forced Line Break, Flush Space, Discretionary Line Break, Zero Width Non-joiner, Zero Width Joiner. Depending on the documents in your project, some of these characters might be added to, or removed from, your custom set.

Unlike general preferences your selection of generic spaces is saved with the current scope, so you can maintain a unique configuration for each distinct project.

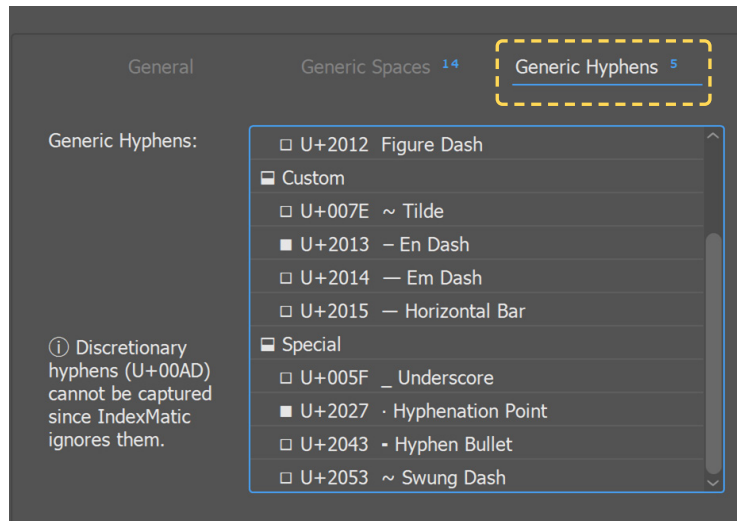


## 5. Fine-Tuning Generic Hyphens

Similarly, the **Generic Hyphens** subpanel allows you to define the working range of the `GENERIC HYPHEN`. Necessarily included is the hyphen-minus `U+002D`, to which you will usually add the Unicode hyphen `U+2010` and the nonbreaking hyphen `U+2011`.

- 1) Activate **Preferences** ► **Generic Hyphens**.
- 2) Scroll down the list and check all the characters to be recognized by the hyphen wildcard when generic hyphen is turned on (`/h`). This applies to both regex and token queries. Here again, three categories are available (*Usual*, *Custom*, and *Special*).

# Advanced Preferences and Features




- *Usual* and *Custom* contain hyphen-like characters such as Figure Dash, Tilde, and En Dash. Only select those that you want to capture.
- The *Special* branch includes characters that are not actual hyphens but might still be treated so under special circumstances: Underscore, Hyphenation Point, Hyphen Bullet, and Swung Dash.

**NOTE** ⚠ The **Generic Hyphens** preference is unrelated to—and has no effect on—the option **Finder ▶ Alphabet ▶ -**. If enabled, the latter strictly considers regular hyphens.

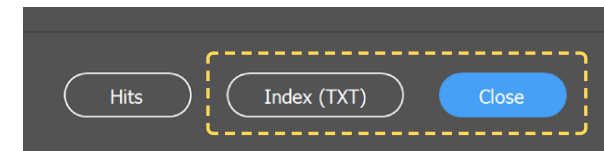
Unlike general preferences, your custom set of generic hyphens is saved with the current scope, so you can maintain a unique configuration for each distinct project.

## 6. Saving Active Settings

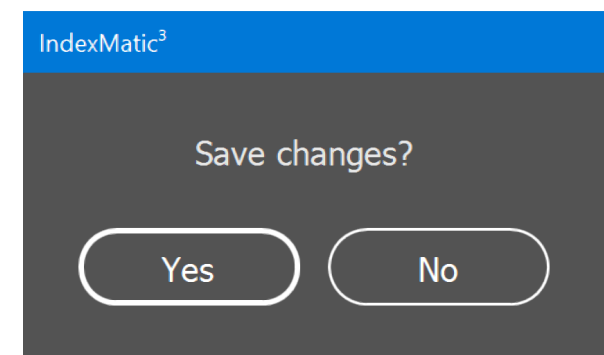
When working on a given document or book, the overwhelming majority of options and parameters you specify apply to this particular scope and should be saved separately. This includes settings defined in the **Finder**, **Scope**, **Filters**, and **Output** panels, the managed markers and special characters controlled from the  button, as well as the preferred **Generic Spaces** and **Generic Hyphens** lists.

There are two normal ways of leaving the main dialog and closing the program:

- Generating the index by clicking the **Index...** button.
- Clicking the **Close** button.




In both cases IndexMatic<sup>3</sup> determines if any settings have changed during your session. If so, a final question box is shown and lets you decide whether the new options must be saved. The default answer is **Yes**. Click **No** if you just made temporary tests or undesired changes: previous settings will then be restored the next time you run the script.





