BLNXREST GETTING STARTED GUIDE

BlackLynx command line interface to the RESTful server

Version 1.1

ABSTRACT

Use the BlackLynx command line to search your data fo your business analytics, business intelligence, or business visualization applications without the need for indexing nor ETL

July 3, 2019

The `blnxrest` command line tool is a simple bash script with syntax that is very similar to the native BlackLynx search tools and uses the `BlackLynx REST server` as a backend. The install package places the `blnxrest` tool in the "/usr/bin" directory. With the `blnxrest` tool, you can:

- Combine different types of primitives on one command to create complex queries. For example, you can combine fuzzy edit search with a date range and a time range in a single command.
- Run in cluster or single server mode.
- Run on remote/distributed servers.
- Change output to JSON, on the fly.

Table of Contents

BlackLynx REST Parameters

The following table details the parameters available to customize queries to the BlackLynx search tools:

Parameter	Type	Description
Search Specific Parameters		
`--backend=<x>`	String	Selects specific BlackLynx backend processor
`--backend-mode=<mode>`	String	Used to select the backend mode defined in RESTful server configuration file.
`--backend-option=<opt>`	String	Options passed through to the selected backend tool.
`-c <path>` `--catalog=<path>`	String	Specify an input catalog name.
`-d <D>` `--fuzziness=<D>`	Integer	Specifies the search distance Default is 0. Used with fuzzy hamming and edit distance.
`-e <E>` `--delimiter=<E>`	String	Specify the delimiter used between result records.
`-f <path>` `--file=<path>`	String Required	Specify an input filename(s). More than one -f <filename> parameter can be used to specify multiple filenames. Wild carded filenames should be enclosed in quotes (“)
`-h` `--help`		Print the help message and detailed syntax.
`-i`	String	Specifies case-insensitive analysis for supported primitives. Default is case-sensitive.
`--lifetime=<t>`	String	Automatically cleanup results files after time expires (ex. 1h, 10m, 20s)
`--line`		Specify the surrounding whole line (same as `-w=line`)
`-n` `--nodes=<N>`	Integer	Specify 1-4 RCAB processing nodes to use. Default uses all available nodes. (Only used on legacy RyftONE hardware)
`--no-reduce`		Do not reduce the duplicates for FEDS.
`-od <file>` `--data=<file>`	String	Specify a data results file.
`-oi <file>` `--index=<file>`	String	Specify an index results file.
`-ov <file>` `--view=<file>`	String	Specify to retain the view results files which speeds up the /search/show endpoint.
`-p` `--mode=<mode>`	String	Specifies the search mode to run, which can be one of the following. This parameter is optional, with the exception of a pcap search. • g – generic search. When used the search mode is taken from the search expression. `Not applicable to pcap.` • `exact_search` (or `es`); DEFAULT • `fuzzy_hamming_search` (or `fhs`)

Parameter	Type	Description
		• `fuzzy_edit_distance_search` (or `feds`), • `date_search` (or `ds`), • `time_search` (or `ts`) • `numeric_search` (or `ns`), also used for currency • `ipv4_search` (or `ipv4`) • `ipv6_search` (or `ipv6`) • pcre2 • pcap (has to be specified for pcap) • pip
`-r` `--reduce`		Reduce the duplicates for FEDS (by default).

`-s <query>`
`-q <query>`
`--query=<query>`
	String	Specify the search/query expression to use with *_search primitives.

