Nuix Workstation

SEARCH GUIDE

Version 9.10.0

September 2022

nuix

Nuix Search Guide

Purpose of the guide

The Nuix Search Guide is designed to help you use various search techniques and syntax to obtain accurate results with Nuix products.

The search syntax detailed in this guide is used for all Nuix applications that perform a search.

Nuix products provide a powerful platform for conducting effective searches. While some Nuix products provide a user interface to build your query, others allow you to build freeform queries using the search syntax. The Nuix search syntax enables you to define queries to extract relevant information from the data that exists in your system. The syntax is easy-to-use and illustrated in this guide with plenty of examples.

Before you learn about the search syntax, it is recommended that you familiarize yourself with the search interface of the Nuix application you are using.

Search tips and tricks

Before we go through queries and operators, here are some search tips and tricks to help you formulate accurate queries.

- Sample your search results to ensure that your syntax is correct for the results you were expecting to be returned.
- Some Nuix products allow global settings that can restrict searches from retrieving certain property fields such as path. Check these settings prior to starting any Nuix search, if your Nuix install is shared with other users. Refer to the corresponding user guide for more information.
- Nuix does not support the use of the reserved SQL word Null. Instead, use a reverse wildcard query to find all null values in a particular field. To find all items without a name use *–name:**
- Searches are executed on the current results set shown and thus can produce different results when the same query is run again. For example, if you search for *name:** and then the reverse -*name:**, the second search will not produce any results. Using the **Clear** button before running the search -name:* will result in a list of items with no names.
- The search term should always be in lower case because words from the content are stored in lower case except in Exact queries.
- Punctuation marks are ignored in searches except in Exact queries.
- If your search contains a distance parameter, ensure that it is in the format “*nn*”. If the 0 is omitted, the “.” is treated as punctuation and ignored.
- Use a hyphen to reverse a search query. For example, to match all system files you would use *kind: system*. To exclude all system files, you can use *–kind:system*.
- It is highly recommended that you explicitly use operators and parentheses for precedence ordering when constructing complex queries. This will assist in the reading and understanding of how the search will be executed, and ensure that there are no unintended or unexpected results.
- Some paths contain virtual names for folders without names in the physical container items such as PSTs and Top of Personal Folders. It is highly recommended that any searches using paths with a 0 result set are double-checked by breaking up the path query into two sections, *path:[path up to the container] * AND path:*[Final folder name that you wish to search]*

We’ll look at queries and operators in detail in this guide.

Query types

The Nuix search syntax is comprised of a wide range of queries from simple single-word queries to complex field queries. The following query types are included in the Nuix search syntax:

Query type	Description
Simple	Search for a single-word to locate all occurrences of the word (search term).
Wildcard	Locate one or more words with the help of wildcard characters.
Fuzzy	Search for terms based on the distance between two strings.
Exact	Search for terms with punctuation and/or exact capitalization.
Field	Search for single-word terms occurring within specific properties of data items.
Phrase	Search for a sequence of words in a particular order.
Regular Expression	Search for terms using pattern matching.
Range	Search for terms within an upper and lower bound.

Each type of query is explained in detail with examples to help you with the syntax.

Simple queries

The simple search query is a single-word query. Enter a single word or term to locate all occurrences of the word within the case index. The word will be found in the properties, the name, the path, and the text content of items, or a combination of any of these items.

Example

Query String	Results
`Joe`	Matches all items with the word "joe" somewhere in the properties, name, path or the text content.

Note:

For Simple Queries, search terms are not case sensitive and will always be searched in lowercase; for example, the queries "joe", "Joe" or "JOE" will return identical results.

By default, a Nuix search starts from the pathname of an item. For example, if you are searching for the term “Joe" in files located in the folder *Evidence**Email**Joe Email**Important*, the query will find all items within the folder *Joe Email* and within any subfolders as well. To exclude the pathname from your search, specify the specific fields within which you want to search for “Joe”. For example: (field:joe), content (content:joe) and/or properties (properties:joe). Alternatively, include a clause to not search with the path, such as (joe AND –path:joe).

Wildcard queries

Purpose of wildcard queries

Use wildcards to substitute characters within a term to allow searching for one or more words that share some of the same characters.

There are two types of wildcard characters that can be used for substitution:

- A single character which is represented by the “?” symbol
- Multiple characters which are represented by the “*” symbol

Single character wildcards

To perform a single character wildcard search, use the "?" symbol within the word at the character that you wish to be variable within the returned results. The "?" will always match exactly one character within the term, except for an empty space, with the remainder of the term required to be exactly the same each time.

The “?” character can also be used in any position within the term; however, when used at the front of a term the search can take longer as the search is looking for any character first before matching the rest of the term.

Examples

Query String	Results
`jo?`	Matches "joe" and "joy", but does not match “jo” or “joke”
`j??e`	Matches "joke" and "jade"
`?oe`	Matches "joe" and "toe"

Multiple character wildcards

To perform a multiple character wildcard search, use the "*" symbol. The query will match zero or more characters at the position of the “*” symbol within the returned results. The "*" symbol can be used in any position; however, when used at the front of a term the search can take longer as the search is looking for any character first before matching the rest of the term. Multiple character wildcards can also be used to return all items within a field that have some value.

Examples

Query String	Results
`nu*`	Matches "nunawading", "nuix", "nu" and "numpages"
`*work`	Matches "work", "network" and "patchwork"
`content:*`	Matches all documents which have text content

Mixing wildcards

Use both the single and multiple character wildcards in a single query to form a mixed wildcard query to create a targeted search.

Example

Query String	Results
`n?u*`	Matches strings that: • start with the character "n", • followed by exactly one more character, • followed by "u", • followed by any number of characters. Therefore, it will match "nsummary", "neutral" and "noun”.

Fuzzy queries

Fuzzy queries allow you to search for terms based on the distance between two strings.

Nuix supports fuzzy searches based on the Levenshtein distance or *Edit distance* algorithm. The Levenshtein distance between two strings (words) is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other.

To perform a fuzzy search, add the '~' (tilde) symbol at the end of the search term. This will find words that are similar to one another in as far as the characters they contain. The match is approximate rather than exact, hence the term "fuzzy search".

You can add an optional parameter after the tilde to specify the maximum edit distance (Levenshtein distance). The value can be 0, 1, or 2, where lower values require a more similar match (using 0 is the same as not using a fuzzy search) and higher values allow more letters to be different. The default value in the absence of this parameter is 2.

- Ensure characters are written in lowercase as uppercase characters will not be matched. Even if a lowercase character is the same letter as the uppercase character in the query, it will count as a difference between the two terms.
- Runs of characters that look like a single term but which are actually broken apart (particularly common in Chinese and Japanese) will not be matched.
- Stemming will not be applied to the term in the query.

Examples

Query String	Results
hot~ (aka hot~2)	Matches the words "hot", "huw", "tlot", "sat", "mat", "riot", ...
cold~	Matches the words "cold", "clod", "mlod", "buld", "coil", "mould", ...
hot~1	Matches the words "hot", "shot", "lot", "hut", "hoc", "how", "got", "pot", ...
hot~0	Matches the words "hot"

Phrase queries

Phrase queries allow you to search for a sequence of words in a specific order.

To construct a phrase query, add double quotes (") at the start and the end of the sequence of words that are to be considered phrase. Only items that contain the words in exactly the same sequence as the phrase will be returned.

Punctuation marks such as “.” and “:” are treated as whitespace when searching. If a punctuation mark is in between two words, it is converted to whitespace and the entire term either side of the punctuation mark is automatically converted to a phrase.

Examples

Query String	Results
`"Joe Bloggs"`	Matches items that contain "Joe Bloggs", in that order.
`"P&L"`	Matches items that contain "P L", in that order.
`joe.bloggs@nuix.com`	Matches items that contain "joe bloggs nuix com", in that order. Note that when your search query contains punctuation marks without any space around them, they are ignored. The remaining words are treated as a phrase query. To include punctuation in your search, use an exact query.
`"joe.bloggs@nuix.com"`	Matches items that contain "joe bloggs nuix com", in that order. This is the same result as without quotes.

`"AAA|BBB"`
 	
`"AAA BBB"`
	Matches nested metadata properties that contain AAA as the parent folder and BBB as the child folder. When you search for the "AAA|BBB" metadata name, the search queries "AAA|BBB" and "AAA BBB"- both return the correct item. This is currently supported for Hancom data.

Note: To include punctuation in your search, use an Exact query.

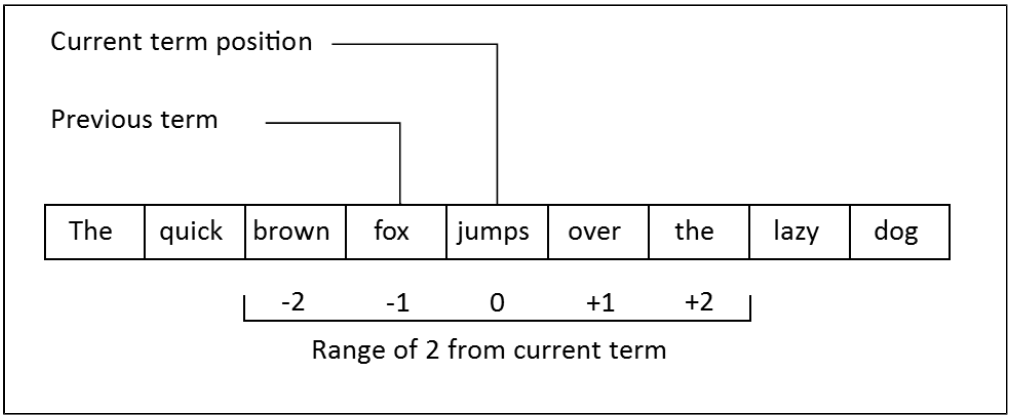
Phrase queries with slop

A "slop" of a phrase query is to search for words within a certain distance of each other, using the tilde (~) symbol at the end of a phrase query along with a numerical value that indicates the number of unrelated words that can occur in between.

The behavior of phrase queries with slop applied is not immediately obvious and has a very different search behavior to proximity searching which it is commonly mistaken to be. Slop values are measured from the word immediately following the first word found.

To understand this concept better, let us take the phrase, "The quick brown fox jumps over the lazy dog." If we want to search for "fox quick"~2. Nuix will first find "fox", and then set about looking for "brown" immediately after fox, allowing it to fall 2 words either side of the starting term which is the term after the initial term in the query.

Visually this can be represented as follows:



The numbers below the words indicate the slop value required to match each term, from -2 up to 2. Therefore, the following queries should (and do) result in a match (this assumes that stop words are not in use):

- "fox jumps"~2
- "fox over"~2
- "fox the"~2
- "fox brown"~2

The following queries do not result in a match:

- "fox quick"~2
- "fox lazy"~2

Additionally, "fox fox"~2 does not return a match as phrase queries can only match each term once for each position in the phrase.

Examples

Query String	Results
`"Joe Bloggs"~2`	Matches items that contain "Joe Bloggs", as well as items that contain "Joe John Bloggs".
`"Joe* Blog*"~2`	Matches items containing "Joe Bloggs", Joe's Blog" or other combinations that match the provided wildcards, with up to two unrelated words in between them.

Note: If quote marks are left off a slop query unintentionally, the last word within the intended phrase will be treated as a fuzzy search within the query using the “~” and any numerical value as parameters.

Exact queries

Exact queries allow you to search using a specific case or punctuation, which are ignored in simple queries. Exact queries can only be used across data that has the **Enable Exact Queries** option selected when indexed.

To search for a sequence of characters that include upper case characters or punctuation marks, add single quotes (') at the start and the end of the search term.

Whilst the syntax is similar to other queries, exact queries can provide very different results. This is because exact queries search sequences of characters, whereas simple queries and phrase queries search the indexed words only. For example, simple queries will not match a search term unless it matches the whole word found in a document. Exact queries with the same search term will result in a match if it matches only part of a word found in a document.

Another example is the *Proximity Operators* that will use the number of characters between search terms for exact queries to calculate distance, whereas simple queries will use the number of words to calculate distance.

Examples

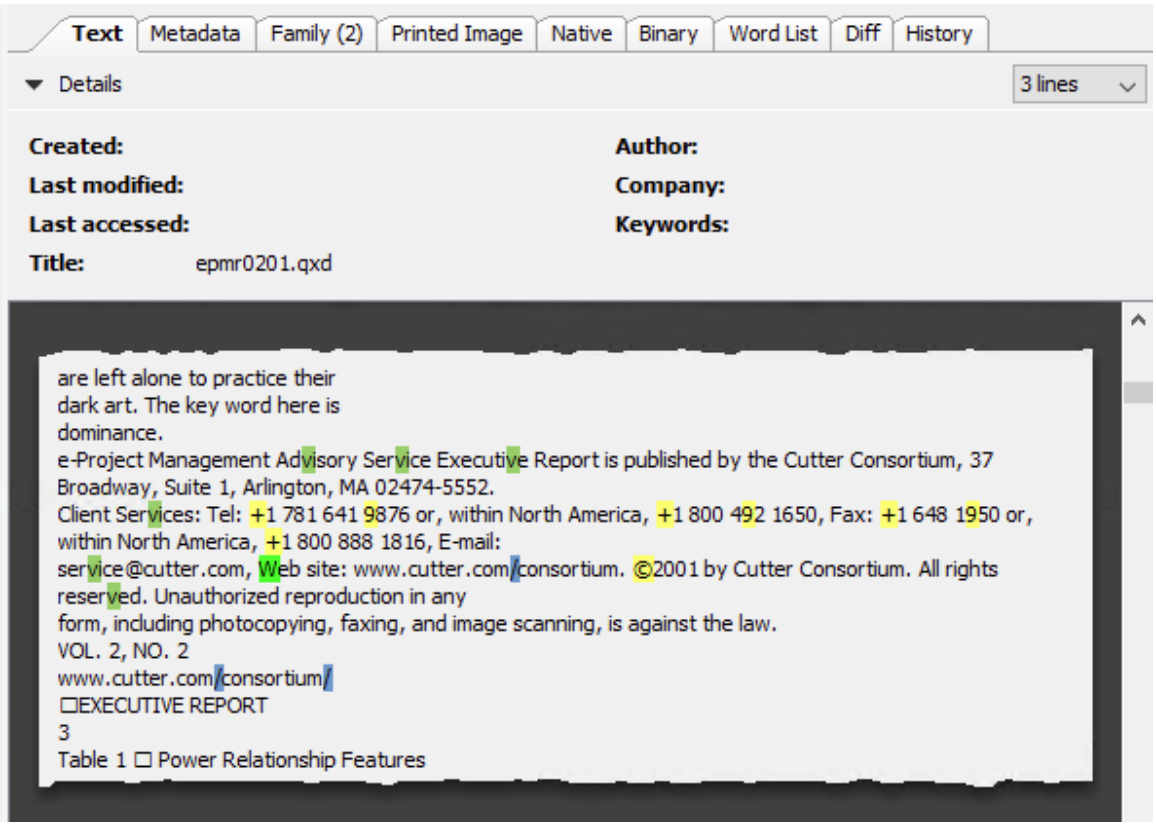
Query String	Results
`'$1,000'`	Matches only items containing the text, "$1,000"
`'123-456-0000'~2`	Matches items containing the text, "123-456-0000", with up to two unrelated characters mixed in

The exact search query result is highlighted. For example where an exact query matching syntax *content:'ding'* highlights the "ding" in the term "fin*ding*" within the item text view.