# Advanced Preferences and Features



1) From whatever active panel, click the **Settings Manager** button  at the top of the main dialog.

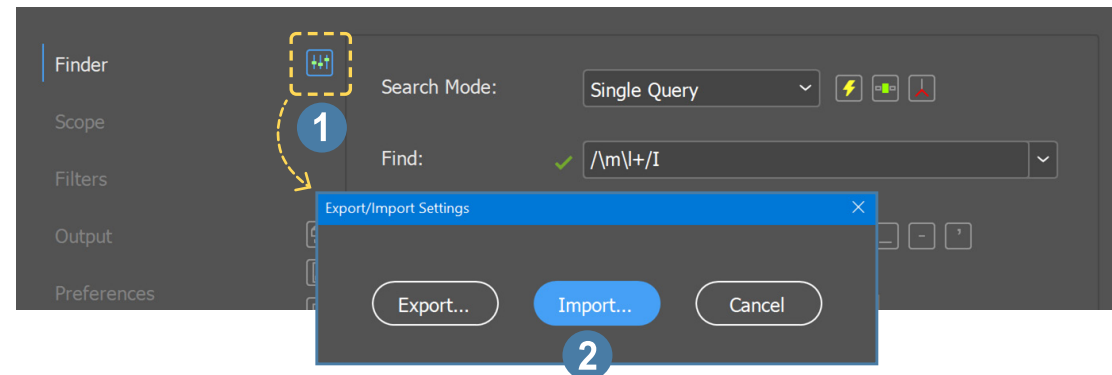
2) In the **Export/Import Settings** box, click **Import...** This allows you to browse your filesystem and choose the JSON file you want to import.

**NOTE** ⚠ Of course IndexMatic<sup>3</sup> cannot parse an arbitrary JSON file!

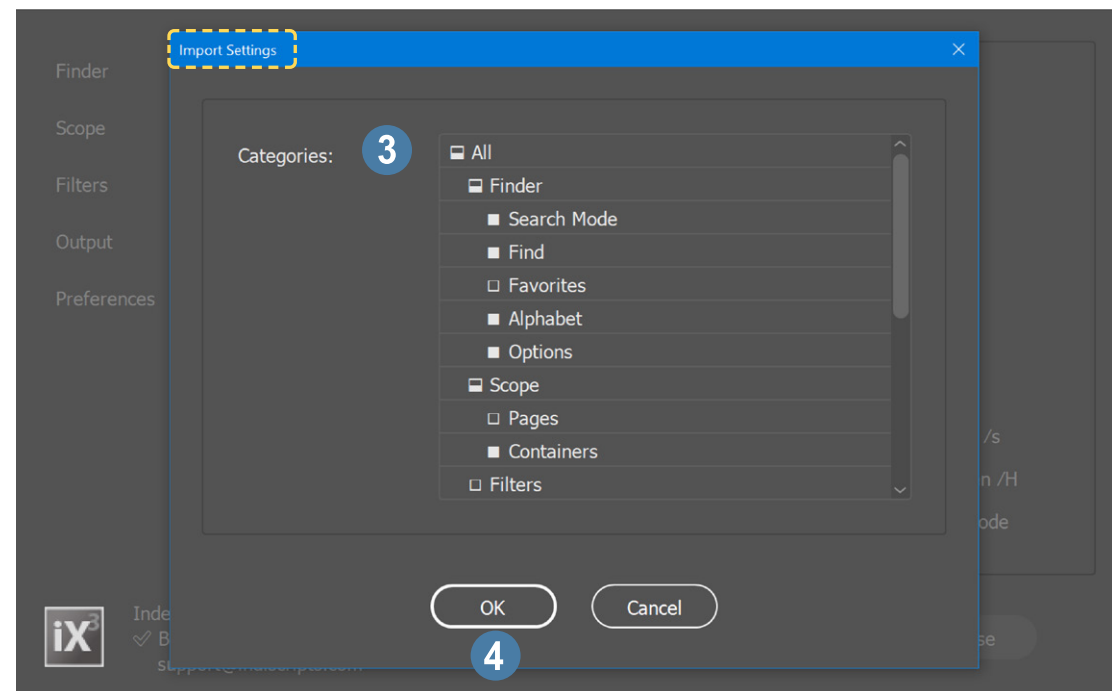
3) In the **Import Settings** dialog, select the particular settings, or group of settings, that you want to recover. The **Categories** checklist allows you to selectively include/exclude items or branches.

