Ryft REST Command

User & Admin Guide

Ryft REST Release Number: 0.11.0

Ryft Document Number: 1191

Document Version: 1.2.0

Revision Date: June 2017

Revision History

Date	Reason for Change	Version
June 2017	Update configuration file Update /FILES endpoint Add catalog feature Add curly braces and square brackets for precedence order of operation Add transform feature Add additional information for each primitive	1.2.0
November 2016	Update API endpoints and examples	1.1.0
October 2016	Initial Release	1.0.0

Contents

Using This Guide

This guide is for the user who will use the Ryft ONE REST API to access the Ryft ONE and use its primitives.

Additional resources: Ryft Open API Library User Guide and the *Ryft ONE: User Guide*.

Ryft Technical Support

You may access our first-tier support directly from our public website by using the Chat Widget and starting a chat with our support agents. All chat conversations are tracked and become help desk tickets in our support system. In addition, you can access our support system through the Ryft Support site: https://support.ryft.com. Log in with your credentials to see your past tickets, create new tickets, and to access limited-access content.

For technical support, contact us:

Email: support@ryft.com

Web: https://support.ryft.com

Phone: 1-855-793-8663 (RYFT ONE)

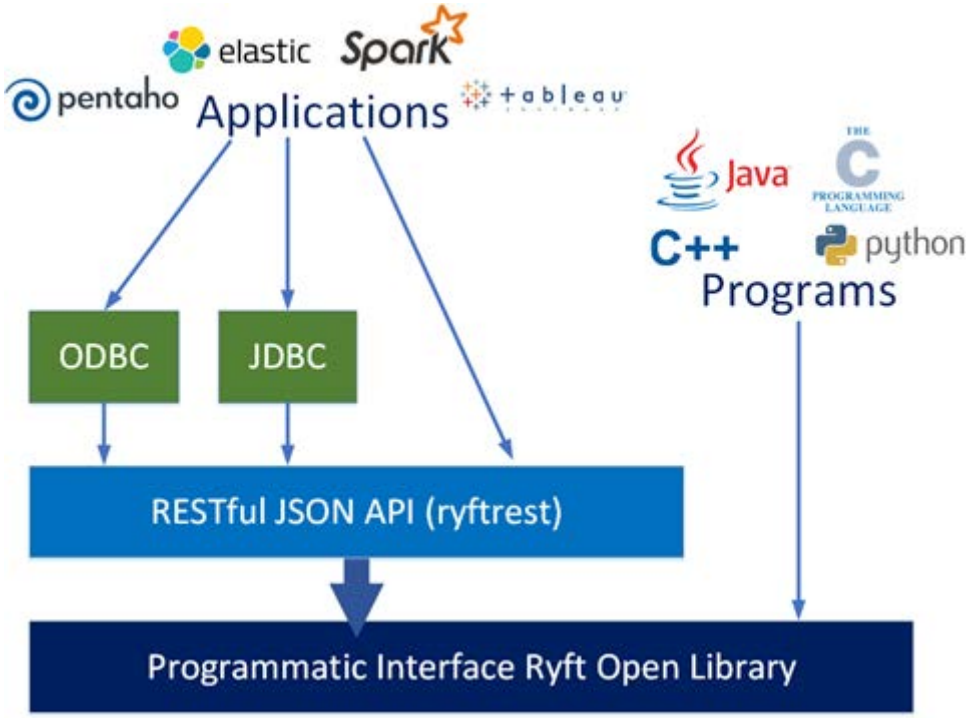
About Ryft Systems Inc.

Ryft makes data analytics fast and simple with the world's top cloud and physical heterogeneous compute accelerators. With more than a decade of experience, Ryft helps deliver instant insights into any data by eliminating the latency associated with data indexing and transformation/curating. Today, industry leaders rely on Ryft to quickly and simply unlock the value hidden in their data for real-time actionable insight.

1

1: Overview

The Ryft RESTful JSON server (`ryft-server`) runs as a daemon and provides access to the Ryft hardware via the `libryftone` library or the `ryftprim` command line tool. It is written in Go using Gin HTTP framework. The server contains a search engine abstraction which unifies access to the RyftONE primitive library and is a natural fit to operating in a Ryft Cluster.



As this diagram shows, the Ryft RESTful JSON API is the interface between the Ryft Open Library (ROL), and the Ryft ODBC/JDBC connectors, and to applications like Spark. You can use `ryftrest` on the command line, using a graphical user interface like the Ryft Web UI or the Swagger UI, via integration with Elasticsearch, or with one of the Ryft connectors (ODBC, JDBC or Spark).

The RESTful JSON API and the `ryftrest` command (used on the command line interface) provides a rich and flexible set of tools, commands and interfaces that enable you to perform data analytics operations that leverage these built-in features:

- Exact Search

- Fuzzy Hamming Search
- Fuzzy Edit Distance (Levenshtein) Search
- Date and Time Search
- Numbers and Currency Search, and
- IPv4 and IPv6 Search

Each of these operations are covered in this guide.

The `ryftrest` command is a simple bash script that uses the ryft-server as a backend. It can run complex queries and sub-queries, and even combine different types of operations such as fuzzy edit combined with date and time search. It also supports both cluster mode and single server mode, and runs on local, remote or distributed Ryft servers, whether on site, in the cloud, or in a hybrid environment. The `ryftrest` results and output are in JSON format.

For more information, refer to the following documents stored on your Ryft ONE server ("/ryftone/documentation") or on the documentation CD included with your Ryft ONE server.

- *Ryft ONE: User Guide*
- *Ryft ONE Connector: ODBC Installation and User Guide*
- *Ryft ONE Connector: JDBC Installation and User Guide*
- *Ryft ONE ODBC/JDBC Connector: SQL Reference Guide*
- *Ryft ONE Spark Integration and User Guide*
- *Ryft ONE REST API-Swagger.io Reference*

Search Engine Interface

The search engine interface looks like this:

```
type Engine interface {
    Search(cfg *Config) (*Result, error)
    Count(cfg *Config) (*Result, error)
    Files(dir string) (*DirInfo, error)

    Options() map[string]interface{}
}
```

Most of the interface methods are related to the corresponding REST API endpoints: `/search`, `/count` and `/files`. The `Options()` method is used to get search engine's internal options which are, initially, customizable via the search configuration file.