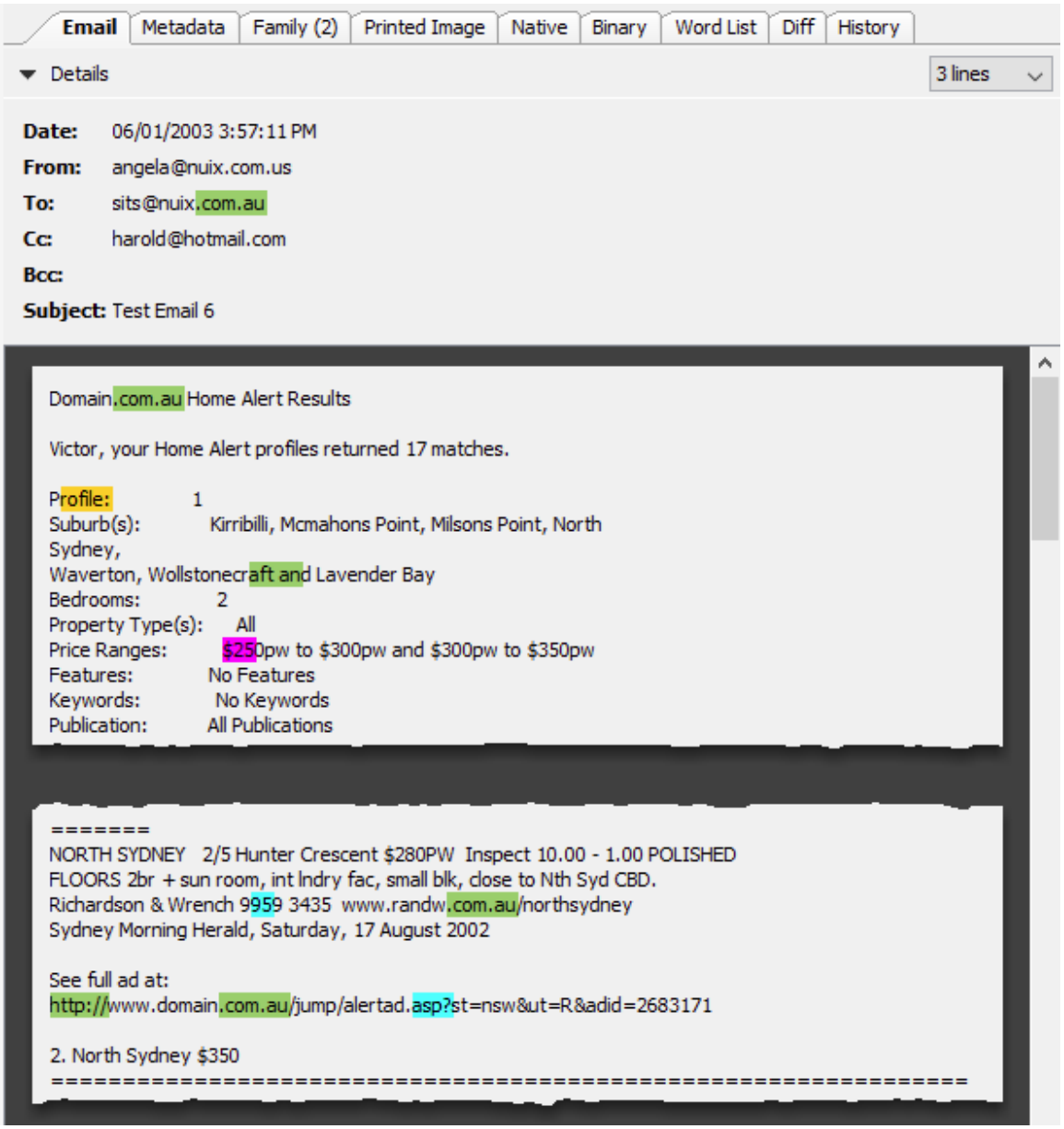
Note: When using exact queries, both sides of a proximity search must be an exact query term otherwise an error will occur. When a proximity search is used within an exact query, the distance is calculated from the end of the first term to the beginning of the second term.

The following screens show exact query searches for a single character, multiple characters, Unicode and slop searches.

Single character query

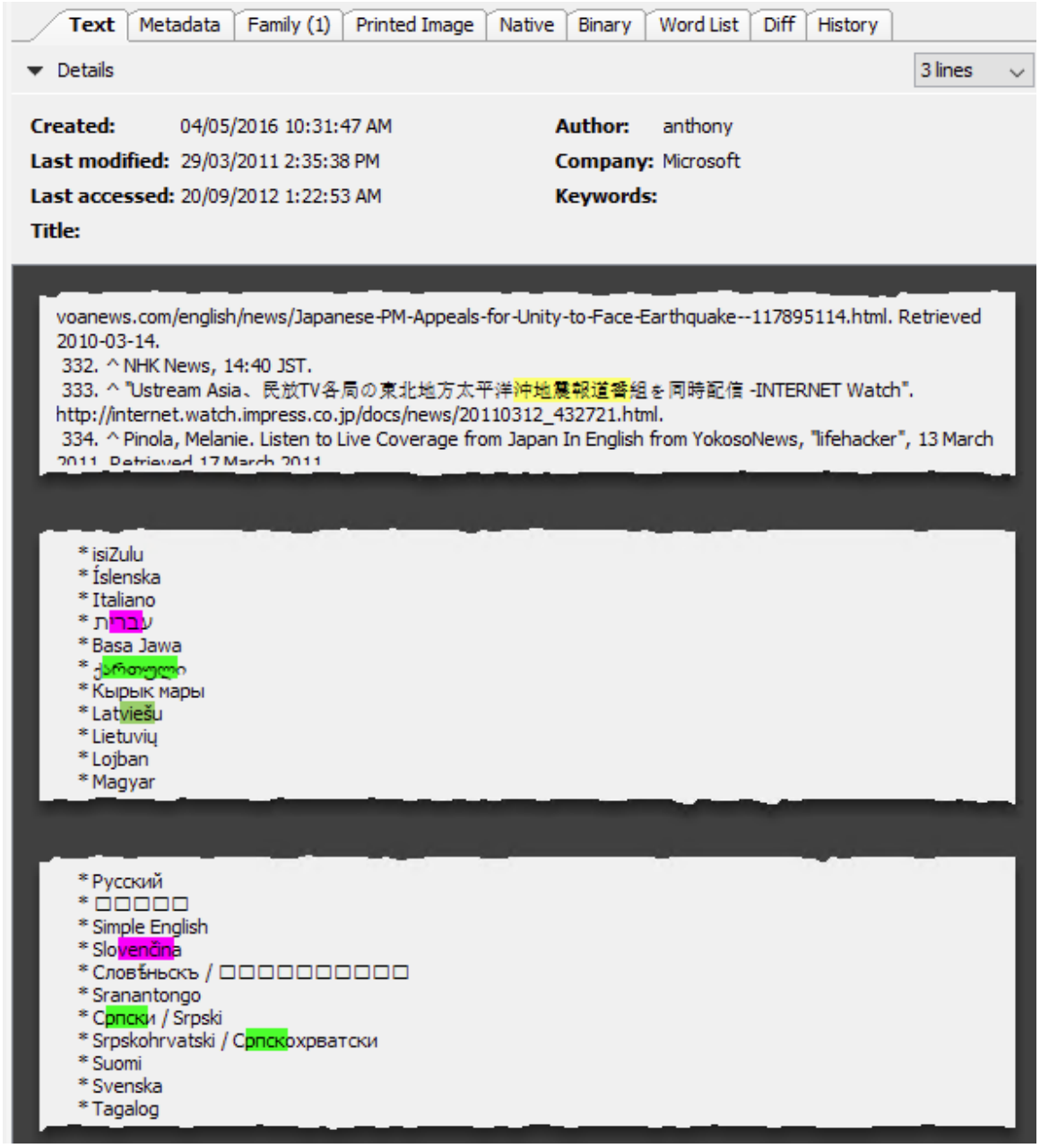


Multiple character query



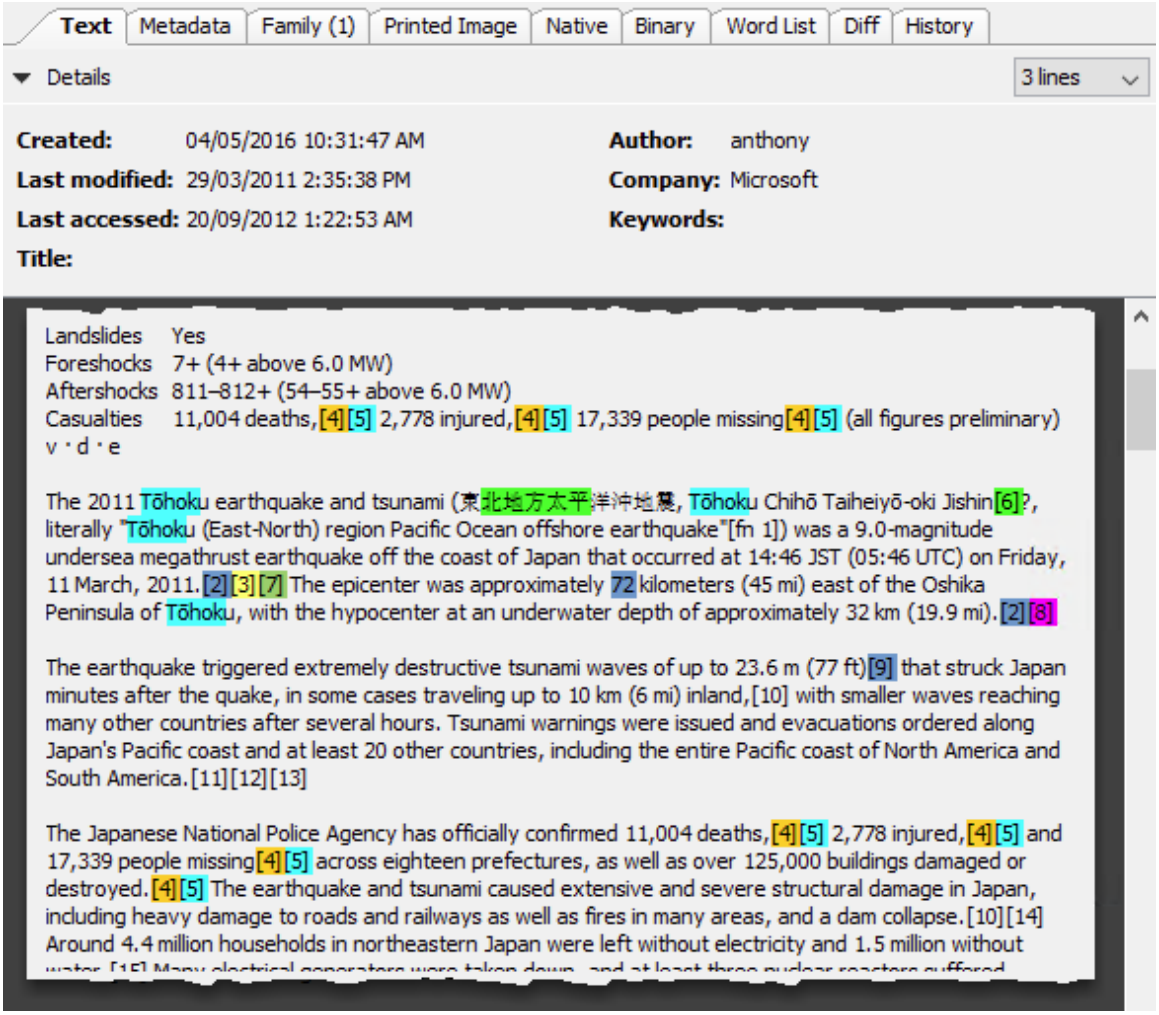
Unicode query

Unicode quotation marks U+2018 (’) and U+2019 (’) are also permitted in addition to the ASCII single quotes.



Slop query

A search performed for `[]`~1`.


Text
Metadata
Family (1)
Printed Image
Native
Binary
Word List
Diff
History
Details
3 lines
Created: 04/05/2016 10:31:47 AM
Last modified: 29/03/2011 2:35:38 PM
Last accessed: 20/09/2012 1:22:53 AM
Title:
Author: anthony
Company: Microsoft
Keywords:
Landslides Yes
Foreshocks 7+ (4+ above 6.0 MW)
Aftershocks 811–812+ (54–55+ above 6.0 MW)
Casualties 11,004 deaths,[4][5] 2,778 injured,[4][5] 17,339 people missing[4][5] (all figures preliminary)
v · d · e
The 2011 Tōhoku earthquake and tsunami (東北地方太平洋沖地震, Tōhoku Chihō Taiheiyō-oki Jishin[6]?, literally "Tōhoku (East-North) region Pacific Ocean offshore earthquake"[fn 1]) was a 9.0-magnitude undersea megathrust earthquake off the coast of Japan that occurred at 14:46 JST (05:46 UTC) on Friday, 11 March, 2011.[2][3][7] The epicenter was approximately 72 kilometers (45 mi) east of the Oshika Peninsula of Tōhoku, with the hypocenter at an underwater depth of approximately 32 km (19.9 mi).[2][8]
The earthquake triggered extremely destructive tsunami waves of up to 23.6 m (77 ft)[9] that struck Japan minutes after the quake, in some cases traveling up to 10 km (6 mi) inland,[10] with smaller waves reaching many other countries after several hours. Tsunami warnings were issued and evacuations ordered along Japan's Pacific coast and at least 20 other countries, including the entire Pacific coast of North America and South America.[11][12][13]
The Japanese National Police Agency has officially confirmed 11,004 deaths,[4][5] 2,778 injured,[4][5] and 17,339 people missing[4][5] across eighteen prefectures, as well as over 125,000 buildings damaged or destroyed.[4][5] The earthquake and tsunami caused extensive and severe structural damage in Japan, including heavy damage to roads and railways as well as fires in many areas, and a dam collapse.[10][14] Around 4.4 million households in northeastern Japan were left without electricity and 1.5 million without

Regular expression queries

Why use Regex queries

A regular expression is a powerful method that you can use to describe a search pattern. Regular Expression queries provide means for matching patterns of text strings, such as particular characters, numbers or words.