- *Finder* reflects the **Finder** panel and contains nodes like *Search Mode*, *Find*, *Alphabet...* *Favorites* is unchecked by default—so your current favorite queries won't be overridden.
- The *Scope* and *Filters* branches list all settings controlled from the corresponding panels.
- The *Output* branch provides three nodes: *Destination*, *Layout*, *Sorting*, addressing the options of the corresponding subpanels.
- Finally, the *Preferences* branch allows you to restore global preferences, generic spaces, and/or generic hyphens.

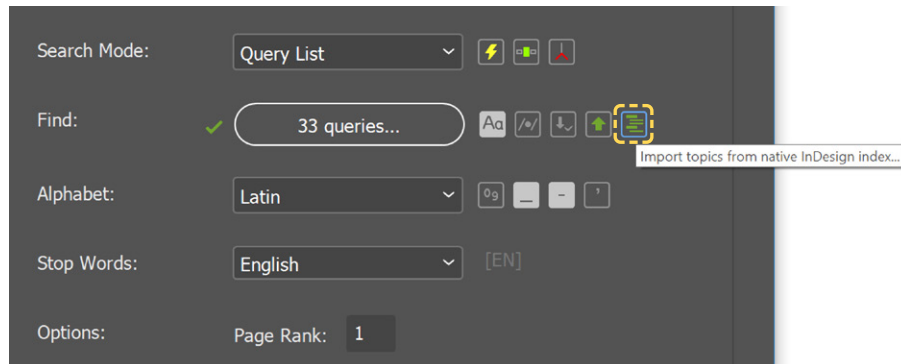
4) Click **OK**.




↓  
Navigate and  
choose JSON file...



# Advanced Preferences and Features



## 8. Importing a Native InDesign Index

If the active scope has a document that already contains a native InDesign index, an extra button  appears to the right of the Find field in *Query List* mode. It allows you to automatically import topics from the original index and make them the items of a new query list.

**NOTE** *At the time of releasing this manual this feature is still experimental. More precise documentation will be provided later.*

## Tables and Figures

TABLE: \$-variables 51  
 TABLE: Alphabets and Unicode blocks 30  
 TABLE: Container codes 19  
 FIGURE: Context size 99  
 FIGURE: Directive processing 69  
 TABLE: Greedy vs. non-greedy quantifiers 56  
 TABLE: GREP-like metacharacters 64  
 TABLE: IDMS style names 75  
 TABLE: Indicators 8  
 FIGURE: Key-Topic general syntax 48  
 TABLE: Local flags 46  
 FIGURE: Main selectors (punctuation) 84  
 TABLE: Markers and special characters 37  
 TABLE: Metacharacters 59  
 TABLE: Migration from IndexMatic<sup>2</sup> — Main changes 11  
 FIGURE: Mixing paragraph and character style filters 23  
 FIGURE: Off-page content 25  
 TABLE: Panels and subpanels 8  
 TABLE: Regular expression quantifiers 55  
 FIGURE: Scanning endnotes 18  
 FIGURE: Scanning footnotes 17  
 FIGURE: Scanning tables 18  
 FIGURE: Style targets 22  
 TABLE: Ternary operator shortcuts 53  
 TABLE: Topic structure cheatsheet 53  
 TABLE: Unicode properties 60  
 FIGURE: Width vs. frequency of a term 81  
 FIGURE: XML DTD 77

## #

\$-variable 51-54, 57-58, 63, 66-72, 80

## A

accent → *diacritics*  
 acronym 41, 91, 94  
 Active Settings 101  
 advanced (options) 1, 11, 28, 33, 39, 43-44, 61, 78, 91-93  
 algorithm 79, 91-92  
 all-uppercase 91  
 alphabet 11, 30-33, 39, 59-61, 80, 92  
 Alphabetic Headings 5, 11, 44, 77, 80, 84-86, 91  
 alphabetizing → *sort*  
 alphanumeric 31, 92  
 alternation (|) 54, 57  
 ambiguity 13-15  
 Anatolian 62  
 anchored object 11, 16-19, 37, 64  
 apostrophe 30-31, 39, 65  
 Apply Indentation 80  
 Arabic 11, 30, 60-62  
 Armenian 30-31, 61  
 ASCII 30-31, 34, 64, 92  
 assertion 59, 63-65  
 asterisk 37, 84

## automatic

- index 4
- search mode 4-5, 23, 27, 30-33, 38-39
- word extraction 38

## B

backslash (\) 54  
 base letter 77, 80  
 Bengali 30  
 bibliography 16, 67, 76  
 block 7, 11, 16, 30, 48  
 → *Unicode (block)*  
 book 6-7, 14, 76-78, 82, 98  
 Bosnian 80  
 brackets 15, 28, 35, 43, 58-59, 84, 96  
 branch → *tree*  
 break 20-22, 26, 37, 62, 72  
 → *line break*  
 Break (action) 37  
 Breton 31, 80  
 bullet 64, 84-86, 101