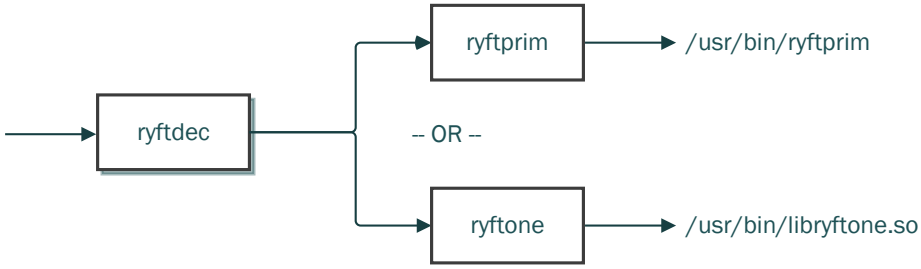
Search Engine Implementations

The `ryft-server` uses this abstract search engine in its main code so that we can easily change the actual search engine implementation. For example, for test purposes it's quite easy to create a fake search engine which uses a simple `grep` tool on the local filesystem.

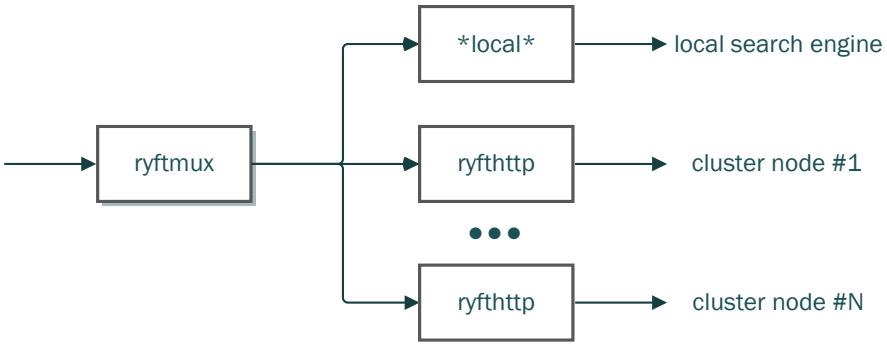
This is a list of search engine implementations. Each is covered in the following pages:

- ryftprim uses the `ryftprim` command line tool
- ryftone uses the `libryftone` library from Ryft Open API (future implementation)
- ryfthttp uses another `ryft-server` instance
- ryftmux multiplexes results from several search engines
- ryftdec decomposes complex search queries

The `ryftprim` and `ryftone` search engines are used for local searches. The `ryftdec` stays ahead and translates complex search queries into several calls to the backend, like this:



In cluster mode, we use a set of `ryfthttp` search engines to access remote Ryft servers on the cluster, and `ryftmux` to multiplex all the results received:



`ryftprim` Search Engine

The `ryftprim` search engine uses the `ryftprim` command line tool to access the Ryft ONE server. Internally, the `ryftprim` tool uses `libryftone`, but to ensure thread-safe limitations of `libryftone` a separate process is spawned for each search call. Implementation sends the corresponding search command via the `ryftprim` tool, and then parses the generated index and data files.

`ryftprim` Options

The `ryftprim` search engine supports the following options (these can be customized via search configuration file):

Option	Description
instance-name	The name of the search engine instance. Used to distinguish different instances.
ryftprim-exec	The path to the `ryftprim` tool. Default is "/usr/bin/ryftprim."
ryftprim-legacy	Use to obtain machine-readable statistics. Default is true.
ryftone-mount	The main volume on the Ryft ONE. Default is "/ryftone."
open-poll	The open file poll timeout. The search engine continually attempts to open the index or data file within this timeout period. Default is 50ms.
read-poll	The read file poll timeout. The search engine will continually attempt to read the index or data file within this timeout period. Default is 50ms.
read-limit	The limit of read attempts. Default is 100. After the 100 fail read attempts, the search engine stops reading and returns an error.
keep-files	Keeps the intermediate data and index files in order to implement the `--keep` option on the server.
index-host	The node name of the cluster. The search engine marks all found record indexes with the name of the cluster node to distinguish where the record was found.

NOTE: The working directory for the `ryftprim` search engine is "$ryftone-mount/$instance-name."

`ryftone` Search Engine

The `ryftone` search engine uses `libryftone` library to access Ryft hardware (See Ryft Open API)

Implementation is very similar to `ryftprim` search engine. It also sends the corresponding search command to `libryftone` library and then parses the generated index and data files.

NOTE: This is for future implementation

`ryftone` Options

The `ryftone` search engine supports the same options as `ryftprim`, above, except for `ryftprim-exec`.

`ryfthttp` Search Engine

The `ryfthttp` search engine uses another `ryft-server` instance to access Ryft hardware. This search engine is used in cluster mode to forward search queries to remote Ryft boxes.

`ryfthttp` Options

The `ryfthttp` search engine supports the following options:

Option	Description
server-url	Remote `ryft-server` address including host name and port. Default is "http://localhost:8765".
local-only	Flag to use local search on remote Ryft box. Related to `local=` server's query parameter
skip-stat	Flag to skip statistics. Related to `stats=` server's query parameter
index-host	Cluster node name. Search engine marks all found record indexes with cluster node's name. This feature lets you distinguish where the record come from. Usually, this option is not applied because all found indexes should be already marked by the local search engine (`ryftprim` or `ryftone`).

Usually these options are automatically filled by the `ryft-server` when the cluster configuration is built. It is also possible to customize these via search configuration file.

`ryftmux` Search Engine

The `ryftmux` search engine multiplexes results from several search engines, such as from one `ryftprim` and a few `ryfthttp` from cluster's nodes. This search engine is configured and used internally by the `ryft-server`.

There are no options for `ryftmux`.