To perform a regular expression search, add the forward-slash character (/) to the start and end of the regular expression to be used in the search. For those familiar with regular expressions, the pattern is matched against each individual word, so using expressions such as the caret (^) to find the start of a line in the text is not possible. Matching is not case sensitive and literal characters must be entered in lowercase.

You can also form complex phrase queries by using spaces within the regular expressions, and “slop” for phrases within regular expressions is supported the same way as described in *Phrase Queries* as shown in the example below.

Available patterns

Some available patterns:

Syntax	Results
`\d`	A digit (0-9)
`\D`	A non-digit
`|`	Matches either the left or right side
`[]`	One of the characters within the brackets
`.`	Any character
`.*`	The same as a multiple character wildcard search
`\b`	A word boundary. Hyphenated words are broken up by word boundaries. This matches hyphen boundaries and the end of a word.
`^`	Start of a word. Will not match hyphen boundaries.
`$`	The end of a word. Will not match hyphen boundaries.

Examples

Query Syntax	Description
`/apple|orange/`	Matches all items that contain either apple or orange.
`/eat|ate apple|orange/ ~2`	Matches all items which contain either eat or ate and then apple or orange, with up to two unrelated terms separating them. Note that only whole words are matched. To match partial words, use wildcard characters.
`/gr[eao]y/`	Matches all items that contain either grey, gray or groy.
`/gr[^eao]y/`	Matches all items that contain at least one word starting with gr followed by a character that is not e, a or o, followed by y. This query would match griy and gr3y.
`/.oe.* not/`	An example of a phrase query. Matches all items that have a word starting with any letter followed by oe, optionally followed by any other characters then the word not. This query would match "does not", "joe not" and "ioexception not".
`/0\d{1,3}/`	Matches all items that start with 0 followed by 1 to 3 digits. This query would match 02, 0404, 00 and 080.
`/0\d{1,3} \d{3,4} \d{3,4}/ OR /0\d{1,3} \d{6,8}/`	Matches all items that may contain local phone number patterns. The first part of this query would match 02 2328 1929, 043 232 192 and 0404 0233 2333. The second part would match 02 23281929, 043 23221923 and 0404 023323. There are different conventions for how phone numbers are grouped, so you will probably need to adjust this query for different cases.
`/[\u0400-\u052f]*/`	Matches all Unicode Cyrillic and Cyrillic Supplement alphabet families. Note that adding the asterisk (*) will highlight whole words for some languages such as Russian and Serbian.
`/[\p{InCyrillic}\p{InCyrillic_Supplementary}]*/`	Matches all Unicode Cyrillic and Cyrillic Supplement alphabet-family using the Unicode block names.

Tip: There are a number of great resources for Regular Expressions on the internet that you can refer to for formulating character-based regex patterns for searching, example http://regexlib.com/CheatSheet.aspx

Range queries

About range queries

Range queries allow you to search for terms within a specified upper and lower range. To formulate a range query, use square brackets [], curly brackets{} or a combination of both [} surrounding the two boundary terms. Square brackets mean that the term on the corresponding side is matched (that is, the range is inclusive of that boundary term), whereas curly brackets mean that the term on the corresponding side is not matched and only the values that exist between that boundary are retrieved (that is, the range is exclusive of that boundary term).

If you wish to omit either bound, the wildcard character ("*") can be used in its place. Omitting both bounds is not possible, and a normal wildcard query should be used instead.

The keyword "TO" (or "to") can optionally be inserted between the upper and lower range to make the query more readable.

Range queries on date fields

The following additional options are available for formulating a range query based on date fields:

- Specify specific dates, which should always be entered in the "yyyyMMdd" format (4 year digits, 2 month digits, 2 day-of-month digits without spaces or punctuation.)
- Specify relative dates, using "-" or "+" followed by a number and then the units to use. The supported units are "d" (days), "w" (weeks), "m" (months) and "y" (years.).
- Specify today using the special value "TODAY", which is the same as using the value "+0D".

The following examples will help you understand this relatively complex query type better.

Examples

Query String	Results
`[Joe TO Johnathan]`	Matches items which contain "Joe", "John" or "Johnathan" somewhere in the properties or the text content.
`{Joe TO Johnathan}`	Matches items which contain "John" somewhere in the properties or the text content, but does not match "Joe" nor "Johnathan".
`[Joe TO Johnathan}`	Matches items which contain "John" or "Joe" somewhere in the properties or the text content, but does not match "Johnathan".
`comm-date:[20070101 TO 20070131]`	Matches items which are inside a top-level communication which was sent in January 2007.
`item-date:[20120101 TO *]`	Matches items which are dated on or after January 2012.
`date-properties:"File Modified":[* TO -7Y]`	Matches items with a "File Modified" property older than 7 years.
`date-properties:"File*":[* TO -7Y]`	Matches items with a "File" property older than 7 years, e.g. "File Modified", "File Created", etc.
`date-properties:"*":[* TO -7Y]`	Matches items with any date property older than 7 years.

Field queries

About field queries

A single-word query locates a search term within text contained in the content of a document as well as within the properties of the document. Field queries allow you to restrict your search to specific properties of documents through the use of field names within the search query.

If you perform a single-word search for the term “wow”, your search results will include items that have this word in the content or in the properties of the documents returned. To restrict this search to a specific field, prefix the search term with the field name followed by the ":" symbol. For example, if you want to search for “wow” in the “name” field only, using the search term "*name: wow*" will locate the items whose name field contains the term "wow", but it will not locate items which contain the term "wow" in the text content only.

For more information about which fields can be used in field queries see the Indexed fields section.

Examples

Query String	Results
`name:Embedded`	Matches items that contain "Embedded" in their names, such as "Embedded Item 1" or "Embedded Item 2"
`name:( Embedded AND 1)`	Matches items that contain both "Embedded" and "1" in their names, such as "Embedded Item 1" or "Embedded Image 1"
`name:( picture* OR itext* )`	Matches items that contain words beginning with "picture" or "itext" in their names, such as "Some more pictures", "Picture 1" or "itext6"
`properties:" AAA|BBB" properties:" AAA BBB"`	Matches items that contain nested metadata properties AAA as the parent folder and BBB as the child folder. When you search for the "AAA|BBB" metadata name, the search queries "AAA|BBB" and "AAA BBB"- both return the correct item. This is currently supported for Hancom data.

Search operators

Nuix allows for different types of queries to be combined with the help of search operators to refine search results. Two or more queries can be combined with a combination of the operators.

- Logical or Boolean operators
- Proximity operators

Operators within a query are governed by precedence and grouping which can dramatically affect the results returned. Precedence refers to the order in which the operators are processed in a query. Grouping refers to combining two or more expressions in a query using parentheses to affect the order or precedence to retrieve the desired data.

Tip: It is extremely important to understand how both governance and precedence affect searching and it is recommended to explicitly use both in constructing search terms to ensure the expected results are returned.

Operator precedence

Query operators execute in the following precedence ordering when used in a search:

Order	Operator	Notes
1.	`Slop suffix ('~n')`	It can only be used with quoted queries, quoted wildcard queries, exact queries or regular expression queries.
2.	`Fuzzy suffix ('~')`	It can only be used with unquoted queries.
3.	`Groups ( '(' ... ')' )`	-
4.	`Field prefix ('field:')`	-
5.	`Alternate logical operators (‘+’, '-')`	
6.	`NOT`	-
7.	`W/n, PRE/n, NOT W /n, NOT PRE/n`	-
8.	`AND`	-
9.	`OR`	-

Note: In instances where no operator has been explicitly used between two terms, the default behavior is to use the **AND** operator between the terms.

Examples:

Query String	Results
`John AND Jenny OR Tom AND Jerry`	Operator precedence means this is equivalent to (John AND Jenny) OR (Tom AND Jerry), and is not equivalent to ((John AND Jenny) OR Tom) AND Jerry

Operator grouping

Use parentheses to group expressions in order to execute operators, different from the default order to produce the desired query. The expression within parentheses will be evaluated as a separate query first, and with the results from that query then used against the expression not enclosed in parentheses.

Examples:

Query String	Results
`(Joe AND Bloggs) OR Smith`	Matches all items that contain both Joe and Bloggs, or only Smith (so it would match an item that contains the phrase "Keith Smith").
`Joe AND (Bloggs OR Smith)`	Matches all items that contain both Joe, and either Bloggs or Smith (so it would not match "Keith Smith", but it would match "Joe Bloggs").

Logical/Boolean operators

Boolean operators enable you to refine your searches to achieve specific results set.

Nuix supports the following Boolean operators: **AND**, **OR**, and **NOT**. While you can chain together any number of logical ANDs or ORs, combining the various operators together can lead to ambiguity in search syntax. In such cases, it is recommended to use parentheses to clarify the order of operations with the innermost pair of parentheses being performed first, followed by the next most inner pair working outwards until all operations within parentheses are complete. Any remaining operations outside the parentheses are then performed last.

AND operator

The AND operator can be used to combine two or more search terms to match only those items that include all of the individual terms.

The AND operator is not case sensitive, so a lowercase "and" in queries is also recognized as the AND operator.

You can combine the AND operator with other types of search syntax. For example, you can use AND in between terms that use a wildcard or a fuzzy search.

If you use two single terms in a query without an operator in between and not enclosed by quote marks, by default Nuix will search for the terms using the AND operator between them. Another syntax for the AND operator is to add the plus (+) symbol to additional terms you want to include in the search; for example, **insider AND trading AND options** is the same as **insider +trading +options**.

Examples

Query String	Results
`Joe AND Bloggs`	Matches items that contain both "Joe" and "Bloggs".
`Joe Bloggs`	Matches the same items as the previous query, because AND is the default operator.
`Joe +Bloggs`	Matches the same items as the previous two queries (alternative syntax).
`J* AND Bloggs`	Matches items that contain both text starting with "J" and the full word "Bloggs".
`Joe~ AND Bloggs`	Matches items that match the fuzzy search results for "Joe" and the full word "Bloggs".

OR operator

The OR operator can be used to combine two or more search terms to match items that include either of the individual terms within them.

The OR operator behaves much like the AND operator with respect to combining with other queries.

Examples

Query String	Results
`Joe OR Bloggs`	Matches items that contain either "Joe" or "Bloggs" (or both).

NOT operator

Use the NOT operator to combine two or more search terms to match items that include only the first term, but do not include the second term.

The NOT operator behaves like the AND operator with respect to mixing with other queries.

Another syntax for the NOT operator is to add the minus (-) symbol to the terms you want to exclude from the search.

Examples

Query String	Results
`Joe NOT Bloggs`	Matches items that contain "Joe", but not "Bloggs".
`Joe -Bloggs`	Matches the same items as the previous query (alternative syntax).

XOR (Exclusive OR) operator

Nuix does not support an explicit XOR operand. However, because it does support the use of parentheses, you can construct equivalent queries by using NOT and OR in the same query.

Examples

Query String	Results
`(Joe NOT Bloggs) OR (Bloggs NOT Joe)`	Matches items that contain either "Joe" or "Bloggs", but not both. That is, it could return matches containing Joe Smith or Karen Bloggs.

Proximity operators

Proximity operators are operators that can be used to combine two or more search terms to match items based on their distance from each other. The distance parameter is specified along with the desired proximity operator in the query. Nuix supports the following proximity operators:

- W/n
- PRE/n
- NOT W/n
- NOT PRE/n

Note: It is important to note that in the legal industry most proximity searching is often associated with a particular product. Proximity searching within Nuix has been implemented to work in the same fashion as would be expected by industry-standard proximity searches, but does not use any third-party tools to do so. This will usually result in slightly differing results if compared with another tool unless the indexes have been built in exactly the same way as Nuix builds its index including additional property fields.

W/n

A search combining two or more search terms using the W/n operator matches only items that are near each other within the maximum permissible number of terms specified by the parameter n. One term is matched within that distance of the other and is symmetrical as a search term.

The operator is not case sensitive, so a lowercase "w/n" in queries behaves the same way as the same operator in uppercase.

Much like the AND operator, the W/n operator can be used to combine most of the above search queries, in particular though, NOT queries are not permitted. Complex nested Boolean queries are permitted but are unlikely to give meaningful results.

Examples

Query String	Results
`Joe W/4 Bloggs`	Matches items that contain "Joe" and "Bloggs" occurring within a maximum distance of 4 terms apart (a maximum of 3 terms can occur between the two).
`(John OR Johnny) W/2 Smith`	Matches items that contain either "John" or "Johnny" occurring within a maximum distance of 2 words from "Smith".
`(John AND Mary) W/2 Smith`	Matches items that contain both "John" and "Mary" occurring within a maximum distance of 2 words from "Smith".

PRE/n

The PRE/n operator works similarly to the W/n operator with the difference being that the matches must occur in the order in which they are specified.

The operator is not case sensitive, so a lowercase "pre/n" in queries behaves the same way as the same operator in uppercase.

Examples

Query String	Results
`Joe PRE/3 Bloggs`	Matches items that contain "Joe" and "Bloggs", with "Bloggs" occurring within the three terms **after** "Joe".
`(John OR Johnny) PRE/2 Smith`	Matches items that contain either "John" or "Johnny", with "Smith" occurring within the 2 words **after** matching either of the first terms.

NOT W/n and NOT PRE/n

By adding NOT in front of W/n or PRE/n, the proximity rule is reversed.

This operator is not case sensitive, so a lowercase "not" in queries behaves the same way. Along with the restrictions already noted for proximity queries above, using NOT proximity queries with nested AND queries is not allowed.

Examples

Query String	Results
`"Acme Corporation" NOT W/3 Copyright`	Matches items that contain any instances of "Acme Corporation" which is not within 3 terms of "Copyright". This is different from normal Boolean queries in that the same document might have some instances of "Acme Corporation" within 3 terms of "Copyright" and other instances that are not which would be hit in that case.
`Acme NOT PRE /1 ( Corporation OR Inc)`	Matches items that contain "Acme" but without "Corporation" or "Inc" as the next term.

Escaping special characters

This search operator is used when you need to include special characters in your search. A backslash (\) is used to escape special characters.

For fields like "content" where these special characters are also treated as punctuation anyway, this isn't particularly useful, but it can be useful for fields like "tags" where natural language analysis is not performed.

Example

Query String	Results
`tag:Blog\*`	Matches all items with the tag "Blog*".

Indexed fields

Indexed Fields in Nuix enhance searching by providing specific fields that can be searched directly. Some of these fields are convenience fields that have been calculated on indexing and stored for later use like the **Item Date** field whereas some are simply stored as a specific property to make searching for them faster like the **Name** field. A simple single search term searches both text content and metadata properties. An indexed field search restricts the search to only the content of the metadata property specified.