## C

capitals 44  
 capturing 28, 31-34, 39, 46, 51-52, 56-58, 61, 65, 70  
 • group 51, 57-58, 62, 65

## case → *formatter*

cased letters 60  
 lowercase/uppercase 19, 33, 44, 52-53, 59-62, 70, 92  
 magic • 44  
 • sensitivity 29, 33, 44-47  
 Smart • 91-94  
 title • 52-53, 60  
 • variants 33, 39, 91-94  
 category 6, 20, 100, 103  
 cell 7-8, 11, 15-20, 25 → *table*  
 chained 19-20  
 character 26-38, 52-55, 58-65, 69-70, 83-86, 100-101 → *style*  
 • class 44, 55, 58-65  
 • range 6, 58  
 unassigned • 60  
 Character Style Filter 8  
 cheatsheet 53  
 checkbox 9-10, 74-76, 89-90, 97-98  
 checklist 10, 13, 21, 24, 100, 103  
 checkmark 11, 43  
 Chicago (elision) 89  
 child (topic) 49  
 Chinese 62  
 class → *character*  
 clipboard 43, 95  
 Close (button) 28, 101  
 cluster 20-22 → *text*  
 code (container) 19  
 codepoint → *Unicode*  
 collation 91-92 → *sort*  
 colon (:) 56, 83-84



comma (,) 15, 70, 84-85  
 command 9, 21, 39, 45-47, 53, 64  
 comment 65-66 → *query*  
 Common Search Settings 29  
 concatenation 52-53, 57-58  
 concordance 35-36, 99  
 → *Matches in Context*  
 connector 60  
 console 12, 19-21, 28-29, 35, 94-97  
 Live Preview 12  
 container 11, 15-16, 19  
 → *text (container)*  
 context 11-12, 35-36, 98-99  
 continued 20, 37  
 Coptic 30, 61  
 Count (button) 43-44  
 counting 6-7, 21, 51-52, 64, 81-83,  
 94  
 Create Hyperlinks 78-79  
 cross-reference 45, 49, 65-69,  
 74-77, 83-86, 90  
 Ctrl (keyboard) 10, 32, 42, 96  
 currency 60  
 custom (stop words) 32  
 Cyrillic 11, 30, 61, 92

## D

Danish 80  
 dash 34, 60, 64, 84-87, 101  
 → *hyphen*  
 data → *Grab data; text data*  
 decimal 60

default  
 • behavior 32, 37, 47, 75, 82-84,  
 88-92, 95, 98  
 • (button) 38, 84, 92  
 delimiter → *separator*  
 depth 11-13, 50, 77, 92  
 descending (order) 97  
 destination 8, 73, 78, 95 → *output*  
 Devanagari 30, 60  
 diacritics 31, 65, 80, 92  
 dialog 9, 13-14, 24, 98, 102-103  
 digit 30, 52, 58-59, 62, 80, 87-89, 93  
 → *hexadecimal; number*  
 directive 11, 66-72, 91  
 ~format 71-72  
 ~include 69, 72  
 ~split 70-71  
 discretionary (character) 34, 37,  
 64, 84, 100  
 display (field) 96  
 distance 88  
 document 4-7, 13-14, 18, 25, 28, 35,  
 78-79, 98  
 double-click 3  
 drop-down 10, 30, 37, 40-41, 73,  
 83, 89-92, 96  
 DTD 77 → *XML*  
 DUCET 5, 92  
 Dutch 91

## E

ECMAScript 62-64  
 edit box 10-11, 15, 39-41, 83, 88, 99  
 editor → *text (editor)*

elision 11, 31, 87-89 → *page (range)*  
 ellipsis 64  
 em, en spaces → *space*  
 encoding 41, 47 → *UTF16, UTF8*  
 ending assertion 63  
 endnote 7-8, 11, 15-19, 37, 89-90  
 English 1, 4, 31, 39, 55, 87, 90  
 entry 5, 31-32, 44, 48, 51, 77, 80-83,  
 86-87, 91-96  
 ePUB 78  
 error 11, 43  
 escape (sequence) 52-54, 59  
 Ethiopic 30  
 European Ordering Rules (EOR)  
 5, 92  
 explicit (topic) 50  
 exporting 11, 77, 85, 102  
 Export/Import Settings 102-103  
 ExtendScript 64-65, 71  
 extracting 1, 6-7, 37-38, 69-71, 99  
 • nodes 36

## F

favorite (query) 11, 40-41, 97,  
 102-103  
 field 11, 40, 74-76, 82, 96-97  
 figure space → *space*  
 file 2, 41-43, 72-76, 95, 99, 102-103  
 • name 102  
 filter 6-8, 11, 21-26, 31, 39, 63, 81,  
 94  
 filtering index data 81, 94  
 Filters panel 6-7, 11  
 • intersection 23