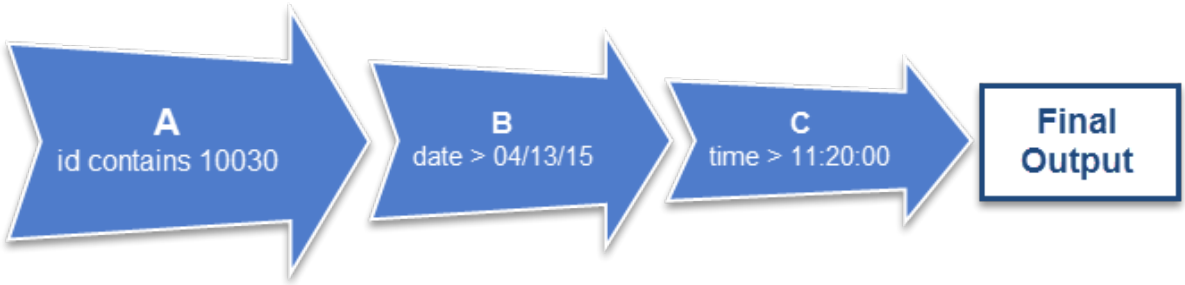
`ryftdec` Search Engine

The `ryftdec` search engine behaves similarly to a filter. It uses the `ryftprim` search engine instance in the background for query execution. The main purpose is to decompose complex search query into several simple sub-queries. These simple sub-queries are forwarded to the backend and the results are properly combined.

For example, let's process this complex search query which contains three different sub-queries:

```
(RECORD.id CONTAINS "10030") AND (RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/13/2015))
AND (RECORD.date CONTAINS TIME(HH:MM:SS > 11:20:00))
```

The `ryftdec` search engine decomposes the complex expression into the following tree:



The expression A is called first, as part of a normal search. The results of A are used as input for the B date search sub-query. The results of B are then used as input for the C time search sub-query.

Here's another way to look at the sub-queries:

- AND operator
  - A: `(RECORD.id CONTAINS "10030")`
  - AND operator
    - B: `(RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/13/2015)`
    - C: `(RECORD.date CONTAINS TIME(HH:MM:SS > 11:20:00)`

NOTE: For structured search, it's critical to keep the temporary file extension the same as the input file. For example, if the input file ends with "*.pcrime" then the temporary file must also have the same "*.pcrime" extension. Otherwise, the Ryft hardware won't use the corresponding RDF scheme.

2

2: Running Ryft REST Server

You can run the `ryft-server` on a single instance, or run multiple instances on different ports by running the appropriate command:

- Single instance, Port 8765 is the default listening port:

```
/usr/bin$ ./ryft-server
```

- Multiple instances on different ports. This command runs another server instance on port 9000 in debug mode.

```
/usr/bin$ ./ryft=server 0.0.0.0:9000 --debug
# or
/usr/bin$ ./ryft-server -l=:9000 --debug
```

It is possible to run multiple server instances on the same machine.

Use `ryft-server --help` to display the list of all supported arguments:

```
      --ldap-basedn=LDAP-BASEDN  LDAP BaseDN for lookups. Required for --auth=ldap.
```

The `ryft-server` Daemon

The normal operation is to run the `ryft-server` as a daemon. The `ryft-server` daemon automatically starts on boot with default arguments, using port 8765 as the listening port.

The service can be manually stopped with this command:

```
$ sudo service ryft-server-d stop
```

Manually start the service using this command:

```
$ sudo service ryft-server-d start
```

Set Arguments

You can pass arguments to the `ryft-server` daemon in the "/etc/ryft-server.conf" file and restarting `ryft-server-d` service.

List of Arguments

The list of available arguments using `ryft-server --help`.

```
ryftuser@ryftone-313:/etc$ ryft-server --help
usage: ryft-server [<flags>]

Flags:
      --help                     Show context-sensitive help (also try --help-long
                                    and --help-man).
      --config=CONFIG            Server configuration in YML format.
      --local-only               Run server is local mode (no cluster).
  -k, --keep                     Keep temporary search result files (debug mode).
  -d, --debug                    Run server in debug mode (more log messages).
      --logging=LOGGING          Fine-tuned logging levels.
      --busyness-tolerance=0     Cluster busyness tolerance.
  -l, --address=":8765"          Address:port to listen on.
  -t, --tls                      Enable TLS/SSL.
      --tls-cert=TLS-CERT        Certificate file. Required for --tls enabled.
      --tls-key=TLS-KEY          Key-file. Required for --tls enabled.
      --tls-address=":8766"      HTTPS address:port to listen on.
  -a, --auth=none                Authentication type: none, file, ldap.
      --users-file=USERS-FILE    User credentials filename. Required for --auth=file.
      --jwt-alg="HS256"          JWT signing algorithm.
      --jwt-secret=JWT-SECRET    JWT secret. Required for --auth=file or --auth=ldap.
      --jwt-lifetime="1h"        JWT token lifetime.
      --ldap-server=LDAP-SERVER  LDAP Server address:port. Required for --auth=ldap.
      --ldap-user=LDAP-USER      LDAP username for binding. Required for --auth=ldap.
      --ldap-pass=LDAP-PASS      LDAP password for binding. Required for --auth=ldap.
      --ldap-query="(&(uid=%s))" LDAP user lookup query. Required for --auth=ldap.
      --ldap-basedn=LDAP-BASEDN  LDAP BaseDN for lookups. Required for –auth=ldap.
```

Configuration File

The `ryft-server` supports additional configuration files. This YAML configuration file can be customized with the `--config` command line option, like this:

```
$ ./ryft-server --config=$path_to_yaml_config_file
```