To restrict a search to a particular indexed field, prefix the term for which you are searching with the field name followed by a colon (:). For example, the search expression *name:“nuix”* locates the items whose name contain the term "nuix", but it will not locate items that contain the term "nuix" only in the text content.

Note: The field search only works against the word that directly follows the colon. If you want to search for a phrase, the phrase must be enclosed in quotes. For example, if you want to search for "Options to sell" in the subject of an email or in the name of an item, you would use name:"Options to sell".

Indexed fields are further classified into the following categories, based on field attributes:

- Common fields
- Annotation Fields
- Communication fields
- GUID fields
- List-based fields

These classifications and the use of associated attributes for searching are explained in this section.

Common fields

Use Nuix Common fields to search for additional attributes about the item (metadata), which are not necessarily part of the content or the item itself. Given below are the different types of common fields available in Nuix including working examples of each. The fields are listed in alphabetical order for easy navigation.

audited-size | boolean-custom-metadata | boolean-custom-metadata-exact | boolean-properties | characters | content | custom-metadata | custom-metadata-exact | date-custom-metadata | date-custom-metadata-exact | date-properties | digest-input-size | duration-properties | evidence-metadata | external-deduplication | family | family-content | family-names | family-properties | file-extension | file-size | flag | float-custom-metadata | float-custom-metadata-exact | float-properties | fuzzy-hash | has-binary | has-communication | has-embedded-data | has-image | has-printed-image | has-stored | has-text | imaging-profile | integer-custom-metadata | integer-custom-metadata-exact | integer-properties | item-date | item-id | item-number | kind | lang | location-2d | md5, md5-original, md5-latest, sha-1, sha-1-original, sha-1-latest, sha-256, sha-256-original, sha-256-latest (digests) | mime-type | mime-type-tag | modifications | name | named-entities | nlp | path-kind | path-mime-type | path-mime-type-tag | path-name | photo-dna | print-method | printed-image-page-count | properties | shannon-entropy | skintone | text-custom-metadata | text-custom-metadata-exact | top-level-item-date

audited-size

The audited-size field is used to search for audited items of a particular size (in bytes). Items that have been marked for auditing (non-immaterial items) can be identified with the search flag:audited.

Examples

Query String	Results
`audited-size:789`	Matches all audited data items with a size of 789.
`audited-size:[400 TO 789]`	Matches all audited data items with a size from 400 to 789.
`audited-size:*`	Matches all data items with an audited size, although flag:audited will run quicker. From Workstation 5.0 onwards, the option to calculate audited size must be enabled.
`-audited-size:*`	Matches all data items without an audited size, although -flag:audited will run quicker.

boolean-custom-metadata

The Boolean-custom-metadata field is used to search custom metadata fields that are of a Boolean type that are associated with data items. You can use "yes" or "1" as an alias for identifying the value “true”, and "no" or "0" as an alias for identifying the value “false”.

The wildcard character "*" can be used to match "any" in either field name or value. Partial wildcard matching within a name or field (e.g. "somePrefix") is not supported.

Examples

Query String	Results
`boolean-custom-metadata:"follow up":true`	Matches data items that have the Boolean custom metadata field "follow up" with the value true.
`boolean-custom-metadata:"follow up":1`	
`boolean-custom-metadata:"follow up":*`	`Matches data items that have the Boolean custom metadata field "follow up" including null values.`
`boolean-custom-metadata:*:0`	`Matches data items that have any Boolean custom metadata field with the value false.`

boolean-custom-metadata-exact

Same as *boolean-custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

boolean-properties

The boolean-properties is used to search over the boolean properties associated with every item of data.

Example

Query String	Results
`boolean-properties:compressed:true`	Matches items with a "compressed" property set to true.

characters

The characters field is used to search for items that contain characters of the character sets listed below.

Type	Explanation
arabic	Includes Arabic characters.
chinese	Includes Chinese characters that are shared with other languages, such as Japanese and Korean.
cyrillic	Includes many East and South Slavic languages, and almost all languages in the former Soviet Union.
hangul	Includes characters from the native alphabet of the Korean language.
japanese	Includes the Japanese Hiragana and Katakana alphabets.
non-latin	Includes any characters not found in common Latin (English, Spanish, German, etc.) text.

Examples

Query String	Results
`characters:japanese`	Matches all items containing Hiragana or Katakana characters.
`characters:non-latin`	Matches all items that contain non-Latin characters.

content

The content field searches text content within the email body or the text portion of a document.

Example

Query String	Results
`content:wow`	Matches all items that contain the term "wow" in the email body or text portion of a document, essentially the Text tab of the Preview pane.

custom-metadata

Use this field to search across all the custom metadata fields associated with every item of data. If a value is provided, it is interpreted by every specific custom metadata type that can satisfy it. For example, the search term "true" will be searched in only text and Boolean entries, while "1" will be searched in text, Boolean, integer (whole number), and float (decimal) fields because they can all interpret the value. This search field is designed as a "catch-all" option. Therefore, if you know the type of custom metadata for which you are searching, it would be more effective to use one of the more specific custom metadata search fields.

Single value queries (or no value specified) search all custom metadata types, while range queries search only integer, float, and date custom metadata types.

The wildcard character "*" can be used to match "any" in either field name or value. If partial wildcard matching is used for a value (e.g. "somePrefix"), only text custom metadata fields are searched.

Examples

Query String	Results
`custom-metadata:"char count":15234`	Matches data items that have the custom metadata field "char count" of any type (text, whole number or decimal) with the value 15234.
`custom-metadata:"char count":*`	Matches data items that have the custom metadata field "char count" of any type with any value.
`custom-metadata:*:true`	Matches data items that have any custom metadata field of any type (text or Boolean) with the value true.
`custom-metadata:"char count":[14000 TO 16000]`	Matches data items that have the custom metadata field "char count" of any type (whole number or decimal) with a value between 14000 and 16000 inclusive.
`custom-metadata:"char count":{14000 TO 16000}`	Matches data items that have the custom metadata field "char count" of any type (whole number or decimal) with a value between 14000 and 16000 exclusive.
`custom-metadata:*:[20021118 TO 20021119]`	Matches data items that have any custom metadata field of any type inclusive (whole number, decimal or date) with a value between 20021118 and 20021119 inclusive.
`custom-metadata:"`	

custom-metadata-exact

Same as *custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

Query String	Results
`custom-metadata:"layer1|layer2":"elem2"`	Matches data items that have the custom metadata property name whose nested metadata property value matches the "elem2".

For a nested metadata property, add the 'Field name'.

date-custom-metadata

The date-custom-metadata field searches over the date number custom metadata fields associated with every item of data.

The wildcard character "*" can be used to match "any" in either field name or value. Partial wildcard matching (e.g. "somePrefix") is not supported.

Examples

Query String	Results
`date-custom-metadata:"last saved":20021118`	Matches data items that have the date number custom metadata field "last saved" with the value 2002-11-18.
`date-custom-metadata:"last saved":*`	Matches data items that have the date number custom metadata field "last saved" with any value.
`date-custom-metadata:*:20021118`	Matches data items that have any date number custom metadata field with the value 2002-11-18.
`date-custom-metadata:"last saved":[20021118 TO 20021119]`	Matches data items that have the date number custom metadata field "last saved" with a value between 2002-11-18 and 2002-11-19 inclusive.
`date-custom-metadata:"last saved":{20021118 TO 20021119}`	Matches data items that have the date number custom metadata field "last saved" with a value between 2002-11-18 and 2002-11-19 exclusive.
`date-custom-metadata:*:[20021118 TO 20021119]`	Matches data items that have any date number custom metadata field with a value between 2002-11-18 and 2002-11-19 inclusive.

date-custom-metadata-exact

Same as *date-custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

date-properties

The date-properties field searches over the date properties associated with every item of data.

Examples

Query String	Results
`date-properties:"last saved":20021118`	Matches data items that have the date property "last saved" with the value 2002-11-18.
`date-properties:"last saved":[20021118 TO 20021119]`	Matches data items that have the date property "last saved" with the value between 2002-11-18 and 2002-11-19.
`date-properties:"last saved":[20021118T093020 TO 20021119T093040]`	Matches data items that have the date property "last saved" with the value between 2002-11-18 9:30:20 AM and 2002-11-19 9:30:40 AM, i.e. down to the second.
`date-properties:"last saved":[20021118T093020.111 TO 20021119T093040.222]`	Matches data items that have the date property "last saved" with the value between 2002-11-18 9:30:20.111 AM and 2002-11-19 9:30:40.222 AM, i.e. down to the millisecond.

digest-input-size

The digest-input-size field can be used to search for items of a specific digest size (in bytes).

Examples

Query String	Results
`digest-input-size: 789`	Matches all data items with a size of 789 bytes.
`digest-input-size: [400 TO 789]`	Matches all data items with a size from 400 to 789 bytes.
`digest-input-size:*`	Matches all data items with a computed size.
`-digest-input-size:*`	Matches all data items without a computed size, which includes directories and evidence folders.

duration-properties

The duration-properties searches over the duration properties associated with every item of data. The duration specified in the search is in milliseconds.

Examples

Query String	Results
`duration-properties:"Edit Time":5000`	Matches items with an edit time of 5 seconds.
`duration-properties:"Edit Time":[60000 TO *]`	Matches items with an edit time longer than 60 seconds.

evidence-metadata

The evidence-metadata field can be used to search any custom metadata that the investigator added to the case when the evidence was loaded. In these examples, "site" is a piece of evidence metadata.

Examples

Query String	Results
`evidence-metadata:site`	Matches items whose top-level evidence folder has the word "site" in the name or value.
`evidence-metadata:" site: 23 Dickson Street, Canberra"`	Matches items whose top-level evidence folder contains the value "23 Dickson Street, Canberra" for the "site" metadata field. This actually matches some other things, such as "site 23 Dickson Street Canberra" all in the value, since the colon character and other punctuation are ignored in the query.

external-deduplication

The external-deduplication field can be used to search for items that have been deduplicated externally.

Examples

Query String	Results
`external-deduplication:original`	Matches data items that have "External deduplication key" custom metadata field of text type with the value "original". 1/true/yes and 0/false/no are interpreted as boolean values, otherwise interpreted as text.
`external-deduplication:true`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value true.
`external-deduplication:1`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value true.
`external-deduplication:yes`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value true.
`external-deduplication:false`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value false.
`external-deduplication:0`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value false.
`external-deduplication:no`	Matches data items that have "External deduplication key" custom metadata field of boolean type with the value false.

family

The family field can be used to search over the name, content; property names and values associated with data items that were processed with this option in Evidence Processing Settings and returns only the associated top-level item for any hits. This is a field that automatically searches over the family-content, family-name and family-properties fields.

Example

Query String	Results
`family:Bloggs`	Matches top-level items which have a family item with Bloggs in any name, content, property name or property value.

family-content

The family-content field can be used to search over item contents associated with data items that were processed with this option in Evidence Processing Settings, and returns only the associated top-level item for any hits.

Example

Query String	Results
`family-content:wow`	Matches all top-level items which have a family item containing the term "wow" in the text content.

family-names

The family-names field is associated with data items that were processed with this option in Evidence Processing Settings and returns only the associated top-level item for any hits.

Example

Query String	Results
`family-names:" Check this out"`	Matches all top-level items which have a family item containing "Check this out" in their name that were processed with this field setting.

family-properties

The family-properties field can be used to search over the property names and values associated with data items that were processed with this option in Evidence Processing Settings, and returns only the associated top-level item for any hits.

Examples

Query String	Results
`family-properties: Bloggs`	Matches top-level items processed with this field setting, which have a family item with the name Bloggs in any property name or property value.
`family-properties:" Author: Joe Bloggs"`	Matches top-level items processed with this field setting, which have a family item which contain the value "Joe Bloggs" for the "Author" property. This actually matches some other things, such as "Author Joe Bloggs" all in the value, since the colon character and other punctuation are ignored in the query.

file-extension

The file-extension field can be used to search for files by their original extension.

Examples

Query String	Results
`file-extension:doc`	Matches data items whose original file extension was "doc".
`file-extension:*`	Matches data items which were identified as having a file extension when originally processed.

file-size

The file-size field can be used to search for files of a specific size (in bytes).

Examples

Query String	Results
`file-size:789`	Matches all file sizes of 789 bytes.
`file-size:[400 TO 789]`	Matches all file sizes from 400 to 789 bytes.
`file-size:*`	Matches all files with a size.
`-file-size:*`	Matches all files without any size.

flag

The flag field can be used to search for items that were flagged by Nuix as being of a particular type when processed.

Examples

Query String	Results
`flag:irregular_file_extension`	Matches all data items marked as having an irregular file extension.
`flag:inline`	Matches all data items marked as being displayed as a part of an outer item. An example is an image inside an RTF document.
`flag:partially_processed`	Indicates that the item's children were only partially processed. Some children were explicitly skipped in the direction of the user.
`flag:not_processed`	Indicates that the item was identified but not processed any further.

`flag:text_not_processed`	Indicates that the item's text was not processed. The item's metadata was still processed. If **Process text** option is disabled in **Data Processing Settings**, the items are flagged with ‘flag:text_not_processed’ since Nuix 6.0.
`flag:text_indexed`	Indicates the item's text aspect was obtained successfully and indexed into the text index. The text aspect may have been obtained via properly parsing the format or by text stripping the binary content. Since Nuix 6.2.
`flag:text_not_indexed`	Indicates the item's text aspect was not stored into the text index. Either the item had no way to expose text for processing (for example, it was a metadata only item, such as a call record), the text wasn't processed, or an error occurred. The item's metadata was still processed.
`flag:images_not_processed`	Indicates that the item's images were not processed.
`flag:identification_disabled`	Indicates that identification was disabled when the item was processed.
`flag:poison`	Matches all data items that caused a critical error during processing after several attempts.
`flag:text_stripped`	Matches all data items whose text was achieved via text stripping rather than normal text extraction methods.
`flag:audited`	Matches all data items marked to be audited for calculating the total size for audited licences. These items would be exported in a legal export.
`flag:top_level`	Matches all data items marked as being top-level items.
`flag:not_top_level`	Matches all data items marked as not being top-level items.
`flag:loose_file`	Matches all data items marked as loose files, which are the files you would see, for example, in Windows explorer when browsing a directory. Loose files within a disk image are also marked with this flag.
`flag:not_loose_file`	Matches all data items marked as not being loose files.
`flag:physical_file`	Matches all data items marked as physical files, which are the highest items in the data tree which contain binary. These will correspond to the "real files" that were loaded into the case. As a consequence, this can flag disk images and forensic containers unlike the loose_file flag.
`flag:not_physical_file`	Matches all data items which have been marked as not being physical files.