Final Touch 79-80  
 Finder 4, 7-8, 26, 31-32, 42, 91,  
 101-103  
 flag 7, 29, 32-33, 39, 46-48, 51,  
 54-56, 65, 71  
 folder 2-5, 72-78, 97, 102  
 font 23, 62  
 footnote 1, 7-8, 11, 15-20, 37, 89-90  
 Footnotes/Endnotes Only 8  
 forbidden (characters) 59-62  
 forced line break 37, 84-86, 100  
 → *line break*  
 format → *directive*  
 Format (subpanel) 5, 10, 60,  
 71-79  
 formatting 6, 69-74, 77-78,  
 85-88, 96  
 topic formatter 52  
 frame → *text (frame)*  
 French 1, 31, 39, 55, 85  
 frequency 1, 6-8, 11, 17, 59, 65,  
 81-82, 94-98  
 full-state filters 24

## G

Generic Hyphen 8, 11, 34, 39, 46,  
 59, 65, 100-103  
 Generic Space 8, 11, 34, 39, 46-47,  
 56, 59, 65, 100-103  
 Georgian 30  
 geresh 31  
 German 1, 88, 91  
 glyph 37, 62, 80  
 Grab data 4, 12, 23, 34-36, 95-97

greater-than (>) 45, 48  
 greedy/non-greedy (quantifier)  
 12, 56  
 Greek 11, 30, 61, 92  
 GREP 24, 58-59, 62-65  
 GREP Features 64  
 group → *capturing* •  
 Gujarati 30

## H

hair space → *space*  
 Hangul 62  
 hapax 82  
 heading 45, 49-50, 67, 74-75, 80, 91  
 → *Alphabetic Headings*  
 Hebrew 11, 30-31, 62, 80  
 hexadecimal 61-62  
 hidden  
 block 11, 16  
 frame 16-17  
 layer 17, 24  
 marker 7  
 text 17, 25  
 hierarchical 47-49  
 hit 12, 28, 39, 93-99  
 Hits and Stats 94-97  
 Hits (button) 12, 93-96  
 processing • report 28, 93, 99  
 homonym 91  
 hyperlink 11, 76-79

hyphen 31, 34, 39, 59, 84-85,  
100-101 → *Generic Hyphen*  
hyphen-minus 31, 34, 58, 85, 100  
soft • 34, 37, 60  
Unicode • 34, 84-85, 100

## I

icon-button 9, 74-78  
ideograph 62  
IDML 11, 74  
IDMS → *snippet*  
implicit (topic) 45-47, 54, 77  
import 9-11, 79, 102-104  
  Import/Export Settings 102-103  
include → *directive*  
INDB → *book*  
INDD 73, 78-79  
indent 37, 48-51, 64, 73-75, 80,  
84-86, 90  
  Indent Character 84-85  
InDesign 1-3, 11-14, 24-25, 37, 61,  
64-65, 73-79, 104  
Index... (button) 93, 101  
  IDMS 76  
  INDD 79  
  TXT 5, 74  
  XML 77  
IndexFrame 75  
indexing 1-3, 6, 11-14, 64-65,  
75-76, 98  
  automatic • 4  
  book • 14  
  index entry 31, 83, 87, 91  
indicator 7-9, 15, 21, 24, 29, 67

Indiscripts menu 3, 98  
input 11, 45, 48, 69-76, 83-84, 91  
insignificant → *stop word*  
installing 1-4, 98  
integer 10, 32, 55 → *number*  
interactive 78  
interface 42, 98  
internal DTD 77  
interpreter → *query*  
intersection vs. union 23  
interval 87 → *page (range)*  
Italian 1

## J

JavaScript 64, 70, 102  
JSON 11, 102-103

## K

Kannada 30  
Keep (action) 37  
keyword 16, 35, 81  
Khmer 30

## L

language 1, 5, 31-32, 80, 91-92  
Latin 4, 11, 30, 61, 80, 92  
layer 6-8, 11-13, 16-17, 24-25  
  Layer (filter) 8  
layout 1, 6-11, 16-17, 73-75, 83-90,  
97, 103  
  Layout (panel) 1, 6-11, 16-17,  
73-75, 83-90, 97, 103

letter 30-33, 60-61, 80, 92  
  → *base letter*  
  letter-by-letter (sort) 92  
level 50  
lexical units 7, 15-20, 23, 31, 91  
ligature 80  
line break 37, 64, 84, 100 → *break*  
list 5, 8-10, 40-44, 68, 71, 91, 95,  
100 → *checklist; drop-down;*  
  *query list; word (list)*  
literal 27-30, 44, 47, 51  
Live Preview 12  
Load (button) 42  
local flag 33, 44-48 → *flag*  
localization 1, 90  
locator 75-78, 86 → *page (number)*  
locked 13, 37  
lookahead/lookbehind 63-65  
lowercase → *case; formatter*