Additionally, if the configuration file is located at "/etc/ryft-server.conf" then it is automatically used by the `init` script when the service starts. The default configuration is provided by the Debian package.

```
### this configuration file contains ryft-server options.
### most of the options may be overriden by corresponding
### command line option (noted in parenthesis).

### main search engine and its options
search-backend: ryftprim
backend-options:
  ryftprim-legacy: true
#  instance-name: .rest-8765         # server's working directory (inside /ryftone)
  ryftprim-exec: /usr/bin/ryftprim
  ryftone-mount: /ryftone
  minimize-latency: false           # false - wait until ryftprim is finished before
start data processing, true - start immediately once ryftprim is launched
  open-poll: 100ms                  # open INDEX&DATA file poll timeout
  read-poll: 100ms                  # read INDEX&DATA file poll timeout
  read-limit: 1000                  # maximum number of read attempts

### query decomposition:
#  compat-mode: true                 # true - compatibility mode, false - generic
```

```
#  optimizer-limit: 10                # maximum number of subqueries to combine, 0
means do not combine at all, -1 means combine all (no limit)
  optimizer-do-not-combine: feds     # comma-separated search modes that should not be
combined

### start listening on this address and port (--address)
address: :8765

### HTTPS support (--tls, --tls-address, --tls-cert, --tls-key)
tls:
  enabled: false
  address: :8766
  cert-file: "<certificate file name>"
  key-file: "<key file name>"

### authentication type: none, file, ldap (--auth)
auth-type: file

### JWT authentication (--jwt-alg, --jwt-secret, --jwt-lifetime)
### secret may be simple string or file reference, for example "@~/.ssh/id_rsa"
auth-jwt:
  algorithm: HS256
  secret: "<secret key>"
  lifetime: 1h

### file based authentication (--users-file)
auth-file:
  users-file: /etc/ryft-users.yaml

### LDAP based authentication (--ldap-server, --ldap-user, --ldap-pass, --ldap-query,
--ldap-basedn)
auth-ldap:
  #server: ldap.forumsys.com:389
  #username: "read-only-admin,dc=ryft,dc=one"
  #password: "password"
  #query: "(&(cn=%s))"
  #basedn: "dc=ryft,dc=one"
  #server: https://172.16.91.99/ldap
  server: 172.16.91.99:389
  username: "cn=admin,dc=ryft,dc=one"
  password: "7R^73jZPhfhKMsy&3"
  query: "(&(cn=%s))"
  basedn: "dc=ryft,dc=one"
  insecure-skip-tls: true
  #insecure-skip-verify: true

### run server in local mode (no cluster, no consul, no load balancing) (--local-only)
# local-only: true

### run server in debug mode - a lot of log messages (--debug)
# debug-mode: true

### advanced logging options (--logging=debug or --logging=release)
### each section contains set of logging levels
# logging: release
logging-options:
  custom:
    core: debug
    core/catalogs: debug
    core/pending-jobs: debug
  debug:
    core: debug
    core/catalogs: debug
```

```
    core/pending-jobs: debug
    core/busyness: debug
    search/ryftprim: debug
    search/ryfthttp: debug
    search/ryftmux: debug
    search/ryftdec: debug
  release:
    core: info

### keep intermediate INDEX and DATA files, used for debugging (--keep)
# keep-results: true

### report extra information in stats
# extra-request: true

### busyness tolerance (--busyness-tolerance)
# busyness-tolerance: 1

### HTTP/HTTPS read/write timeout
# http-timeout: 1h

### REST server's shutdown timeout (wait for active requests)
# shutdown-timeout: 10m

### server's runtime settings (list of pending jobs, etc...)
settings-path: /var/ryft/server.settings

### custom hostname (from OS if empty)
# hostname: node-1

### catalogs related options
catalogs:
  max-data-file-size: 64MB       # data file size limit: KB, MB, GB, TB
  cache-drop-timeout: 10s        # internal cache lifetime
  default-data-delim: "\n\f\n"   # default data delimiter
  temp-dir: /tmp/ryft/catalogs   # for temporary files

### post-processing scripts
post-processing-scripts:
  false:
    path: [/bin/false]
  cat:
    path: [/bin/cat]
  jq_c:
    path: [/usr/bin/jq, -c, .]
  jq_ab:
    path: [/usr/bin/jq, -c, "{\"a+b\": (.a + .b), \"a\": .a, \"b\": .b}"]
```

Each section of the configuration file is covered in the following pages.

Main Search Engine and Its Options

Using a configuration file, it is possible to change the main search engine and its options. This is the file format:

```
### main search engine and its options
search-backend: <search engine>
backend-options:
  <search engine options>
```

The `search-backend` is the search engine name and may be one of these values:

- `ryftprim` - use the `ryftprim` command line tool to access Ryft server (used by default).
- `ryftone` - use the `libryftone` library to access Ryft server.
- `ryfthttp` - use another `ryft-server` instance to access Ryft server.

The `backend-options` is the search engine specific option. For example, `ryftprim` engine supports the following options and would look like this:

```
### main search engine and its options
search-backend: ryftprim
backend-options:
  ryftprim-legacy: true
  instance-name: .ryft/8765    # server's working directory (inside /ryftone)
  ryftprim-exec: /usr/bin/ryftprim
  ryftone-mount: /ryftone
  minimize-latency: false      # false – wait until ryftprim is finished before start
data processing, true – start immediately once ryftprim is launched
  open-poll: 100ms             # open INDEX&DATA file poll timeout
  read-poll: 100ms             # read INDEX&DATA file poll timeout
  read-limit: 1000             # maximum number of read attempts
```

Query Decomposition

This section enables you to specify how the queries should be parsed or decomposed of its individual parts, and its queries.

```
### query decomposition:
# compat-mode: true            # true – compatibility mode, false – generic
# optimizer-limit: 10          # maximum number of subqueries to combine, 0 means do
not combine at all, -1 means combine all (no limit)
optimizer-do-not-combine: feds # comma-separated search modes that should not be
combined
```

Field	Description
`compat-mode`	True or False Flag. A flag used to switch REST server into "compatibility" mode. Run the REST server on old firmware that does not use the "generic" syntax. True = run in compatibility mode to run in both the new and old syntax. False = run only the new "generic" syntax.
`optimizer-limit`	Integer. Maximum number of subqueries that may be combined in one command line. Default is "-1" which is no limit. 0 = do not combine any subqueries -1 = combine all subqueries, with no limit.
`optimizer-do-not-combine`	String. Comma-separated list of search modes that should not be combined. By default, "feds" cannot be combined any FEDS subqueries into one command line – it must remain as separate subqueries. Multiple modes can be specified by separating with a comma.

Server Listening Port

Specify the listening port on this Ryft ONE server. The port specified in the configuration file can be overridden at the command line.

By default, the configuration file specifies port 8765, like this:

```
### start listening to this address and port (--address)
address::8765
```

But if the server starts as this:

```
ryft-server –config=/etc/ryft-server.conf –address=0.0.0.0:9000
```

Then the actual option for the address will be 0.0.0.0:9000, since it goes last.

Using the `address` option, it's possible to customize the server's listen address. Its equivalent to the `--address` command line option. By default, "0.0.0.0:8765" is used.

TLS Server

Configure the Transport Layer Security (TLS) protocol server to run with HTTPS enabled . Use the `--tls`, `--tls-address`, `--tls-cert`, and `--tls-key` command line options or just the corresponding `tls` section in the configuration file:

```
### HTTPS support (--tls, --tls-address, --tls-cert, --tls-key)
tls:
  enabled: false
  address: :8766
  cert-file: "<certificate file name>"
  key-file: "<key file name>"
```

The HTTPS can be enabled or disabled using the boolean `enabled` flag. The listen port address can be customized using the `address` option. Note that the port number should be different from the normal address.