`flag:licence_restricted`	Matches all data items which have not been processed fully due to a licence restriction.
`flag:reloaded`	Matches all data items which have been reloaded from source data or a new binary file.
`flag: suppressed_immaterial_item`	Matches all data items which contain an immaterial item that has not been exposed as a separate data item. This is only present when the "Hide Immaterial Items" processing option is enabled.
`flag:encrypted`	Matches all data items which were encrypted.
`flag:decrypted`	Matches all data items that were encrypted but successfully decrypted by Nuix.
`Flag: deleted`	Matches items that were deleted, and recovered from slack space.

float-custom-metadata

The float-custom-metadata field searches over the decimal number custom metadata fields associated with every item of data.

The wildcard character "*" can be used to match "any" in either field name or value. Partial wildcard matching (e.g. "somePrefix") is not supported.

Examples

Query String	Results
`float-custom-metadata:"kilobytes":0.4`	Matches data items that have the decimal number custom metadata field "kilobytes" with the value 0.4.
`float-custom-metadata:"kilobytes":*`	Matches data items that have the decimal number custom metadata field "kilobytes" with any value.
`float-custom-metadata:*:0.4`	Matches data items that have any decimal number custom metadata field with the value 0.4.
`float-custom-metadata:"kilobytes":[0.3 TO 1.5]`	Matches data items that have the decimal number custom metadata field "kilobytes" with a value between 0.3 and 1.5 inclusive.
`float-custom-metadata:*:[0.3 TO 1.5]`	Matches data items that have any decimal number custom metadata field with a value between 0.3 and 1.5 inclusive.

float-custom-metadata-exact

Same as *float-custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

float-properties

The float-properties field can be used to search over the fractional number properties associated with data items.

Example

Query String	Results
`float-properties:"horizontal pixel size":[0.3 TO 0.4]`	Matches data items that have the fractional number property "horizontal pixel size" with the value between 0.3 and 0.4.

fuzzy-hash

The fuzzy-hash field can be used to search on the SSDeep fuzzy hash values of the data items. Fuzzy hashes take the form *blocksize*:*hash1*:*hash2* where *blocksize* is an integer, and *hash1* and *hash2* are base-64 strings.

A search can accept an optional minimum score supplied as an integer between 0 and 100 inclusive. Moreover, a range of acceptable scores can be supplied.

Wildcard characters can be used in queries that do not contain a threshold score.

Examples

Query String	Results
`fuzzy-hash:*`	Matches all items that have a computed fuzzy hash value.
`fuzzy-hash:"48:*"`	Matches all items with a fuzzy hash value computed with a block size of 48.
`fuzzy-hash:"48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM"`	Matches all items with the exact fuzzy hash value of 48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM.
`fuzzy-hash:"48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQn?Pu1otM"`	Matches all items with the 64 possible fuzzy hashes implied by the wildcard character ?.
`fuzzy-hash:"48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM;75"`	Matches all items that score at least 75 (out of 100) against the fuzzy hash 48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM.
`fuzzy-hash:"48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM;[75 TO 95]"`	Matches all items that score between 75 and 95 (inclusive) against the fuzzy hash 48:ZyY6uERAD3iU77c/rQxZ7YPuqI7LqB2X+c8HxM:ZlEmlIjQnYPu1otM.

has-binary

The has-binary field is a Boolean field which can be used to search for items that could have binary data. Items such as files and email messages have binary data while few types of items such as file folders /directories, mail folders or folders inside compressed zip files lack binary data.

Examples

Query String	Results
`has-binary:1`	Matches all items with binary data.
`has-binary:0`	Matches all items without binary data.

has-communication

The has-communication field is a Boolean field that can be used to search for items that have communication-based data. This search matches items that are communication items, such as email or instant messages, but not the items that are attached to, or associated with them. To search for attachments, see the *Communication Fields* category.

Examples

Query String	Results
`has-communication:1`	Matches all items that contain communications fields (To, Cc, Bcc, From fields).
`has-communication:0`	Matches all items that do not contain communication fields (To, Cc, Bcc, From fields).

has-embedded-data

The has-embedded-data field is a Boolean field that can be used to search for items that could contain embedded data. Using this search matches items that have the ability to contain embedded data. For instance, it will match all directories even if a directory does not contain any file.

Note: This search will retrieve only the items that have the ability to contain embedded items, **not** the embedded items.

Examples

Query String	Results
`has-embedded-data:1`	Matches all items that could contain embedded items. This does not mean that the item *does* contain embedded items.
`has-embedded-data:0`	Matches all items that cannot contain embedded items.

has-image

The has-image field is a Boolean field that can be used to search for items that contain image data.

Examples

Query String	Results
`has-image:1`	Matches all items that could contain images.
`has-image:0`	Matches all items that cannot contain images.

has-printed-image

The has-image field tests if this item has a printed image. This field contains either 0 or 1 and thus similar query search as the *has-binary* field.

has-stored

The has-stored field can be used to match items that have the stored data type specified by the query below.

Examples

Query String	Results
`has-stored:binary`	Finds all items that have stored binaries.
`has-stored:pdf`	Finds all items that have stored PDFs.
`has-stored:text`	Finds all items that have stored text.
`has-stored:thumbnail`	Finds all items that have stored thumbnails.

has-text

The has-text field is a Boolean field that can be used to search for items that could contain text data. This type of search does not imply the document has text, but rather just that the item type could contain text.

Examples

Query String	Results
`has-text:1`	Matches all items that contain text.
`has-text:0`	Matches all items that contain no text.

imaging-profile

The imaging-profile field is used to match items that have the imaging profile specified by the query.

Example

Query String	Results
`imaging-profile:"my imaging profile"`	Finds all items that have the imaging profile specified by the query.

integer-custom-metadata

The integer-custom-metadata field can be used to search over the whole number custom metadata fields associated with data items.

The wildcard character "*" *can be used to match "any" in either field name or value. Partial wildcard matching (e.g. "somePrefix*") is not supported.

Examples

Query String	Results
`integer-custom-metadata:"char count":15234`	Matches data items that have the whole number custom metadata field "char count" with the value 15234.
`integer-custom-metadata:"char count":*`	Matches data items that have the whole number custom metadata field "char count" with any value.
`integer-custom-metadata:*:15234`	Matches data items that have any whole number custom metadata field with the value 15234.
`integer-custom-metadata:"char count":[14000 TO 16000]`	Matches data items that have the whole number custom metadata field "char count" with a value between 14000 and 16000 inclusive.
`integer-custom-metadata:"char count":{14000 TO 16000}`	Matches data items that have the whole number custom metadata field "char count" with a value between 14000 and 16000 exclusive.
`integer-custom-metadata:*:[14000 TO 16000]`	Matches data items that have any whole number custom metadata field with a value between 14000 and 16000 inclusive.

integer-custom-metadata-exact

Same as *integer-custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

integer-properties

The integer-properties field can be used to search over whole number properties associated with data items.

Examples

Query String	Results
`integer-properties:"char count":15234`	Matches data items that have the whole number property "char count" with the value 15234.
`integer-properties:"char count":[14000 TO 16000]`	Matches data items that have the whole number property "char count" with the value between 14000 and 16000.

item-date

The item-date field can be used to search for an item by the date of the item. The date of the item will generally be the communication date for items which are a communication item or the modified date for other types of item. Items without a date inherit the date of their parent.

This field is most useful when used in conjunction with a range query.

Note: This field will only be present on cases created with Nuix version 3.0 and above. Using it in earlier cases will not return any search result.

Examples:

Query String	Results
`item-date:20070101`	Matches items with date of 1 January, 2007.
`item-date:[20070101 TO 20070131]`	Matches items with date of January 2007.
`item-date:[20070101T093020 TO 20070131T093040]`	Matches items with date of January 2007 between 9:30:20 AM and 9:30:40 AM, i.e. down to the second.
`item-date:[20070101T093020.111 TO 20070131T093040.222]`	Matches items with date of January 2007 between 9:30:20.111 AM and 9:30:40.222 AM, i.e. down to the millisecond.

item-id

The item-id field can be used to search for a short ID that is unique for the case.

Examples:

Query String	Results
`item-id:11234`	Matches the data item with the given id.
`item-id:0-1-1234`	Matches the data item, contained in a compound case, with the given id. In this case, the item is within a hierarchy two levels deep.

item-number

The item-number field can be used to search over the item number fields associated with items that have had them applied.

Item numbers are stored as text custom metadata, therefore you can perform wildcard searching like *text-custom-metadata* but does not support range searching like *integer-custom-metadata*. The wildcard character "*" can be used to match "any" number. Partial wildcard matching (e.g. "someNumberPrefix*") is supported for numbers.

In simple cases, text is searched using an SQL style "LIKE" syntax, but with "*" instead of "%". Queries are not tokenized as they are for text searching, rather SQL style phrase matching is used. In Elasticsearch cases, queries are tokenized and searched in the same way as the text index and all searches are case-sensitive.

Examples

Query String	Results
`item-number:DOC-123`	Matches items that have the item number "DOC-123".
`item-number:DOC-12*`	Matches items that have item numbers starting with "DOC-12".
`item-number:*12345`	Matches items that have item numbers ending with "12345".
`item-number:*`	Matches items that have any item number.

kind

The kind field can be used to search for items based broadly on the type of data they contain. This is similar to using the mime-type field, but simpler to use.

The supported categories of kind types are:

Kind	Explanation
email	Email messages
calendar	Meetings, appointments and other things that appear on a calendar
contact	Contacts, like virtual business cards
document	Documents, mostly (but not entirely) word processor formats
spreadsheet	Spreadsheets
presentation	Presentations, also known as slide shows
drawing	Vector drawings and diagrams
other-document	Other types of document a user might create
image	Bitmap (raster) images
multimedia	Audio and video files
database	Structured database files, such as Microsoft Access
container	Data types which resemble directories, such as archives or mailboxes
system	System files
unrecognised	Files of a type not detected by the software

Example

Query String	Results
`kind:email`	Matches all email messages.

lang

The lang field can be used to match items that have been identified as containing text in specified languages. The value is ISO 639-3 code. The full list of languages identified and their corresponding codes can be found here.

Language type is identified during ingestion and is based on the majority language found in the text content of the first 8 kb of items that are deemed to have usable text, for example, documents and emails. Only the majority language found is recorded as the language although the item may contain multiple languages.

Examples

Query String	Results
`lang:eng`	Matches all items that contain English text.
`lang:jpn`	Matches all items that contain Japanese text.
`lang:(rus OR ukr)`	Matches all items that contain Russian or Ukrainian text.

location-2d

The location-2d search field matches items which have Latitude and Longitude properties within the specified geographical perimeter. This field only supports `geodistance` queries in the form below. The first 2 parameters are latitude and longitude (in degrees) followed by distance in kilometers.

Examples

Query String	Results
`location-2d:GEODISTANCE ((50.0 20.0) 10.0)`	Matches items that are within 10 km from point 50N 20E.
`location-2d:GEODISTANCE ((50N 20.0E) 10km)`	Matches items that are within 10 km from point 50N 20E.
`location-2d:GEODISTANCE ((-33.856784 151.215737) 3.14mi)`	Matches items that are within 3.14 miles from point -33.856784 151.215737.

md5, md5-original, md5-latest, sha-1, sha-1-original, sha-1-latest, sha-256, sha-256-original, sha-256-latest (digests)

Digest fields, md5, md5-original, md5-latest, sha-1, sha-1-original, sha-1-latest, sha-256, sha-256-original, sha-256-latest can be used to search across the digests of items.

These fields can be used to find data items with content identical to other items in the same data set, and also items outside the data set.

Note: Due to the nature of digests, queries on these fields may return data items that are not actually identical to the data item for which you are looking. This is mostly applicable to email items because only a subset of email data is used to generate the digest.

By default, Nuix Workstation will compute digests only on files less than 256MB in size for faster processing unless this value is increased in Evidence Processing Settings on ingestion. If the value is not increased, items over 256MB will have a null digest value.

Digests supported by Nuix Workstation have lengths as detailed in the following table. The number of hexadecimal digits represents how many digits will come after the colon when using this field in a search query.

Digest	Bits	Hexadecimal Digits
md5	128	32
md5-original		
md5-latest		
sha-1	160	40
sha-1-original		
sha-1-latest		
sha-256	256	64
sha-256-original		
sha-256-latest		

Note:

When an item is reloaded, the latest value is stored in the "-latest" field as well as its "active" field (that is, md5, sha-1, and sha-256).

When an item is updated using File Import Import Replacement Files, the -latest field will be unchanged but its active field will be updated.

Tip: Wildcard searching is possible with digest identifiers.

Examples

Query String	Results
`md5: 678467f81cf6275822396e8fab08df31`	Matches all items with the md5 digest "678467f81cf6275822396e8fab08df31".
`md5:678*`	Matches all items with the md5 digest that contains the digits “678” anywhere in the digest identifier.

`md5-original: 678467f81cf6275822396e8fab08df31`	Matches all items with the md5 digest "678467f81cf6275822396e8fab08df31" when the item is loaded for the first time.
`md5-latest: 678467f81cf6275822396e8fab08df31`	Matches all items with the md5 digest "678467f81cf6275822396e8fab08df31" after the item is reloaded.
`sha-1:354d8b33aa51aed2a7fcb8 ad5476a5d5ede8bb2a`	Matches all items with the sha-1 digest "354d8b33aa51aed2a7fcb8ad5476a5d5ede8bb2a".
`sha-1-original: 354d8b33aa51aed2a7fcb8 ad5476a5d5ede8bb2a`	Matches all items with the sha-1 digest "354d8b33aa51aed2a7fcb8 ad5476a5d5ede8bb2a" when the item is loaded for the first time.
`sha-1-latest: 354d8b33aa51aed2a7fcb8 ad5476a5d5ede8bb2a`	Matches all items with the sha-1 digest "354d8b33aa51aed2a7fcb8ad5476a5d5ede8bb2a" after the item is reloaded.
`sha-256: 59836db73f6bc16f524050b1ca5b77f5 1305f63fe68b3918916b1b9cd4b8f347`	Matches all items with the sha-256 digest "59836db73f6bc16f524050b1ca5b77f51 305f63fe68b3918916b1b9cd4b8f347" .
`sha-256-original: 59836db73f6bc16f524050b1ca5b77f5 1305f63fe68b3918916b1b9cd4b8f347`	Matches all items with the sha-256 digest "59836db73f6bc16f524050b1ca5b77f5 1305f63fe68b3918916b1b9cd4b8f347" when the item is loaded for the first time.

`sha-256-latest: 59836db73f6bc16f524050b1ca5b77f5 1305f63fe68b3918916b1b9cd4b8f347`	Matches all items with the sha-256 digest "59836db73f6bc16f524050b1ca5b77f5 1305f63fe68b3918916b1b9cd4b8f347" after the item is reloaded.

mime-type

The mime-type field can be used to search on specific MIME types. This field is the more advanced alternative to the **kind** field and allows you to select more specific types of items in your query.