## M

macOS 1-2, 41  
Magic Case Sensitivity 44 → *case*  
Magic Regex Term 44 → *regex*  
main dialog 4, 12-14, 42, 101-103  
Malayalam 30  
marker 6-7, 11, 20, 26, 35-38, 64,  
75, 88-90, 101  
  • & special character 26, 37-38  
markup 23, 54, 74, 85-86

## N

match 11-12, 26-36, 45, 50-51,  
54-59, 63, 69, 93-94, 99  
  → *quick match*  
  Matches in Context 12, 35-36,  
98-99  
math (symbol) 60  
maximum 8, 27, 38-39, 81-82  
measure → *statistics*  
menu 3-8, 56, 98  
merging 11, 22, 33, 91, 94  
  Merge field 91  
  Merge Spaces 11, 22  
message 10-13, 29, 36, 43  
metacharacter 30-31, 59-61, 64-65  
minimum 1, 8, 13, 27, 38-39, 82, 88  
modifier (Unicode) 60  
mouse 12-14, 25, 36, 95  
multiple  
  • books 14  
  • conditions 19  
  • locations 20  
  multi-document 25, 79  
  multi-level 11, 50, 72, 84 → *level*  
  multilingual 5, 61, 92  
  multi-state object 16  
  • pages 15, 20  
  • paragraphs 20, 26  
Myanmar 30  
  
{N} 90  
native index 104  
negated class 59-60  
  → *character (class)*

negative lookahead 63-65  
nested 16, 19, 24, 37, 49, 57, 64, 67  
  • topic 49, 67  
newline 41  
New Query List 42  
next-page marker 59, 88  
node 36, 99, 103  
noise word → *stop word*  
non-breaking  
  • hyphen 31, 34, 64, 84-85, 100  
  • space 62-64, 84  
non-digit 59  
non-joiner → *ZWNJ*  
non-printable 84  
Norwegian 91  
note 19, 90 → *endnote; footnote*  
  Note Markers 89  
  • number 11, 19, 27, 37, 78, 90  
noun 44, 55  
number 5-7, 14, 21, 35, 60, 81-82,  
87-90, 96 → *page (number)*  
  numbering 87, 90  
  numeral 41

## O

occurrence 6, 26, 54-55 → *match*  
off-page 17, 25  
operator 45, 49, 52-57, 65-67  
  → *ternary operator*  
ordering rules 5, 92-93 → *sort*  
Original List Order 8  
Oriya 30

output 8, 45, 68-78, 85

- Destination 10, 73-82
- file 73
- folder 5, 74-78, 97
- Layout 9, 88-90
- panel 5-8, 73, 101
- pattern 69-71
- Sorting 31-33, 44, 80, 91

override 23, 32, 46-48, 74

overset text 17, 25

Oxford (elision) 89

## P

{P} 90

page → *off-page; shared pages*

- number & locator 6, 13, 28, 35, 78, 81, 85-90, 96-97
- Page Rank 4-5, 8, 32, 38, 46, 81, 94
- range 8, 11, 15, 19-20, 85-88

paragraph 23, 26, 37, 63, 75, 86

→ *style*

parent

- folder 72-74, 97
- page 14, 25
- topic 47-51, 67, 77, 80, 86, 94-96

parenthesis 19, 28, 51, 55-58, 94-96

→ *capturing (group)*

parsing 11, 26, 37, 43, 54, 62, 65, 69, 72, 102-103

path 5, 19, 48, 72-78 → *text (path)*

pattern 6-7, 45-47, 53-58, 61-71, 90

phonebook 91

pilcrow 64, 86

pipe (|) → *alternation*

Place automatically 76

placeholder (InDesign) 76, 79

placeholder 51, 69-71

placing → *snippet (InDesign)*

plain text → *text*

plural 55, 72

positive lookahead 63

POSIX 65

preferences 3, 8, 34-36, 75, 97-100, 103

- & settings 97

prefix 31, 35, 51, 57, 62, 66-68, 83

preprocessing 12, 69

Preselect all documents 98

preview 11-12, 28, 35-36, 93, 97

primary

- content 17-18
- frames 16
- key 96
- text 16

Private Use Area 60

progress 12, 28

proper name 6-7, 41

punctuation 60, 92

## Q

QL → *query (list)*

quantifier 55-58, 61-62, 65

query 7, 11, 26-29, 32-35, 39-51, 54, 65-69, 72, 91

- advanced • 11, 33, 39, 43-44
- comment 65-69
- favorite • 11, 40-41, 97, 102-103
- interpreter 45, 51, 62, 69
- list (QL) 8, 11, 28, 41-44, 50, 65-68, 72, 80, 91, 95, 104