Two files should be provided to enable HTTPS:

- Certificate file `cert-file`
- Corresponding certificate key `key-file`.

Authentication

There are a few sections related to authentication.

```
### authentication type: none, file, ldap (--auth)
auth-type: none
```

The `auth-type` is used to select authentication provider. It can be one of the following options:

- `auth-type: none` – disables authentication.
- `auth-type: file` – enables authentication
- `auth-type: ldap` – enables authentication

If authentication is enabled via file or ldap, then you must include and enable the auth-jwt section, below.

JWT Authentication

If JSON Web Token (JWT) authentication is enabled, then include the `auth-jwt` section:

```
### JWT authentication (--jwt-alg, --jwt-secret, --jwt-lifetime)
### secret may be simple string or file reference, for example "@~/.ssh/id_rsa"
auth-jwt:
  algorithm: HS256
  secret: "<secret key>"
  lifetime: 2h
```

- Customize the sign in by using the algorithm option. In this example, it uses HS256.
- The `secret` can be simple string or file reference:
  - `secret: "my super secret key"` or
  - `secret: "@~/.ssh/id_rsa"` to use `~/.ssh/id_rsa` file content as a secret.
- JWT token lifetime can be customized via `lifetime` option. By default, it's `lifetime: 1h`.

File Based Authentication - `auth-file`

If simple file is used as an authentication provider `auth-type: file`, then provide the user credentials like this:

```
### file based authentication (--users-file)
auth-file:
  users-file: /etc/ryft-users.yaml
```

The file formats are described in the Authentication section below.

LDAP Based Authentication

If Lightweight Directory Access Protocol (LDAP) is used as an authentication provider `auth-type: ldap`, then provide the LDAP credentials:

```
### LDAP based authentication (--ldap-server, --ldap-user, --ldap-pass, --ldap-query,
--ldap-basedn)
auth-ldap:
  #server: ldap.forumsys.com:389
  #username: "read-only-admin,dc=ryft,dc=one"
  #password: "<password>"
  #query: "(&(cn=%s))"
  #basedn: "dc=ryft,dc=one"
  server: 172.16.91.99:389
  username: “cn=admin,dc=ryft,dc=one”
  password: “7R^73jZPhfhKMsy&3”
  query: “(&(cn=%s))”
  basedn: “dc=ryft,dc=one”
  insecure-skip-tls: true
#  insecure-skip-verify: true
```

- Use `server` to customize the LDAP server address.
- Use `username` and `password` to customize the read-only user which is used to send search request to the LDAP service.

- Use `query` and `basedn` to specify attribute name which is used to search and base DN.

There are also a few options related to security. By default, `ryft-server` tries to connect LDAP using TLS.

- To disable TLS, set `insecure-skip-tls: true`.
- To disable certificate verification (may be useful if LDAP uses self-signed certificate), set `insecure-skip-verify: true`.

Run Server in Local Mode

Use to run `ryft-server` outside cluster, in local mode, when consul service is not running. Performs all requests locally, even if `local=false` variable is set. There is no load balancing. It's equivalent to using the `--local-only` command line option.

```
### run server in local mode (no cluster, no consul, no load balancing) (--local-only)
# local-only: true
```

Run Server in Debug Mode

Use to enable extensive logging; useful for troubleshooting. It's equivalent to using the `--debug` command line option.

```
### run server in debug mode - a lot of log messages (--debug)
# debug-mode: true
```

Advanced Logging Options

Customize and fine-tune system logging of what to log and the corresponding logging levels. By default, all logs are set to “info” level. Set to “debug” to include in debugging.

```
### advanced logging options (--logging=debug or --logging=release)
### each section contains set of logging levels
# logging: release
logging-options:
  custom:
    core: debug
    core/catalogs: debug
    core/pending-jobs: debug
  debug:
    core: debug
    core/catalogs: debug
    core/pending-jobs: debug
    core/busyness: debug
    search/ryftprim: debug
    search/ryfthttp: debug
    search/ryftmux: debug
    search/ryftdec: debug
  release:
    core: info
```

You can also change the logging options on the command line using the `--logging=debug` or `--logging=release` options.

Log File

The `ryft-server-d` service log file, “server.log,” is located in the “/var/log/ryft/” directory.

To view logs in real-time:

```
$ tail -f /var/log/ryft/server.log
```

Keep Search Result Files

The `ryft-server` uses the dedicated instance directory "/ryftone" to keep temporary results and files created by `ryft-server`. The name of the instance directory is generated using port number "/ryftone/.rest-$PORT/" but it can be customized using the `keep-results` search configuration file:

```
### keep intermediate INDEX and DATA files, used for debugging (--keep)
# keep-results: true
```

By default, `ryft-server` removes all search results from the "/ryftone/.rest-$PORT/" directory, but it may be prevented by using the `--keep` command line option:

```
$ ./ryft-server --keep
```

All temporary result files will be kept under the server's instance directory. This feature is useful for troubleshooting as the files are available for use.

Report More Information in Stats

Use to include extra information in the statistics.

```
### report extra information in stats
# extra-request: true
```

Customize Node Grouping Algorithm

Use in cluster mode to customize node grouping algorithm. See Ryft Cluster documentation for more details. It's equivalent to using the `--busyness-tolerance` command line option.