Examples

Query String	Results
`mime-type: message /rfc822`	Matches all RFC 822 email messages.
`mime-type: application/ vnd.ms- outlook-note`	Matches all Outlook email messages.
`mime-type: application/ vnd.ms- outlook*`	Matches all Outlook data items.
`mime-type: application/ vnd.ms-*`	Matches all Microsoft Office documents (and potentially documents from a few other Microsoft applications).
`mime-type: image/*`	Matches all images (some image types, however, may have different MIME types, for instance Adobe Illustrator does not fall into this category).
`mime-type:*/ vnd.ms-*`	Matches all files in Microsoft proprietary formats.

mime-type-tag

The mime-type-tag field can be used to search based on the kind of data they contain. This is similar to using the *kind* field, but item types can have multiple tags, whereas they can only have one kind.

Available tags are:

Tag	Explanation
`browser-bookmark-entry`	A browser bookmark entry.
`browser-cache-entry`	A browser cache entry.
`browser-download-entry`	A browser download entry.
`browser-history-entry`	A browser history entry.
`browser-shortcuts-entry`	A browser shortcut entry.
`chat-conversation`	A chat conversation.
`chat-message`	A single chat message.
`compressed-archive`	An item that is a compressed archive (ZIP, RAR, CAB, etc).
`database-row`	An item that is a row of a database.
`dxl-item`	A Lotus Notes DXL item.
`email-archive`	An item that is restricted to email archive examiner licences.
`email-exportable`	An item that is exportable to email formats (e.g. MSG, EML, PST, etc.)
`excel`	An item that is equivalent to an Excel spread sheet.
`expensive-retrieval`	Tags items whose children should be read once and cached or inlined to avoid expensive re-reads, e.g. to avoid downloading an item over the network.
`google-drive`	A Google Drive related item.
`immaterial-unless-loose`	An item that should always be considered immaterial unless it is a loose file.
`load-file`	A load file.

`load-file-item`	A load file item.
`log-entry`	An entry in a log file.
`logical-forensic-image`	A logical forensic image.
`loose-file-container`	An item that is not considered as being a loose file itself, but which contains loose files.
`mailbox`	An item that is a mailbox.
`mailbox-folder`	An item that can be treated as simple directory or branch.
`mapi-item`	A Microsoft Mail API (MAPI) item.
`mobile-forensic-backup`	A backup of a mobile device.
`mobile-forensic-image`	A forensic image of a mobile device.
`note`	A note taking note, e.g. Evernote note, OneNote note, etc.
`physical-disk-image`	A physical disk image.
`physical-forensic-container`	A physical forensic container.
`pdf-item`	An Adobe PDF item.
`realtime`	An item that is real time data.
`server-account`	An account on a server.
`server-folder`	A folder on a server.
`sqlite`	An SQLite database.
`task`	A task item.

`thumbnail-container`	A thumbnail container item.
`uses-inline-flag`	An item that has children that could be inline or not inline, depending on the inline flag.

Example

Query String	Results
`mime-type-tag:chat-message`	Matches all chat messages.

modifications

The modifications field can be used to search for items that have been modified since their initial ingestion.

Examples

Query String	Results
`modifications:text_updated`	Matches all data items which have had their text updated, for example, images that are processed using OCR.
`modifications:reloaded`	Matches all data items which have been reloaded since the initial load.

name

The name field can be used to search specifically on the file name of the item, or in the subject of email messages.

Examples

Query String	Results
`name:"Check this out"`	Matches items with the phrase "Check this out" somewhere in the file name, including email items with the phrase somewhere in the subject.

named-entities

The named-entities field can be used to search for items that contain a specified named entity. This field is only present for items that have been ingested with this option turned on in Evidence Processing section of Nuix.

Examples

Query String	Results
`named-entities:ip-address;*`	Matches all items that contain an IP address.
`named-entities:email;*`	Matches all items that contain an email address.

nlp

The nlp field can be used to search over all the nlp fields associated with every item of data. Each nlp field is a key-value pair where the key is in the following format:

```
model.modelName.[type.typeName].propertyName
```

The keys are case insensitive. The key always starts with the word "model" followed by the model name, then by zero or more pairs of type and name (always in pairs), and followed by the property name at the end.

The nlp property value of type text is case insensitive only on Elasticsearch cases, similar to *text-custom-metadata-exact*.

The wildcard character "*" can be used to match "any" type name. If partial wildcard matching is used for a property value (for example, "somePrefix*") then the application searches only property text fields.

Examples

Query String	Results
`nlp:model.defaultModel.dictionary.Business.indexed:true`	Matches items that have the NLP model "defaultModel" and dictionary "Business" with the "indexed" value equals true. Tip: Every node contains the indexed:true property and can be used to find nodes.
`nlp:model.defaultModel.language:English`	Matches items that have the NLP model "defaultModel" with the "language" property "English".
`nlp:model.defaultModel.language:Eng*`	Matches items that have the NLP model "defaultModel" with the "language" property matching "Eng*" (partial wildcard).
`nlp:model.defaultModel.dictionary.Business.proximity:0.4`	Matches items that have the NLP model "defaultModel" and dictionary "Business" containing a "proximity" value of 0.4.

`nlp:model.defaultModel.dictionary.Business.proximity:[0.3 to 0.4]`	Matches items that have the NLP model "defaultModel" and dictionary "Business" containing a "proximity" value between 0.3 and 0.4 inclusive.
`nlp:model.defaultModel.dictionary.Business.proximity:{0.3 to 0.4}`	Matches items that have the NLP model "defaultModel" and dictionary "Business" containing a "proximity" value between 0.3 and 0.4 exclusive.
`nlp:model.defaultModel.dictionary.*.indexed:true`	Matches items that have any NLP model "defaultModel" and any type of dictionary with an "indexed" value equals true.
`nlp:model.*.dictionary.*.proximity:0.12`	Matches items for any NLP model with any dictionary containing a "proximity" value of 0.12.
`nlp:model.defaultModel.risk.ClassifierRisk.riskType.DictionaryRisk.riskName.Banking.riskValue:[0 TO 1]`	Matches items with dictionary risk name 'Banking' and a 'riskValue' of 0 to1.
`nlp:"model.defaultModel.risk.entityRisk.riskType.it information.riskValue":[0 TO 1]`	Matches items with entity risk name 'IT Information' and a 'riskValue' of 0 to1. (Note: Names with spaces require ""quotation marks)
`nlp:model.defaultModel.risk.summaryRisk.MaxRisk:[0 TO 1]`	Matches items with summary risk name "MaxRisk" and a value of 0 to 1.

path-kind

The path-kind field can be used to match items where one of the ancestors of the item contains the specified kind of data. This is similar to using the path-mime-type field, but simpler to use. This can be useful if you know that the items you are searching for are inside a certain kind of data.

Example

Query String	Results
`path-kind:document`	Matches all items which are inside documents such as word processor documents and PDFs.

path-mime-type

The path-mime-type field can be used to match items where one of the ancestors of the item has the provided MIME type. This can be useful if you know that the items you are searching for are inside a certain kind of data.

Examples

Query String	Results
`path-mime-type:message/rfc822`	Matches all email attachments as well as files contained in those attachments. This also matches emails sent as attachments of another email.
`path-mime-type:application/vnd.ms-*`	Matches all items contained within Microsoft documents.

path-mime-type-tag

The path-mime-type-tag field can be used to match items where one of the ancestors of the item is of a type with the provided tag. This search field can be used when your search is inside a certain kind of data.

Example

Query String	Results
`path-kind:chat-conversation`	Matches all items which are inside chat conversations.

path-name

The path-name field can be used to match items where one of the ancestors of the item has the specified name. This can be useful if you know that the items you are searching for are inside items with a certain name.

Examples

Query String	Result
`path-name:" Documents and Settings/username"`	Loosely matches files underneath a specific Windows user's Documents and Settings directory.
`path-name:deleted.pst`	Loosely matches emails within a PST file called deleted.pst.
`path-name:doc`	Loosely matches items that were inside Word documents.

photo-dna

The photo-dna field searches on the PhotoDNA robust hash values of image items. PhotoDNA hashes are represented as base-64 strings. A search can accept an optional minimum score supplied as an integer between 0 and 100 inclusive. A range of acceptable scores can be supplied.

Wildcard characters can be used in queries that do not contain a threshold score.

Note: Since PhotoDNA hashes are long, for brevity we use the variable *hash* in some of the ensuing examples.

Examples

Query String	Results
`photo-dna:*`	Matches all image items that have a computed PhotoDNA hash value.
`photo-dna:abc*`	Matches all image items PhotoDNA hash beginning with "abc".
`photo-dna:"hash;75"`	Matches all image items that score at least 75 (out of 100) against the value *hash*.
`photo-dna:"hash;[75 TO 95]"`	Matches all image items that score between 75 and 95 (inclusive) against the value *hash*.
`photo-dna:"hash;{75 TO 95}"`	Matches all image items that score between 75 and 95 (exclusive) against the value *hash*.

print-method

The print-method field can be used to search for items that are stored as PDF, based on how the PDF was created.

This is a useful method to identify items that have been printed in a less than ideal fashion so that PDFs can be regenerated or custom PDFs can be substituted for these items.

Examples

Query String	Results
`print-method:printed`	Matches items that were properly printed to PDF.
`print-method: text_converted`	Matches items that had their text converted to PDF without formatting it.
`print-method: imported_by_user`	Matches items that had their PDF imported by a user, or a script run by a user.
`print-method: slip_sheet_by_user`	Matches items that were given a slipsheet by a user, or a script run by a user.
`print-method: slipsheet_unprintable`	Matches items that generated a slipsheet because they were unprintable either because the type was not supported or it was explicitly disabled through print options.
`print-method: slipsheet_encrypted`	Matches items that generated a slipsheet because they were encrypted.
`print-method: slipsheet_error_printing`	Matches items that generated a slipsheet because they failed to print due to an error occurring during printing.

printed-image-page-count

Matches items that have stored PDF and page count within the given range.

Examples

Query String	Results
`printed-image-page-count:[1 to 10]`	Matches items that have stored PDF and page count within the range from 1 to 10.

properties

The properties field can be used to search for property names and associated values for every item. Properties are stored as a text string, therefore, use quotes to correctly identify a specific field and value pair as a match.

Examples

Query String	Results
`properties: Bloggs`	Matches data items with the name Bloggs in any property name or property value.
`properties:" Author: Joe Bloggs"`	Matches data items that contain the value "Joe Bloggs" for the "Author" property. This also matches "Author Joe Bloggs" in the value because the colon character and other punctuation are ignored in the query.

shannon-entropy

The Shannon Entropy field is used to find data items that contain no data. It contains values between 0.0 and 8.0 where items with a value close to zero will have little or no data stored, and items with a value close to eight show high information content stored, relative to their length.

This can be every byte of the sample value, such as 00. It can also be used to search for JPEGs, ZIPs and other compressed data, and encrypted data, which will all have a high entropy value (tending to be closer to 8 than 0).

Examples

Query String	Results
`shannon-entropy:*`	Matches all items that have a computed Shannon Entropy value.
`shannon-entropy:0`	Matches all items with a Shannon Entropy value of zero.
`shannon-entropy:[5 TO 7.9]`	Matches all items with a Shannon Entropy value between 5 and 7.9 inclusive.

skintone

The skintone field can be used to search for image items with a skintone score within a specified range. This search only works on items where the skintone analysis option was selected in the Evidence Processing Settings during ingestion.

The skintone filter uses the following ranges.

Level	Lower Range	Upper Range
Severe	0.50	1.01
High	0.20	0.50
Medium	0.05	0.20
Low	0.00	0.05

Examples

Query String	Results
`skintone:[0.00 TO 0.05]`	Matches all data items with Low skin tone values.
`skintone:[0.20 TO 1.01]`	Matches all data items with Severe or High skin tone values.

text-custom-metadata

The text-custom-metadata field can be used to search over the text custom metadata fields associated with data items. The wildcard character "*" *can be used to match "any" in either field name or value. Partial wildcard matching (e.g. "somePrefix*") is supported for values but not field names.

Examples

Query String	Results
`text-custom-metadata:author:" Joe Bloggs"`	Matches data items that have the text custom metadata field "author" with the value "Joe Bloggs".
`text-custom-metadata:author:*`	Matches data items that have the text custom metadata field "author" with any value.
`text-custom-metadata:*:"Joe Bloggs"`	Matches data items that have any text custom metadata field with the value "Joe Bloggs".
`text-custom-metadata:*:Joe*`	Matches data items that have any text custom metadata field with a value starting with "Joe".

text-custom-metadata-exact

Same as *text-custom-metadata* except the search is faster and is performed case-sensitive over the field name and value.

top-level-item-date

The top-level-item-date field can be used to search for items by the item-date of their top-level item.

Note: This field will only be present on cases created with version 6.2 and above. Using it in earlier cases will return no search results. Items above the top-level will never be returned as matches.

Examples

Query String	Results
`top-level-item-date: 20070101`	Matches top-level items with an item-date of 1 January, 2007 and their associated family items.
`top-level-item-date: [20070101 TO 20070131]`	Matches top-level items with an item-date of January 2007 and their associated family items.

Annotation fields

Nuix Annotation fields contain information added by investigators or reviewers.

automatic-classifier | cluster | comment | custodian | document-id | exclusion | export-rules-applied | faces | face-confidence | has-comment | has-custodian | has-exclusion | has-item-set | has-print-preview | has-print-preview-markup | has-production-set | has-tag | is-item-original | item-set | item-set-batch | item-set-duplicates | item-set-originals | markup-set | number-valid | print-preview | production-set | production-set-guid | scored | score-confidence | tag

automatic-classifier

The automatic-classifier field can be used to search for items associated with the given automatic classifier. The parameters are classifier, classifier subset, and classification. They are separated by semicolons and are all optional. If a parameter is omitted it is treated as a "match all" filter.

Available options for classifier subset are training, automatically-classified and skipped. Additionally, auto can be used as an alias for automatically-classified.

Examples

Query String	Results
`automatic-classifier:"classifier1;training;relevant"`	Matches items which have been marked as relevant training items for the classifier "classifier1".
`automatic-classifier:"classifier1;automatically-classified"`	Matches all items which have been automatically classified by the classifier "classifier1" regardless of classification.
`automatic-classifier:"classifier1;;irrelevant"`	Matches all items which have been identified as irrelevant by the classifier "classifier1", that is, both training items and automatically classified items.
`automatic-classifier:";automatically-classified;relevant"`	Matches all items which have been automatically classified as relevant regardless of the classifier used.
`automatic-classifier:";auto;relevant"`	
`automatic-classifier:";;"`	Matches all items which form any part of any automatic classifier.

cluster

The cluster field can be used to search for items that have been clustered using up to three parameters. The parameters are cluster run name, cluster name and pivot status. They are separated by semicolons and are all optional. If a parameter is omitted, it is treated as a "match all" filter.