`-w <width>` `--width=<W>`	Integer	Specify the surrounding width. Default is `width=0`. Used with unstructured *_search primitives. Can use `-w=line` return the entire matching line.
REST Specific Parameters		
`-a <addr>` `--address=<addr>`	String	Specifies the BlackLynx RESTful server address. Default is `http://localhost:8765`.
`--accept=<fmt>`	String	Accept format can be “json” or “csv”.
`-b <json>` `--body <json>`	String	Additional request body in JSON format.
`--cluster`		Specify a cluster search. Opposite to `--local` which is the default for search.
`--count`		The `/count` endpoint; print just statistics. Default. Use in place of `--search`.
`--details`		Reformat output to add Data Rate and details on intermediate results.
`--drate`		Reformat output to add Data Rate.
`--dry` `--dry-run`		Do a “dry-run” of the command. Does not invoke the primitive.
`--fields=<list>`		Specify comma-separated list of fields to return. Useful with XML and JSON formats.
`--format=<fmt>`	String	Specify format of the result records. Available options are: • `raw – base-64 encoded data. Default.` • `xml` - input file set contains XML data, the found records could be decoded by using `format=xml` query parameter. Records will then be translated from XML to JSON. • `json` - input file set contains JSON data, use the `format=json` query parameter to decode data to JSON

Parameter	Type	Description
		• csv – decode CSV records (column names customized via tweaks) • `utf8` - to get human readable data (instead of base-64 encoded bytes). This parameter asks `RESTfull service` to interpret found bytes as UTF-8 string instead of base-64 encoded raw bytes • `null/none` – ignores all data **Note**: format and fields parameters only work on data being written to sdtout.
`--limit=N`	Integer	Specify the limit on total number of records printed (use with `/search`). Default is no limit.
`--line`		Used with *_search primitives to specify that returned results should include the encompassing line. Similar to the line behavior of standard Linux grep.
`--local`		Specify a local search. Opposite to `--cluster`. Default.
`--no-stats`		Disable statistics output.
`--performance`		Add performance metrics to the output statistics.
`--search`		The `/search` endpoint (used by default); print all found items.
`--share-mode`		Sharing mode. Options are “ignore”, “skip” or “wait-10s”.
`--stream`		Use stream output format. Provides a sequence of JSON "tag-object" pairs to be able to decode input data on the fly (used for node communication within cluster).
`--transform=<tx>`		Specifies one of three custom post-process transformations: • `match(“expr”)` – regexp to match “expr” • `replace(“expr”, “template”)` – regexp to replace • `script(“name”)` – call an external script Several transformations can be specified in one call, creating a chain transformation.

`-u <creds>`
`--user<creds>`
`--auth`
	String	Specify user credentials, “<username>:<password>”.

`-v | --verbose`		Tell curl to be verbose.
`-vv | --pretty`		Specify properly indented formatting with jq tool. Cannot be used with the `--search` parameter.

Examples

The following examples include both structured and unstructured datasets and are noted with (S) or (U), respectively.