Ryftrest keeps track of how many requests sent to the R1 servers, in cluster, and uses this to help load balance the requests/serves.

```
### busyness tolerance (--busyness-tolerance)
# busyness-tolerance: 1
```

Timeout for HTTP/HTTPS Connections

Use as read request/write response timeout for HTTP/HTTPS connections. By default, set to `1h` (one hour).

```
### HTTP/HTTPS read/write timeout
# http-timeout: 1h
```

Shutdown Timeout

Use to set the ryftrest server's shutdown timeout. The amount of time to wait for active requests.

```
### REST server's shutdown timeout (wait for active requests)
# shutdown-timeout: 10m
```

Server Runtime Settings

Supply the full path to the server's runtime settings. By default, set to "/var/ryft/server.settings".

```
### server's runtime settings (list of pending jobs, etc...)
settings-path: /var/ryft/server.settings
```

Custom Hostname

Specify the custom hostname. Value from OS is used if a hostname is not provided.

```
### custom hostname (from OS if empty)
# hostname: node-1
```

Catalog Configuration

Some catalog related options can be customized in the "catalogs" section:

```
catalogs:
  max-data-file-size: 64MB       # data file size limit: KB, MB, GB, TB
  cache-drop-timeout: 10s        # internal cache lifetime
  default-data-delim: "\n\f\n"   # default data delimiter
  temp-dir: /tmp/ryft/catalogs   # for temporary files
```

Field	Description
`max-data-file-size`	use this to customize the catalog data size limit. If there is no more space in current catalog's data file, then new one will be started. It's possible to use various units, for example MB for megabytes (10241024) and GBfor gigabytes (10241024*1024)
`cache-drop-timeout`	There is an internal catalog cache. Each catalog entry has it's own drop timeout or lifetime. By default it's 10 seconds but can be changed via cache-drop-timeout option. Options are h (hours), m (minutes), s (seconds), and ms (milliseconds).
`default-data-delim`	Data delimiter is used to separate different small files inside a bigger data file. If delimiter is non empty, it will be placed each time a new file part is written to catalog. The main purpose of this delimiter is to separate RAW text to avoid possible collisions on the file boundaries. For structured data the data delimiter is not so important. Anyway it can be customized via default-data-delim option.
`temp-dir`	Sometimes catalog need to save file content into temporary file. These temporary files are placed in temp-dir directory.

See 5: Create Catalog for Large Data Sets for more information about catalogs.

Post Processing Script Configuration

Configure the list of custom external trusted applications or scripts that you can use to transform a record:

```
### post-processing scripts
```

```
post-processing-scripts:
  false:
    path: [/bin/false]
  cat:
    path: [/bin/cat]
  jq_c:
    path: [/usr/bin/jq, -c, .]
  jq_ab:
    path: [/usr/bin/jq, -c, "{\"a+b\": (.a + .b), \"a\": .a, \"b\": .b}"]
```

Each item is a script name and a path containing the full pathname to the external application or script, along with additional command line options. For instance, the last path line, above, includes a path "/usr/bin/jq" and an option "-c" for the script "jq_ab."

See Post-Process Transformations for information about how to use the transform function.

Authentication

The following types of authentication are supported (these hyperlinks take you to other websites):

- Basic - Basic Access Authentication
- JWT – JSON Web Tokens

To verify user credentials, the LDAP service or simple file may be used.

The following endpoints are protected:

- /search
- /count
- /files

If authentication is enabled, the `ryft-server` checks for `Authorization` HTTP header.

If the `Authorization` header contains the `Basic` keyword, the basic authentication is used. The `ryft-server` extracts username and password from the header and checks that the user is authorized to access the requested resources.

Otherwise, if the `Authorization` header contains the `Bearer` keyword, the JWT is used. The `ryft-server` extracts JWT token from the header and uses it.

There are two special endpoints for JWT authentication:

- `/login` - used to get JWT token
- `/token/refresh` - used to refresh existing token

JWT Login

The `/login` endpoint expects `{"username":"login", "password":"password"}` JSON structure as an input. If the credentials are valid, the JWT token is provided.

Here's an example:

```
curl -d '{"username":"admin","password":"admin"}' "localhost:8765/login"
```

It returns the following:

```
{"expire":"2016-07-11T08:13:09-04:00",
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0NjgyMzkxODksImlkIjoiYWRtaW4i
LCJvcmlnX2lhdCI6MTQ2ODIzNTU4OX0.X_sO1pimiDQ9XGg37PzTYIB9ohu4DJM8VG9lgqd4sqg"}
```

Having this token, it's now possible to send authorized requests, like this one:

```
TOKEN=`curl -d '{"username":"testuser","password":"testpsswd"}' "localhost:8765/login"
| jq -r .token`
curl -H "Authorization: Bearer $TOKEN"
http://localhost:8765/search?query=Joe&files=*.txt
```

JWT Options

To pass JWT secret to the server, use the configuration file or `--jwt-secret` command line option:

```
ryft-server --jwt-secret=my-secret-key
ryft-server --jwt-secret=@my-secret-file
ryft-server --jwt-secret=hex:6D792D7365637265742D6B6579
ryft-server --jwt-secret=base64:bXktc2VjcmV0LWtleQ==
```

By default, HS256 signing algorithm is used. You can override it by using the `--jwt-alg` command line option, like this: `--jwt-alg=RS256`.

The default token lifetime is 1 hour. To change it, use `--jwt-lifetime` command line option. The overall token refresh timeout is 10 lifetimes.

LDAP Customization

Most LDAP customization can be done via the configuration file. Some command line options are also available. (Check `ryft-server --help` output for more information.)

- Provide the LDAP server address. See `--ldap-server` command line option.
- Provide special read-only account for search requests. Customize the account credentials using the `--ldap-user` and `--ldap-pass` command line options.

The `--ldap-query` and `--ldap-basedn` are used to finish the LDAP search request. Query format is used to select appropriate RDN, for example `(&(cn=%s))`.

There are a few security related options (currently in configuration file only) that could be used to disable TLS or to disable TLS certificate verification. Please check corresponding auth-ldap section of the configuration file.

Simple Text File

Simple text file may be used as a list of user credentials. Here are two examples in YAML and JSON format.

YAML format:

```
- username: "admin"
  password: "admin"
  home: "/"
- username: "test"
  password: "test"
  home: "/test"
  cluster-tag: "test"
- username: "foo"
```

```
  password: "foo"
  home: "/foo"
  cluster-tag: "foo"
```

JSON format:

```
[
  {"username":"admin", "password":"admin", "home":"/"},
  {"username":"test", "password":"test", "home":"/test", "cluster-tag":"test"},
  {"username":"foo", "password":"foo", "home":"/foo", "cluster-tag":"foo"}
]
```

The “home” is a directory inside the “/ryftone” mount point. It is used to separate data of various users. A user is only authorized to access its home directory.

The `cluster-tag` is used for partitioning. If a user has custom partitioning rules, they are located under the `{cluster-tag}/partitions` KV prefix.

To run the server, use this command line:

```
ryft-server --auth=file --users-file "ryft-users.yaml"
```

Cluster Mode

All cluster nodes should have the same list of users (or the same LDAP configured) and the same secret key. It's important to be able to pass authentication tokens between cluster nodes. Ryft server uses the `Authorization` HTTP header "as is" to redirect search requests.