New Query List 42

- options 29, 32, 35
- regex • 28-30, 38-39, 44-46, 50-51, 54, 65, 68-69, 91
- token • 28, 44, 51, 65, 100

quick match 1, 11, 28-31, 93-94

Quick Matches button 28-29

quotes 31, 64, 68-71

## R

radio (button) 15

range → *character; page*

reference → *cross-reference; note (number)*

regex (regular expression) 28, 39, 44-47, 50-51, 54-56, 61, 64-65, 71

RegExp (JavaScript) 62-64, 70

Remember book chapters 98

Remove (action) 37, 41, 99

Rename (button) 41

report 1, 28, 93-94, 97-99

restore 24, 40

right-click 2, 41

Russian 1

## S

sample 12, 28, 36

- area 12, 36, 95

saving 2, 40-43, 87, 97, 102

Save to JSON 102

scanning 17-18

scheme 45, 48, 60-63

scope 7-8, 11-18, 25, 28, 101

Scope panel 7, 13-15, 21

script 3-4, 14, 20, 98, 101

Scripts Panel (InDesign) 2-3, 98

search

- engine 6-7, 11, 15, 23, 26, 29, 35-39, 44, 47, 61, 81, 88, 93
- mode 4, 8, 27-29, 35, 38-43, 103
- options 29, 39, 43, 48

“See”, “See also” 66-67, 75, 84-86, 90 → *cross-reference*

See Ref 90

selector 9-11, 19, 83-88

semantic 23, 82

semicolon (;) 15, 70, 84

separator 66, 70, 75, 83-87, 90

- & delimiters 66, 83

sequence 52, 56, 61, 65, 69, 85-88, 96

Settings manager (button) 9, 102-103

shared pages (width) 8, 11, 78, 81-82, 94

shortcut 5, 53-55

Show Output Folder 74

sibling (topic) 50

single query 11, 27-28, 39-40, 54-55, 66, 94 → *query*

singular 55-57, 72

Sinhala 30

slash (/) 27-29, 39, 43, 46-47, 54, 65-66, 69-71, 84

Smart Case 91-94 → *case*

SmartSort 92

snippet (InDesign) 73-79, 84-85, 97-99

soft hyphen → *hyphen*

sort 5-8, 32, 44, 73, 80, 91-92

- index entries 91
- Sort By 96
- Sorting panel 5, 32, 91

space 22, 34, 37, 56, 59, 64, 83-85, 100 → *Generic Space*

- em • 64, 83-85
- en • 64, 83-85
- figure • 64, 84, 100
- flush • 64, 84, 100
- hair • 64, 84, 100
- horizontal • 59
- punctuation • 64
- quarter • 64, 84
- sixth • 64, 84
- Space (action) 34, 37, 64, 83, 100
- Space After 83-85, 90
- Space Before 83-85
- thin • 64, 84
- third • 64, 84
- unstyled • 22
- white • 22, 34, 52, 56, 100

spacing 60, 74, 84, 100

Spanish 1, 80

special character 20, 26, 29-30, 35-38, 46, 54, 59-61, 84, 101

- & marker 26, 37-38

spelling 6, 50, 82

split 22, 25, 69-71 → *directive*

spreadsheet 95

star 40-41

starting assertion 63

statistics 1, 6, 32, 81, 94-97

stepper 10

Stop (button) 12, 28

stop word 4, 31-32, 38-39, 81, 95, 99

story 16-18, 22-23, 57, 75

strand 20

stream 17, 20, 26, 33-37

style 7-8, 21-24, 63, 75

- character • 7-8, 11, 21-25, 39, 63, 74-75
- filter 8
- Make • independent 23
- paragraph • 8, 11, 23-24, 74-75, 84

sublevel 6, 75, 91

submatch 51-53, 57-58 → *match*

subpanel 5, 8, 73, 91, 98-100

substring 20, 25, 39, 51, 56, 70-71

subtopic 6, 45, 50, 73, 77

suffix 11, 31, 44, 48, 83, 88

surrogate pair 60-62

switch (button) 3, 8-9, 30, 43-44, 65, 91

swung dash 101

syllables 62

symbol 25, 53, 60, 64-65, 83

syntax 11, 43, 58, 62-64, 68-70

system requirements 1

## T

tab 34, 59, 64, 73, 83-85, 95, 100

table 7-8, 11, 16-20, 37

- Tables Only 8, 18-19

Tamil 30

target 7, 13, 19-22, 48-49, 98

- document 4, 7, 13-14, 46, 79, 98
- topic 47-51, 66, 81-83, 96

taxonomy 49

Telugu 30

template (checkbox) 89-90

term 47-50, 77, 81-84, 91, 94-97

ternary operator 52-53

text

- container 7-8, 11-21, 24-25, 28, 75
- data 1, 6-7, 15-20, 37
- editor 5, 11, 36, 41-43, 62, 73-74, 95
- frame 11, 15-20, 25, 37, 74-75, 79
- path 15, 19-20
- plain • 12, 28, 36, 41, 73-74, 85, 95
- variable 35-38, 64