Fuzzy Hamming Search	
S	`blnxrest -q '(RECORD.block CONTAINS FHS("TRUMBOWL",CS=true,DIST=2))' -f ODBC/Chicago_Crime/chicago.pcrime -n 4 --local --count -od tt2-1.pcrime`
U	`blnxrest -s '(RAW_TEXT CONTAINS FHS("Babe Ruth",CS=true,DIST=1,WIDTH=20))' -f regression/wikipedia-20150518.bin -n 4 -vv --local --count -od br1.txt`
Fuzzy Edit Distance Search	
S	`blnxrest -q '(RECORD.block CONTAINS FEDS("TRUMBOWL",CS=true,DIST=2))' -f ODBC/Chicago_Crime/chicago.pcrime -n 4 -vv --local --count -od tt2-1.pcrime`
U	`blnxrest -s '(RAW_TEXT CONTAINS FEDS("Babe Ruth",CS=true,DIST=1,WIDTH=20))' -f regression/wikipedia-20150518.bin -n 4 -vv --local --count -od br1.txt`
Date Search	
S	`blnxrest -q '(RECORD.date CONTAINS DATE(MM/DD/YYYY = 04/14/2015))' -f ODBC/Chicago_Crime/chicago.pcrime --local --count -od tt4-1.pcrime -n 4`
U	`blnxrest -s '(RAW_TEXT CONTAINS DATE(04-01-1927 < MM-DD-YYYY < 09-30-1927))' -f regression/wikipedia-20150518.bin --local --count -w 50 -od tu5-1.txt -n 4`
Time Search	
S	`blnxrest -q '(RECORD.date CONTAINS TIME(HH:MM:SS = 11:53:00))' -f ODBC/Chicago_Crime/chicago.pcrime --local --count -od tt4-3.pcrime -oi itt4-3.txt -n 4`
U	`blnxrest -s '(RAW_TEXT CONTAINS TIME(11:00:00 <= HH:MM:SS < 11:53:00))' -f regression/wikipedia-20150518.bin --local --count -w 50 -od tu5-2.txt -n 4`
Currency Search	
S	`blnxrest -q '(RECORD.fare_amount CONTAINS CURRENCY(CUR > "$450.00", "$", ",", "."))' -f ODBC/NYC_Taxi/122015.NYC_Taxi --local --count -od lb.txt -e $'\n-\n'`
U	`blnxrest -q '(RAW_TEXT CONTAINS CURRENCY(CUR > "$450.00", "$", ",", "."))' -f ODBC/NYC_Taxi/122015.NYC_Taxi --local --count -od lb.txt -e $'\n-\n'`
Numeric Search	
S	`blnxrest -q '(RECORD.lat CONTAINS NUMBER("41.85" <= NUM <= "41.86",",",".")) AND (RECORD.lon CONTAINS NUMBER("-87.75" <= NUM <= "-87.68",",",."))' -f ODBC/Chicago_Crime/chicago.pcrime --local --count --format=xml -od tt4-5.pcrime -oi itt4-6.txt -n 4`
U	`blnxrest -q '(RAW_TEXT CONTAINS NUMBER("310-555-1000" <= NUM < "310-555-3000", "-", "."))' -f regression/passengers.txt -w 30 -od lb.txt -e $'\n-\n'`
IPv4 Search	
S	`blnxrest --local --count -f ODBC/Internet_Service/internet_service.isxml -s '(RECORD.ip CONTAINS IPV4("10.72.0.0" <= IP < "10.76.255.255"))' -od tt4-8.isxml -oi itt4-7.txt -n 4`
U	`blnxrest -f regression/passengers-ipv4.txt -q '(RAW_TEXT CONTAINS IPV4("10.0.0.0" <= IP < "11.0.0.0"))' -od lb.txt -e $'\n-\n'`
IPv6 Search	

S	`blnxrest --local --count -f ODBC/Internet_Service/internet_service.isxml -s '(RECORD.ip CONTAINS IPV6("::ffff" <= IP < "::ffff:200.255.255.255"))' -od tt4-8.isxml -oi itt4-9.txt -n 4`
U	`blnxrest -f regression/wiki*.bin -q '(RAW_TEXT CONTAINS IPV6("2000::" <= IP <= "2001:0470:1f06:514::0022"))' -od lb.bin -e $'\n-\n'`
PCRE2 Search	
S	`blnxrest -vv -f ODBC/CallDetailRecs/cdr.csv -q '(RECORD.31 CONTAINS PCRE2("(886682|886271)\d{6}",FIELD_DELIMITER=","))' -od results.csv`
U	`blnxrest -vv -f "splunk_40*/*" -q '(RAW_TEXT CONTAINS PCRE2("client=(?!127\.\d+\.\d+\.\d+)(\d+\.\d+\.\d+\.\d+) latency",LINE="true"))' -od results.txt`
PIP Search (only available for structured files)	
S	`blnxrest -vv -f "blgeo_query_od_1-1.txt" -q '(RECORD.coordinates.coordinates CONTAINS PIP(VERTEX_FILE="/ryftone/miscTestFiles/polygons/usa/Texas.txt", VERTEX_FILE_IS_FILELIST="true"))' -od "blgeo_query_od_1-2.txt" -oi "blgeo_query_oi_1-2.txt"`
PCAP Search (combined OSI level 2 – 4 search with BlackLynx primitive search on payload)	
S	`blnxrest -vv -p pcap -f PCAP/newData/*.pcap -q 'tcp.port != 443 and (RECORD.payload CONTAINS PCRE2("[^-0-9]*\d{3}-\d{2}-\d{4}[^-0-9]*"))' -od results.pcap`
Complex Search	
S	`blnxrest --local --count -q '(RECORD.date CONTAINS DATE(04/14/2015 < MM/DD/YYYY < 04/16/2015)) AND (RECORD.date CONTAINS TIME(08:30:00 <= HH:MM:SS <= 09:30:00)) AND (RECORD.arrest EQUALS "true")' -w 5 -f ODBC/Chicago_Crime/chicago.pcrime -od tt5-2.pcrime -oi itt5-2.txt -n 4`
U	`blnxrest --local --count -f regression/wikipedia-20150518.bin -s '(RAW_TEXT CONTAINS FHS("Babe Ruth",CS=true,DIST=2,WIDTH=100)) AND (RAW_TEXT CONTAINS NUMBER(NUM = "1927",",",","."))' -od tu4-2.bin -n 4`