NOTE: Each user should have the same home directory on each cluster node.

3

3: REST API

The `ryft-server` supports these REST endpoints:

- /files
- /search
- /count
- /version

The main API endpoints are `/search` and `/count`. Both have similar parameters. However, the `/count` endpoint does not transfer all found data. Instead, it prints the number of matches and associated performance numbers. The minimum required parameters are search query and the set of files to search.

NOTE: All examples in this section assume that "ryftone-777" is the `ryft-server` host name.

Here's an example of a simple request:

```
curl "http://ryftone-777:8765/search?query=Joe&files=*.txt"
```

You can also customize the search.

The following command captures 5 bytes of data surrounding the search value, and executes a fuzzy edit distance search (fuzziness=2) instead of fuzzy hamming search, which is used by default:

```
curl "http://ryftone-
777:8765/search?query=Joe&files=*.txt&mode=feds&surrounding=5&fuzziness=2"
# - or -
curl --get --data-urlencode 'query=(RAW_TEXT CONTAINS "Joe")' \
  "http://ryftone-777:8765/search?files=*.txt&mode=feds&surrounding=5&fuzziness=2"
```

By default, cluster mode is used. To execute a search on a single node use the `local` query parameter:

```
curl "http://ryftone-777:8765/search?query=Joe&files=*.txt&local=true"
```

To get human readable data (instead of base-64 encoded bytes) use `format=utf8`. This parameter asks `ryft-server` to interpret found bytes as UTF-8 string:

```
curl "http://ryftone-777:8765/search?query=Joe&files=*.txt&format=utf8"
```

The `/version` endpoint is used to check server's version:

```
curl "http://ryftone-777:8765/version"
```

This request prints current server version and corresponding git hash number, which is useful for bug reporting.

Files Endpoint - `/files`

Three actions are associated with the `/files` endpoint:

- `GET /files` – list the directories, subdirectories and files for the Ryft server node or cluster. Also used to download a standalone file or part of a catalog.
- `POST /files` – upload a file to Ryft server node or cluster. Supports the Catalog feature to upload multiple small files into one large file, or to append a datafile to an existing datafile.
- `DELETE /files` – delete any file, directory or catalog on the Ryft server node or cluster.

NOTE: The `/files` endpoint is protected, and the user is prompted to provide valid credentials. See authentication for more details.

GET /files

Use `GET /files` to get a list of the directories, subdirectories and files on the specified Ryft server node or cluster, and to download a standalone file or part of a catalog.

The parameters for `GET /files` endpoint are:

Parameter	Type	Description
`dir`	String	Directory from which to get content. Must be a valid path. By default, it is the root "/ryftone" directory. Directory name should be relative to the Ryft volume, so using `dir=/test` means report the contents of "/ryftone/test" directory on the Ryft server.
`hidden`	Boolean	Report hidden files flag. True or false. By default, it is false.
`local`	Boolean	The local/cluster flag. True or false. By default, it is false, which means it is cluster.

For example, this command specifies "dir=/" which means to print the contents of the "/ryftone" directory on the local server:

```
/files?dir=/&local=true
```

The resulting output may look like this:

```
{
  "dir": "/",
  "files": [
    "chicago.pcrime",
    "passengers.txt"
  ],
  "folders": [
    "demo",
    "regression",
    "test"
  ]
```

```
}
```

By default, “dir=/” is the root directory. The directory name should be relative to the Ryft volume and the user’s home. For example, “dir=/mydata” reports the contents of “/ryftone/mydata” on the Ryft ONE.

Get Catalog’s Content

To get a catalog’s content, leave the `file` parameter empty and use these parameters:

Parameter	Type	Description
`catalog`	String	The valid full path of the catalog, relative to the directory specified by `dir` to download.
`local`	Boolean	The local/cluster flag. True or false. By default, it is false, which means it is cluster.

Download a Standalone File

To download a standalone file, use these parameters:

Parameter	Type	Description
`dir`	String	Directory from which to get content. Must be a valid path. By default, it is the root “/ryftone” directory. Directory name should be relative to the Ryft volume, so using `dir=/test` means report the contents of “/ryftone/test” directory on the Ryft server.
`file`	String	The full file path name of the file to download, relative to the specified directory. • Standalone file: full file path name relative to the directory specified by • Catalog: filename within the specified catalog.

Specify the directory and file by using one or both parameter. The following queries are the same:

- Specify both files, separately:
  ```
  curl 'http://localhost:8765/files?dir=webtraffic&file=jan2017.txt'
  ```
- Specify file with full path:
  ```
  curl 'http://localhost:8765/files?file=webtraffic/jan2017.txt'
  ```
- Specify the directory in the URL, and file separately:
  ```
  curl 'http://localhost:8765/files/webtraffic?file=jan2017.txt'
  ```

Download File from a Catalog

To download a datafile from a catalog, use these parameters:

Parameter	Type	Description
`catalog`	String	The full file path name of the catalog to download, relative to the specified directory.
`dir`	String	The directory where the catalog is located and from which to download.
`file`	String	The filename inside the specified catalog to download.

Specify one, some or all the parameters. The following queries are the same:

- Specify all three parameters, individually:
```
curl 'http://localhost:8765/files?dir=yelp&catalog=2016yelp.cat&file=Feb16.txt'
```
- Specify catalog name with full path
```
curl 'http://localhost:8765/files?catalog=yelp/2016yelp.cat&file=Feb16.txt'
```
- Specify the directory in the URL, and the catalog and file separately.
```
curl 'http://localhost:8765/files/yelp?catalog=2016yelp.cat&file=Feb16.txt'
```

The `Content-Range` and `If-Modified-Since` headers are supported so you can download only part of a file.

POST /files – Standalone and Catalog

To post (upload) a file, you need to provide the content using `Content-Type`. There are two types of headers:

- `Application/octet-stream`
```
$ curl -X POST –data-binary @test.txt\ -H 'Content-Type: application/octet-
stream' -s "http://localhost:8765/files?file=/regression/test.txt" | jq .
```
- `Multipart/form-data` – actual file content provided via `-F file=` key.
```
$ curl -X POST -F file=@</path/to/file.txt> -s
"http://localhost:8765/files?file=/test/file.txt&length=100" | jq .
```

Two sets of parameters are available, depending on whether you are posting standalone files or posting files to a catalog. More information about appending to a standalone file or creating a catalog from the command line are covered in Chapter 5: Create Catalog for Large Data Sets.