Examples

Query String	Results
`cluster: investigation23`	Matches items which have been clustered during the cluster run "investigation23".
`cluster:inv*`	Matches items which have been clustered during a cluster run beginning with "inv".
`cluster: investigation23; 568`	Matches items belonging to the 568th cluster in cluster run "investigation23".
`cluster: investigation\ 23; 568`	
`cluster:" investigation 23; 568"`	Matches items belonging to the 568th cluster in cluster run "investigation 23".
`cluster: investigation23; 568;pivot`	Matches the pivot item of the 568th cluster in cluster run "investigation23".
`cluster: investigation23;; pivot`	Matches all the pivot items in cluster run "investigation23".
`cluster:;;pivot`	Matches all the pivot items in any cluster run.
`cluster:*`	Matches all items in any cluster run.
`cluster:foobar; unclustered`	Matches items that did not cluster during cluster run "foobar" despite being eligible to exist in a cluster.
`cluster:foobar; ignored`	Matches items that were ignored during cluster run "foobar" because they were ineligible to exist in a cluster.

comment

The comment field can be used to search for text stored in the comments field by users. The text string used for searching is automatically treated as a match on characters rather than a whole word or, in other words, like a multi-character wild card has been used either side for the search term.

Example

Query String	Results
`comment:bank`	Matches items containing the word "bank" in the comments field, as well as items containing the words "banking" and "embank" in the comments field.

custodian

The custodian field can be used to search for items that have been assigned to a specific custodian.

Examples

Query String	Results
`custodian:"Bob"`	Finds all items that have been assigned to "Bob".
`custodian:"High*"`	Finds all items that have been assigned to a custodian with a reference name starting with "High".

document-id

The document-id field can be used to match all items that have the specified document id that was created by a production set.

Examples

Query String	Results
`document-id:DOC-00001`	Finds any items that have the document ID "DOC-00001".
`document-id:DOC-00001*`	Finds any items with a document ID that start with, or match, "DOC-00001", such as "DOC-000012232".
`document-id:[DOC-00001 TO DOC-00003]`	Finds any items with a document ID that have the prefix "DOC" and are numbered, ignoring padding, from 1 to 3.
`document-id:[DOC-1 TO DOC-4}`	Matches the same items as the query above, as padding is ignored and 4 is explicitly excluded with '}'.

exclusion

The exclusion field can be used to match items that have been excluded by the given exclusion name.

Examples

Query String	Results
`exclusion:"my exclusion"`	Finds all items that have been excluded under "my exclusion".
`exclusion:"irrelevant*"`	Finds all items that have been excluded with an exclusion name starting with "irrelevant".

export-rules-applied

The export-rules-applied field can be used to match all items assigned to any production set based on whether they have export rules applied.

Examples

Query String	Results
`export-rules-applied:true`	Finds all items assigned to any production set with export rules applied.
`export-rules-applied:false`	Finds all items assigned to any production set without export rules applied.
`export-rules-applied:true:5bca1b8cfcaa4896915f685d6a873eaf`	Finds all items assigned to the production set that has GUID 5bca1b8cfcaa4896915f685d6a873eaf with export rules applied.

faces

The faces field can be used to match items with a specified number of detected faces or a specified range of detected faces.

Examples

Query String	Results
`faces:4`	Matches items which have exactly 4 detected faces.
`faces:[3 TO 6]`	Matches items which at least 3 and no more than 6 detected faces.
`faces:[5 TO *]`	Matches items that have 5 or more detected faces.

face-confidence

The face-confidence field can be used to match items with detected faces using a single confidence value or a range of confidence values.

Note: This query will only work if face detection was selected during initial processing.

The detected face filter uses the following ranges.

Level	Lower Range	Upper Range
High	30	*
Medium	10	29
Low	1	9

Examples

Query String	Results
`face-confidence: 15`	Matches items that have at least one detected face possessing a confidence value of exactly 15.
`face-confidence: [20 TO 35]`	Matches items that have at least one detected face possessing a confidence value between 20 and 30 (inclusive).
`face-confidence: {20 TO 35}`	Matches items that have at least one detected face possessing a confidence value between 20 and 30 (exclusive).
`face-confidence: [40 TO *]`	Matches items that have at least one detected face possessing a confidence value greater than or equal to 40.
`face-confidence: [* TO 30]`	Matches items that have at least one detected face possessing a confidence value less than or equal to 30.
`face-confidence:*`	Matches items that have at least one detected face possessing any confidence value.

has-comment

The has-comment field searches for the existence or absence of any comments made by investigators.

Examples

Query String	Results
`has-comment:1`	Matches items that have an associated comment.
`has-comment:0`	Matches items that do not contain a comment.

has-custodian

The has-custodian field is a Boolean field that can be used to match all items that have been assigned to any custodians.

Examples

Query String	Results
`has-custodian:1`	Finds all items assigned to any custodian.
`has-custodian:0`	Finds all items not assigned to any custodian.

has-exclusion

The has-exclusion field is a Boolean field that can be used to match all items based on whether they have any exclusions.

Examples

Query String	Results
`has-exclusion:1`	Finds all items that have been assigned to an exclusion.
`has-exclusion:0`	Finds all items that have not been assigned to an exclusion.

has-item-set

The has-item-set field matches all items based on whether they are or are not assigned to any item set.

Examples

Query String	Results
`has-item-set:1`	Finds all items assigned to any item set.
`has-item-set:0`	Finds all items not assigned to any item set.

has-print-preview

The has-print-preview field matches all items that have a print preview as part of any production set.

Query String	Results
`has-print-preview:1`	Matches all items that have a print preview within any production set.
`has-print-preview:0`	Matches all items that do not have a print preview within any production set.

has-print-preview-markup

Matches all items based on whether they have a print preview as part of any production set.

Examples

Query String	Results
`has-print-preview:1`	Matches all items that have a print preview within any production set.
`has-print-preview:0`	Matches all items that do not have a print preview within any production set.

has-production-set

The has-production-set field is a Boolean field that can be used to match all items based on whether they are assigned to any production set.

Examples

Query String	Results
`has-production-set:1`	Finds all items assigned to any production set.
`has-production-set:0`	Finds all items not assigned to any production set.

has-tag

The has-tag field is a Boolean field that can be used to search for the presence of tags on data items.

Examples

Query String	Results
`has-tag:1`	Matches items that have been tagged.
`has-tag:0`	Matches items that are not tagged.

is-item-original

The is-item-original field is a Boolean field that can be used to match all items that are deemed to be originals after deduplication in any item set.

Examples

Query String	Results
`is-item-original:1`	Matches all items that were deemed originals in any item set.
`is-item-original:0`	Matches all items that were not deemed originals in any item set. This includes duplicates in all sets and items not in any item set.

item-set

The item-set field can be used to match items that are assigned to the specified item set in the named batch using up to three parameters separated by a semicolon.

Examples

Query String	Results
`item-set:"Item Set 1"`	Finds any items are assigned to the set "Item Set 1".
`item-set:"5bca1b8cfcaa4896915f685d6a873eaf"`	Finds any items are assigned to the set with the GUID 5bca1b8cfcaa4896915f685d6a873eaf.

item-set-batch

The item-set-batch field matches all original or duplicate items that are assigned to the specified item set in the named batch. This field takes two or three parameters separated by a semicolon:

- the set name or GUID,
- optionally "originals" for original items or "duplicates" for duplicate items, and
- the name of the batch load.

Examples

Query String	Results
`item-set-batch:"Item Set 1;load 1"`	Finds all items that are assigned to the set "Item Set 1" in the “load 1" batch.
`item-set-batch:"Item Set 1; originals;load 1"`	Finds all items that are assigned to the set "Item Set 1" in the “load 1" batch.
`item-set-batch:"Item Set 1; duplicates;load 1"`	Finds all duplicate items that are assigned to the set "Item Set 1" in the "load 1" batch.
`item-set-batch:" 5bca1b8cfcaa4896915f685d6a873eaf; load 1"`	Finds all items that are assigned to the set with GUID 5bca1b8cfcaa4896915f685d6a873eaf in the "load 1" batch.
`item-set-batch:" 5bca1b8cfcaa4896915f685d6a873eaf; originals;load 1"`	Finds all original items that are assigned to the item set with GUID 5bca1b8cfcaa4896915f685d6a873eaf in the "load 1" batch.
`item-set-batch:" 5bca1b8cfcaa4896915f685d6a873eaf; duplicates;load 1"`	Finds all duplicate items that are assigned to the set with GUID 5bca1b8cfcaa4896915f685d6a873eaf in the "load 1" batch.

item-set-duplicates

The item-set-duplicates field can be used to match all items that are assigned to the specified set as duplicates.

Examples

Query String	Results
`item-set-duplicates:"Item Set 1"`	Finds any items that are assigned to the set "Item Set 1" that were deemed to be duplicates.
`item-set-duplicates:" 5bca1b8cfcaa4896915f685d6a873eaf"`	Finds any items that are assigned to the set with GUID 5bca1b8cfcaa4896915f685d6a873eaf that were deemed to be duplicates.

item-set-originals

The item-set-originals field can be used to match all items that are assigned to the specified item set as originals.

Examples

Query String	Results
`item-set-originals:"Item Set 1"`	Finds any items that are assigned to the set "Item Set 1" that were deemed to be originals.
`item-set-originals:" 5bca1b8cfcaa4896915f685d6a873eaf"`	Finds any items that are assigned to the set with GUID 5bca1b8cfcaa4896915f685d6a873eaf that were deemed to be originals.

markup-set

The markup-set field can be used to search for items that have markups stored against them.

Examples

Query String	Results
`markup-set:first`	Matches items which have markups stored in markup set "first".
`markup-set:" general public"`	Matches items which have markups stored in a markup set with a name that includes spaces, in this case, "general public".

number-valid

The number-valid field matches all items assigned to any production set depending on whether the document numbering is valid.

Examples

Query String	Results
`number-valid:true`	Finds all items assigned to any production set with valid document numbering.
`number-valid:false`	Finds all items assigned to any production set with invalid document numbering.
`number-valid:true: 5bca1b8cfcaa4896915f685d6a873eaf`	Finds all items assigned to the production set that has GUID 5bca1b8cfcaa4896915f685d6a873eaf with valid document numbering.

print-preview

Matches all items that have a print preview within the production set GUID or name.

Examples

Query String	Results
`print-preview: 2b09b47c-efc8-11e0-a03f-8d0a4924019b`	Matches any items that have a print preview within the production set that has GUID "2b09b47c-efc8-11e0-a03f-8d0a4924019b".
`print-preview:" Production Set 1"`	Matches any items that have a print preview within the production set named "Production Set 1".

production-set

The production-set field can be used to match all items that are assigned to the specified production set.

Examples

Query String	Results
`production-set:" Production Set 1"`	Finds any items are assigned to the production set "Production Set 1".

production-set-guid

The production-set-guid field can be used to match all items assigned to the specified production set based on its GUID.

Examples

Query String	Results
`production-set-guid:2b09b47c-efc8-11e0-a03f-8d0a4924019b`	Finds any items are assigned to the production set that has GUID "2b09b47c-efc8-11e0-a03f-8d0a4924019b".

scored

The scored field can be used to search for “scored” or predicted items by an automatic classifier against a specific classification. This matches both training items and automatically classified items. This field can be used with two parameters: classifier and classification. Both parameters are optional and separated by a semicolon. The omitted parameter(s) is treated as a “match all” filter.

Examples

Query String	Results
`scored:”classifier1;relevant”`	Matches items that have been scored as relevant by the classifier "classifier1".
`scored:classifier1`	Matches all items which have been scored by the classifier "classifier1" regardless of classification.
`scored:;relevant`	Matches all items which have been scored as relevant regardless of the classifier used.

score-confidence

The score-confidence field can be used to match items that have a score confidence (or prediction confidence) created by an automatic classifier in the specified range. This matches both training items (if a model has been built), and automatically classified items.

The minimum confidence for an item is 0.5, therefore there will be no matches below this value. Score confidence does not discriminate by classifier name, so you would need to combine it with the automatic-classifier search field in your search string if you want to limit it to one classifier only.

Example

Query String	Results
`score-confidence:[0.5 TO 0.6]`	Matches all data items with a confidence score between 0.5 and 0.6 inclusive.

tag

The tag field can be used to search for a specific tag created by a user. If the tag name has any spaces, such as **Not Relevant**, it must be enclosed in double-quotes. You can also use a minus sign (-) in front of the tag field to exclude a tag from a search. All tags that exist as part of a nested structure, should be separated by the "|" when searching.

Examples

Query String	Results
`tag: Pornography`	Matches items that have been tagged with the tag named pornography.
`tag:"Not Relevant"`	Matches items that have been tagged with the tag named not relevant.
`tag:top*`	Matches items which have been tagged with a tag beginning with 'top', or any sub-tags of such tags.
`tag:"top|*"`	Matches items tagged under a top-level tag named “top”.

Communication fields

Communication fields are fields such as To, Cc, Bcc, and From, which are related to message type files such as emails. The data in these fields can be searched with the use of communication fields.

Note: The To, Cc, Bcc, and From fields in Nuix are extracted from the message transport headers, and therefore do not have a direct item level metadata property correlation.

---

address | to, cc, bcc | comm-date | from | recipient | recipient-count | sender

address

The address field can be used to search for any address contained within a communication which matches the pattern. The string used for searching will only match the complete email address and will not match just the personal part of the address.

It is case sensitive in versions below 6.0.4.

Example

Query String	Results
`address:user@hotmail.com`	Matches items inside messages sent to or received by user@hotmail.com. Note that this won't match user@hotmail.com.au or User@Hotmail.com

to, cc, bcc

The to, cc, and bcc fields can be used to search for items contained within a communication sent to an address that matches the pattern. The string used for searching may be a partial address.

The fields maintain a meaning that is the same as that used in email:

- **to**: direct recipient
- **cc**: carbon copy
- **bcc**: blind carbon copy (copied, but unknown to all recipients)

Examples

Query String	Results
`to:example`	Matches all messages sent to user@example.com, or example@anywhere.com.
`cc:"user@example.com"`	Matches all messages carbon copied to user@example.com, or addresses like another.user@example.com.
`to-mail-domain:example.com`	Matches items inside messages sent to the example.com domain. This search will only match this exact domain, so sub-domains will not be returned.
`to-mail-address:user@hotmail.com`	Matches items inside messages sent to user@hotmail.com.
`to:"user@example.com"`	Matches items inside messages sent to user@example.com, or another.user@example.com, or user@example.com.au.

comm-date

The comm-date field can be used to search for an item by the date of the communication which is contained in it. This field is only of practical use in conjunction with a range query.