Query Optimization Rules

These optimization rules are followed when executing `blnxrest` queries, with a few exceptions:

FEDS sub-queries are typically not combined for RECORD or RAW_TEXT searches. This is controlled by a search type based exception list configured in the RESTful server configuration file, using the `optimizer-do-not-combine` option, like this:

```
optimizer-do-not-combine: feds, pcre2 # coma-separated search modes that should not be combined
```

For RECORD-based sub-queries:

Combine all RECORD-based sub-queries. For example:

```
(RECORD.text CONTAINS FHS("A",d=1)) AND (RECORD.id CONTAINS NUMBER(NUM>100))
```

For RAW_TEXT-based sub-queries:

Combine all RAW_TEXT-based sub-queries that use the "OR" logical operator into one command call:

```
(RAW_TEXT CONTAINS "Apple") OR (RAW_TEXT CONTAINS "Orange")
```

EXCEPTION: Does NOT combine any RAW_TEXT-based sub-queries that use the "AND" logical operator; they can never be combined into one command call. Instead, program issues the queries in a serial mode with each subsequent search performed on the output of the previous search(es).

Precedence of Operations

Use parenthesis "( )", curly braces "{ }" and square brackets "[ ]" to control the precedence (order of) operations.

Parenthesis "( )"

Use additional whitespaces to make it more legible and simplify comprehension. For example:

```
((RECORD.city EQUALS EXACT("Rockville")) OR (RECORD.city EQUALS EXACT("Gaithersburg"))) AND
(RECORD.state EQUALS EXACT("MD"))
```

Curly Braces "{ }"

Used to control query optimization rules and for manual optimization. All queries in the curly braces are combined into just one primitive call. Here are two simple examples:

```
{Hello} OR {Apple AND Orange}
```

The service then splits it into two calls:

```
(RAW_TEXT CONTAINS "Hello")

(RAW_TEXT CONTAINS "Apple") AND (RAW_TEXT CONTAINS "Orange")
```

The curly braces make two exceptions to the query optimization rules:

"Apple" and "Orange" subqueries are combined into one primitive call, even though the optimization rule says that RAW_TEXT and AND should not be combined.

"Hello" is not combined even though the optimization rule says that RAW_TEXT and OR should be combined.

Square Brackets "[ ]"

Square brackets are like parenthesis, with one major difference.

Let's say you have this query:

```
"(Hello) AND (Apple)"
```

Generally, two primitive calls are generated. The first call looks for "Hello" and saves the results to a temporary file. Then the second call looks for "Apple" in that temporary results file.

Let's modify it and add square brackets around Hello, like this:

```
"[Hello] AND (Apple)"
```

This query works in almost the same way, but with one exception. Once the first call for "Hello" is executed, the subsequent search for "Apple" is performed on the list of files contained in the INDEX file of the first search (not on the data output file).

This feature is useful to perform nested or subsequent searches on a large group of files or on a properly created catalog. The file names inside a catalog should be relative to user's home directory, otherwise no files will be found for the second primitive call.

For more information, see the *BlackLynx REST Command Reference Guide*.