TextEdit 41

Thai 30

thin space → *space*

thread 16, 20

Tibetan 30

tilde (~) 59, 64, 69, 84, 101

- *directive*

time-consuming 12, 28, 36, 79

title case → *case; formatter*

token (query) 28, 44-48, 51, 65, 100

tolerance 88

topic 45-53, 66-67, 75-77, 80, 94

- formatters 51-52
- structure 53

tree 10, 24, 67, 84, 100-103

trigram 31

trim 52-53 → *formatter*

tryout version 1

typography 65, 85-87

## U

Ukrainian 80

unassigned 60

underscore ( \_ ) 30, 59, 101

Unicode 30, 58-61, 92

- *Private Use Area; surrogate pair*
- block 30
- codepoint 38, 58-62, 65, 84, 92
- properties 59-60, 65

unstyled spaces 22

update 3, 76, 93, 102

uppercase → *case; formatter*

user interface 7

UTF

- UTF16 5, 40, 58, 61, 92
- UTF8 41, 73

## V

validity 11-13, 43

variable

- InDesign → *text v. query* → *-\$variable*

version(s)

- 3.23041 65, 70, 98
- 3.23042 39, 46-47
- 3.23052 32, 95-97
- 3.23081 98
- 3.23111 10-12
- 3.24011 23
- 3.24012 24, 64, 98

tryout • 1

## W

warning 4, 11-15, 21-26, 29, 32-35, 38, 42-44, 59-67, 71-72, 76, 79-80, 85, 89-95, 101-103

white space → *space*

whole word → *word*

width 37, 81-82, 96, 100

- *shared pages*

wildcard 34, 100

Windows 1-2, 41-42

word → *stop word*

- automatic • extraction 38
- list 1, 7, 23, 32, 43-44, 68, 72, 95
- whole • 9, 29-30, 33, 39, 46-47, 64
- word-by-word (sort) 92

## X

XML 11, 73-74, 77, 85

## Z

ZIP 1-3, 102

zoom setting (hyperlink) 79

ZWNJ (Zero Width Non-Joiner) 37, 64, 84, 100



# IndexMatic 3.x

www.indiscripts.com

A plug-in for Adobe® InDesign® based on Adobe® ExtendScript and ScriptUI. Created, designed and developed by Marc Autret. User Interface available in English, French, German, Spanish, Italian, and Russian.

*My very very special thanks to Peter Kahrel, Roland Dreger, Laurent Tournier, Janus Bahs Jacquet, Ariel Walden, Valentine Lontsikh, Camilo Umaña, Sascha Fronczek & Marien van Westen. I also wish to thank the people who have helped promote this product, in no particular order: Franck Payen & Jean loup Fusz (InDesign User Group Paris), Stéphane Baril (Adobe France), David Blatner, Mike Rankin & Jeff Potter (CreativePro), Jean-Claude Tremblay, Rainer Klute, Jean-Christophe Courte—*not forgetting the graphic designers, trainers, authors, users, and gurus who have all contributed at some level to the success of IndexMatic!

Main Product Page: <https://indiscripts.com/category/projects/IndexMatic>  
Tryout version: <https://indiscripts.com/blog/public/scripts/IndexMatic3Try.zip>  
Technical Support: [support@indiscripts.com](mailto:support@indiscripts.com)

Purchasing IndexMatic<sup>3</sup>: <https://indiscripts.com/store/IXME>  
End User License Agreement: <https://indiscripts.com/pages/eula>  
Terms and Conditions of Sale: <https://indiscripts.com/pages/cgv>  
Copyright Notice: <https://indiscripts.com/pages/copyright>

This manual, as well as the software documented in it, is released under license and may be used or copied only in accordance with the terms of that license. The content of this document is subject to change without notice. Every effort has been made to ensure that the information in this document is accurate. However, Indiscripts assumes no responsibility or liability for any error that may appear in this document. InDesign, the InDesign logos, are trademarks of Adobe Systems Incorporated.

© Indiscripts, 2006-2024.  
All rights reserved. Made in France.