Examples

Query String	Results
`comm-date:20070101`	Matches items that are inside a top-level communication which was sent on 1 January 2007.
`comm-date:[20070101 TO 20070131]`	Matches items that are inside a top-level communication which was sent on January 2007.

from

The from field can be used to search for items contained within a communication sent from an address that matches the pattern. The string used for searching may be a partial address.

Examples

Query String	Result
`from:example`	Matches all messages sent from user@example.com, or example@anywhere.com.
`from:"user@example.com"`	Matches all messages sent from user@example.com, or addresses like another.user@example.com.
`from-mail-address:"user@hotmail.com"`	Matches all messages sent from the user@hotmail.com address.
`from-mail-domain:"example.com"`	Matches all messages sent from the specific address. This search will only match this exact domain, so sub-domains will not be returned.
`from-address:INSTANT_MESSAGE;123@g.usa@WhatsApp`	Matches all WhatsApp instant messages sent from 123@g.usa@WhatsApp
`from-address:INSTANT_MESSAGE;*Brian*`	Matches all instant messages sent from Brian (matches "J. Brian", "Brian W." and other variants)
`from-address:"PHONE;+61 422 094 222"`	Matches messages (SMS) and calls sent from the phone number +61 422 094 222.
`from-domain:INSTANT_MESSAGE;skype`	Matches all skype instant messages
`from-domain:INSTANT_MESSAGE;WhatsApp`	Matches all WhatsApp instant messages.
`from-domain:INSTANT_MESSAGE;*`	Matches all instant messages.
`from-domain:PHONE;*`	Matches messages sent from a phone (SMS or call).
`from-address:*;user@example.org`	Matches messages sent from user@example.org regardless of address type (INSTANT_MESSAGE, PHONE or E-MAIL).

Note: The from field is a derived metadata field that is populated either from the transport headers or if not present, a combination of the PR_SENDER_EMAIL_ADDRESS / PR_SENDER_NAME.

recipient

The recipient field can be used to search for any recipient address (to/cc/bcc) contained within a communication that matches the pattern. The string used for searching will only match the complete email address.

It is case sensitive.

Examples

Query String	Results
`recipient: user@hotmail.com`	Matches items within messages sent to user@hotmail.com. Note that this won't match user@hotmail.com.au or User@Hotmail.com.
`recipient: *doe*`	Matches items within messages sent to john.doe@hotmail.com, john.doe@gmail.com, doe.jane@hotmail.com and so on.

recipient-count

The recipient-count field can be used to search for items containing recipients in the specified count range.

Note: It can also be specified within the Search, navigate to **Advanced** in the search bar, and select **Recipient Count** from the list for **Pick search criterion** and specify the criterion options.

Examples

Query String	Results
`recipient-count:[2 TO 5]`	Matches items that have recipient count between 2 and 5 (both inclusive).
`recipient-count:[10 TO *]`	Matches items that have at least 10 (inclusive) recipients.
`recipient-count:[* TO 10]`	Matches items that have at most 10 (inclusive) recipients.
`recipient-count:[3 TO 3]`	Matches items that have exactly 3 recipients. It is the same as recipient-count:3.

sender

The sender field can be used to search for an item that is contained inside a communication sent from a delegate party that matches the pattern. The string here may be a partial address.

Examples

Query String	Results
`sender: example`	Matches items inside messages sent from a delegate user@example.com, or example@anywhere.com.
`sender:"user@example.com"`	Matches items inside messages sent from a delegate user@example.com, or addresses like another.user@example.com, or user@example.com.au.

GUID fields

Nuix assigns a unique identifier to each item that it processes, which is the Globally Unique Identifier or GUID. The GUID is unique across all cases, and should not be confused with the digest identifier (MD5, SHA-1, SHA-256), as these are based on the item content and are designed to show authenticity and to find duplicate content.

Nuix GUIDs are created as 32-character strings that contain a mix of alpha-numeric characters and may or may not contain the hyphens: d05307e6-0402-4042-a75e-a6d24a1235b1.

Nuix also supports wildcards with all searches that use the GUID fields.

batch-load-guid | comm-guid | guid | parent-guid | path-guid

batch-load-guid

The batch-load-guid can be used to search for items that are contained, loaded or reloaded in a particular batch.

Example

Query String	Results
`batch-load-guid:5000`	Matches all items loaded in the batch with GUID 5000. When items are reloaded or new items are added to a case they will receive a new batch GUID.

comm-guid

The comm-guid field can be used to search for items that are contained within a communication with a given GUID. Therefore, the GUID provided to search must be a communication; that is, the query *has-communication:1* should show the item in question.

Example

Query String	Results
`comm-guid:5000`	Matches items contained as attachments of the communication with GUID 5000, or contained as attachments of another message that was in turn an attachment of GUID 5000 (and so on).

guid

The guid field can be used to search for an item with a specific GUID. By definition, this will always return either one or zero search results unless a wildcard was used with a partial guid.

Example

Query String	Results

`guid:5000`	Matches the item whose GUID is 5000.

parent-guid

The parent-guid field can be used to search for all child items that are embedded one level deep for a specific GUID. The parent-guid is different from the path-guid in that it searches for only embedded items directly under the guid specified, rather than all embedded items.

Examples

Query String	Results
`parent-guid:5000`	Matches all items whose parent GUID is 5000.
`parent-guid:empty`	Matches items without a parent.

path-guid

The path-guid field can be used to search for **all** of the child items for a specific GUID. This is useful when you need to search for all items associated with a piece of Nuix evidence.

The path-guid is different from the parent-guid in that it searches for all embedded items of the guid being searched, not just those one level deep.

Example

Query String	Results
`path-guid:5000`	Matches items whose parent's or any other ancestor's GUID is 5000.

List-based fields

List-based search fields allow you to leverage imported lists such as word, shingle or digest lists to find matches that contain any or all of the elements within the specified list.

For more information about importing and working with Word lists, Shingle lists and Digest lists, refer to the Nuix documentation.

Note: The name of the lists are case sensitive.

---

digest-list | fuzzy-hash-list | shingle-list | word-list

digest-list

The digest-list field can be used to search for items whose MD5 digest exactly matches a digest in the named list, effectively equivalent to using a digest list filter. Digest lists are frequently used to eliminate duplicate data from previously processed or reviewed evidence, typically done by using the NOT option in front of the digest-list field to ensure that only unique content is returned.

Examples

Query String	Results
`digest-list:”Known Software Files”`	Matches items whose hashes are present in the digest list named "Known Software Files".
`-digest-list:"Known Software Files"`	Matches items whose hashes are not present in the digest list named "Known Software Files".

fuzzy-hash-list

The fuzzy-hash-list field can be used to search for items matching the fuzzy hashes in the named list, effectively equivalent to using a fuzzy hash list filter. An optional score between 0 and 100 (inclusive) can be provided. Also, a range of acceptable scores can be supplied.

Examples

Query String	Results
`fuzzy-hash-list:" MyFuzzyHashes"`	Matches items whose fuzzy hash values are an exact match with the fuzzy hashes in "MyFuzzyHashes".
`fuzzy-hash-list:" MyFuzzyHashes; 75"`	Matches items whose fuzzy hash values score at least 75 against at least one fuzzy hash in "MyFuzzyHashes"

`fuzzy-hash-list:"MyFuzzyHashes;[75 TO 95]"`	Matches items whose fuzzy hash values score between 75 and 95 (inclusive) against at least one fuzzy hash in "MyFuzzyHashes".

shingle-list

The shingle-list field can be used to search for items having at least one shingle contained in the named shingle list, effectively equivalent to using a shingle list filter to find a similar document.

Examples

Query String	Results
`shingle-list:"MyFiles"`	Matches items with at least one shingle present in the shingle list named "MyFiles".
`shingle-list:"MyFiles;0.9"`	Matches items with at least one shingle present in the shingle list named "MyFiles", overriding the default resemblance threshold of 0.5 with 0.9.

word-list

The word-list field can be used to search for items containing any of the words or phrases in the given word list file, effectively equivalent to using a word list filter.

Example

Query String	Results
`word-list:Fraud`	Matches items whose words are present in the word list named "Fraud".

Sortable fields

Use Nuix sortable fields to search and sort the fields.

Note: The sort keys are used only on Scripting API. Refer to the search method of case interface in the Scripting API documentation.

family-content | family-names | family-properties | evidence-metadata | recipient-count | mime-type | family-stats | sender-address | sender-domain | has-binary | has-text | has-image | has-communication | has-embedded-data | audited-size | file-size | root-uri | position | path-position | named-entities | named-entity-types | modifications | item-id | item-date | top-level-item-date | faces | face-info | face-confidence | deep-learning-label | deep-learning-confidence | photo-dna | store-item-id | parent-item-store-id | guid | parent-guid | top-level-guid | comm-guid | digest-md5 | binary-properties | location-2d | other-item-link | other-item-link-root

family-content

The family-content field can be used to search over item contents associated with data items that were processed with this option in Evidence Processing Settings, and returns only the associated top-level item for any hits.

family-names

The family-names field is associated with data items that were processed with this option in Evidence Processing Settings and returns only the associated top-level item for any hits.

family-properties

The family-properties field can be used to search over the property names and values associated with data items that were processed with this option in Evidence Processing Settings and returns only the associated top-level item for any hits.

evidence-metadata

The evidence-metadata field can be used to search any custom metadata that the investigator added to the case when the evidence was loaded. In these examples, "site" is a piece of evidence metadata.

recipient-count

The id field can be used to search the ID of the data item.

mime-type

The mime-type field can be used to search on specific MIME types. This field is the more advanced alternative to the **kind** field and allows you to select more specific types of items in your query.

family-stats

The family-stats field is used to search the count and size of the family.

sender-address

The sender-address field name prefix is used to record delegated sender address information.

sender-domain

The sender-domain field name prefix is used to record delegated sender domain information.

has-binary

The has-binary field is a Boolean field that can be used to search for items that could have binary data. Items such as files and email messages have binary data while few types of items such as file folders /directories, mail folders or folders inside compressed zip files lack binary data.

has-text

The has-text field is a Boolean field that can be used to search for items that could contain text data. This type of search does not imply the document has text, but rather just that the item type could contain text.

has-image

The has-image field is a Boolean field that can be used to search for items that contain image data.

has-communication

The has-communication field is a Boolean field that can be used to search for items that have communication-based data. This search matches items that are communication items, such as email or instant messages, but not the items that are attached to, or associated with them. To search for attachments, see the *Communication Fields* category.

has-embedded-data

The has-embedded-data field is a Boolean field that can be used to search for items that could contain embedded data. Using this search matches items that have the ability to contain embedded data. For instance, it will match all directories even if a directory does not contain any file.

Note: This search will retrieve only the items that have the ability to contain embedded items, **not** the embedded items.

---

audited-size

The audited-size field is used to search for audited items of a particular size (in bytes). Items that have been marked for auditing (non-immaterial items) can be identified with the search flag:audited.

file-size

The file-size field name is used to record file sizes, for use with the field cache.

root-uri

The root-uri field is used to list the root URI of this data item, if known.

position

The position field is used to store the position of the data item in relation to its siblings.

path-position

The path-position is used to store the list of data position values from the root to the parent data item, separated by newlines.

named-entities

The named-entities field can be used to search for items that contain a specified named entity. This field is only present for items that have been ingested with this option turned on in the Evidence Processing section of Nuix.

named-entity-types

The named-entities-types field is used to list named entity types found within the document.

modifications

The modifications field can be used to search for items that have been modified since their initial ingestion.

item-id

The item-id field can be used to search for a short ID that is unique for the case.

item-date

The item-date field can be used to search for an item by the date of the item. The date of the item will generally be the communication date for items which are a communication item or the modified date for other types of item. Items without a date inherit the date of their parent.

This field is most useful when used in conjunction with a range query.

Note: This field will only be present on cases created with Nuix version 3.0 and above. Using it in earlier cases will not return any search result.

---

top-level-item-date

The top-level-item-date field can be used to search for items by the item-date of their top-level item.

Note: This field will only be present on cases created with version 6.2 and above. Using it in earlier cases will return no search results. Items above the top-level will never be returned as matches.

---

faces

The faces field can be used to match items with a specified number of detected faces or a specified range of detected faces.

face-info

The face-info field is used to store the detected face information.

face-confidence

The face-confidence field can be used to match items with detected faces using a single confidence value or a range of confidence values.

Note: This query will only work if face detection was selected during initial processing.

The detected face filter uses the following ranges.

Level	Lower Range	Upper Range
High	30	*
Medium	10	29
Low	1	9

deep-learning-label

The deep-learning-label field is used to store the label resulting from Deep Learning model evaluation.

deep-learning-confidence

The deep-learning-confidence field is used to store the confidence value for a Deep Learning label.

photo-dna

The photo-dna field searches on the PhotoDNA robust hash values of image items. PhotoDNA hashes are represented as base-64 strings. A search can accept an optional minimum score supplied as an integer between 0 and 100 inclusive. A range of acceptable scores can be supplied.

Wildcard characters can be used in queries that do not contain a threshold score.

Note: Since PhotoDNA hashes are long, for brevity we use the variable *hash* in some of the ensuing examples.

---

store-item-id

The store-item-id field is used to store the pseudo-docID, unique within the store but not prone to being reordered.

parent-item-store-id

The parent-item-store-id field is used to store the reference to the parent document's ES store ID.

guid

The guid field is used to search for an item with a specific GUID. By definition, this will always return either one or zero search results unless a wildcard was used with a partial guid.

parent-guid

The parent-guid field is used to search for all child items that are embedded one level deep for a specific GUID. The parent-guid is different from the path-guid in that it searches for only embedded items directly under the guid specified, rather than all embedded items.

top-level-guid

The top-level-guid field is used to store the guid for the top-level ancestor of the item. An item that has a GUID the same as its top-level-item GUID is a top-level item.

comm-guid

The comm-guid field can be used to search for items that are contained within a communication with a given GUID. Therefore, the GUID provided to search must be a communication; that is, the query *has-communication:1* should show the item in question.

digest-md5

The digest-md5 field contains MD5 digest in lowercase hexadecimal form. This is used as performance optimization of sorts, you can also perform this using the multi-value "digest" now.

binary-properties

The binary-properties field contains the map of properties for all binary properties.

location-2d

The location-2d field matches items which have Latitude and Longitude properties within the specified geographical perimeter. This field only supports `geodistance` queries in the form below. The first 2 parameters are latitude and longitude (in degrees) followed by distance in kilometers.

other-item-link

The other-item-link field is used to link to another data item, represented as an ID path.

other-item-link-root

The other-item-link-root field is used to set the path offset value that identifies a root item that is common to linked items.