The parameters for posting standalone and catalog files are listed below. Where possible, the full description is only provided in the first occurrence.

Parameter	File Type	Type	Description
`catalog`	Catalog	String	Full path for the catalog to append the named file. If the catalog does not exist, one is automatically created. Use the keyword `{{random}}` to generate a unique catalog name, like this: `catalog=test-{{random}}.catalog` will automatically create a catalog with a name like "test-aabbccddeeff.catalog".

Parameter	File Type	Type	Description
			Response body displays the randomly-generated catalog name.
`delimiter`	Catalog	String	Data delimiter; a separator between different file parts. It is very important, especially for RAW text files, to use something like `delimiter=%0a`. Otherwise, unexpected text matches can be found on file part boundaries. If no delimiter is provided the default value will be used. The default delimiter can be customized via ryft-server's configuration file. Once provided, the delimiter cannot be changed for the same catalog.
`file`	Standalone & Catalog	String	The full path of file to upload For example, if `file=bar/foo.txt` then the data will be saved under "/ryftone/<userid>/bar/foo.txt". If `catalog` parameter is not specified, then the `file` parameter contains the full path. All non-existing sub-directories are created automatically. Use the keyword `{{random}}` to generate unique filenames. This keyword is replaced with some unique hexadecimal string. For example, `file=foo-{{random}}.txt` is replaced with something like "foo-aabbccddeeff.txt". The actual filename is reported in the response body.
`length`	Standalone & Catalog	Integer	Optional. Use this to specify the uploading data length in bytes.
`lifetime`	Standalone & Catalog	String	The lifetime of the uploaded file. Optional. If specified, the file or catalog will be deleted after specified amount of time. For example if "lifetime=1h" is provided the file will be available during a hour and then will be automatically removed.
`local`	Standalone & Catalog	Boolean	Local/cluster flag. Optional. Default is cluster.
`offset`	Standalone & Catalog	Integer	Position of uploaded chunk. Optional. It's possible to upload just a part of file. If offset query parameter is present then the data will be saved using this offset as write position in destination file. Using this parameter, it's possible to continue upload of failed data, or to split a file and upload it in chunks.
`share-mode`	Catalog	String	Use to control access to data files. Optional.

Parameter	File Type	Type	Description
			By default, uploads are not allowed if an operation is in progress and vice versa. Does not support `share-mode=force_ignore` / `share-mode=ignore`. See Search Parameter: share-mode for more information.

DELETE /files - Delete Files, Directories or Catalogs

It's possible to specify a file, a group of files using a wildcard, a directory or catalog to delete. Multiple parameters can be used together. Wildcards are supported, so to delete all JSON files you would use `file=*.json`.

All file names should be relative to the Ryft volume and your home directory. For example, the request "file=bar/foo.txt" request will delete "/ryftone/mydir/bar/foo.txt" on the Ryft ONE (assuming user's home directory is "mydir").

Here are the delete parameters:

Parameter	Type	Description
`dir`	String	Directory to delete
`file`	String	Standalone file to delete
`catalog`	String	Catalog to delete
`local`	Boolean	Local/cluster flag

Search Endpoint - `/search`

The `GET /search` endpoint is used to search data on Ryft servers. This endpoint is protected; the user should provide valid credentials. See Authentication for more details.

These are the content types that the server can produce:

- `Accept: application/json` – used by default
- `Accept: application/msgpack` – used internally in cluster mode
- `Accept: text/csv`

These are the supported query parameters. Detailed description for each follows the table:

Parameter	Type	Description
`accept`	String	Search endpoint produces data encoded as: • json – used by default • msgpack – used internally in cluster mode

Parameter	Type	Description
		• csv
query	String	Required. The search expression.
file	String	Required. The set of files to search. Wildcards are supported.
mode	String	The search mode. If no search mode is specified, fuzzy hamming search is used, by default, for simple queries.
surrounding	Uint16	The data surrounding width. Number of characters to return before and after the search string. Used with RAW text search. Default is `surrounding=0`.
fuzziness	Uint8	The fuzziness distance. Measured as the maximum Hamming distance for “fhs” (fuzzy hamming search) mode, or edit distance for “feds” (fuzzy edit distance search) mode. Default is `fuzziness=0`.
format	String	The structured search input data format. Used with structured search.
cs	Boolean	The case sensitive flag. Default is `cs=false`.
reduce	Boolean	The reduce flag for FEDS.
fields	String	The set of fields to get. Comma-separated list of requested fields. Use with structured search to minimize output.
transform	String	The post-process transformation. Available options are: • `match(“<expression>”)` • `replace("<expression>", "<template>")` • `script("<script name>")`
data	String	The name of data file to keep. File will be overwritten.
index	String	The name of index file to keep. File will be overwritten.
delimiter	String	The string used to separate found records.
share-mode	String	Customize sharing mode, related to protecting data files from simultaneous read and write.

Parameter	Type	Description
nodes	Integer	The number of processing nodes to use (1-4). If omitted, all available nodes are used.
local	Boolean	The local/cluster search flag. Default is `local=false`, cluster mode.
stats	Boolean	The statistics flag. Include search statistics in results. Default is `stats=false`.
performance	Boolean	Flag to report performance metrics.
limit	Int	Limit the total number of records reported.
stream	Boolean	Internal. The stream output format flag.
ep	Boolean	Internal. The error prefix flag. Default is `ep=false`.

Search Parameter: `query`

This parameter enables you to use one or more subqueries, connected using logical operators. It can be used for both unstructured and structured searches.

To execute an unstructured text search for "The Batman" use this search expression:

```
query=(RAW_TEXT CONTAINS "The Batman")
```

To execute a structured search, use this expression to define the search field:

```
query=(RECORD.AlterEgo CONTAINS "The Batman")
```

Depending on which search mode you use, the exact search query format may differ.

See Ryft Open API or General Search Syntax section for more details on search expressions.

Simple/Plain Queries

Use `ryft-server` to execute simple, plain queries without any key words. From our previous example, the statement `query=Batman` will automatically convert to `query=(RAW_TEXT CONTAINS "Batman")`. (In the background, the query is translated to be: `query=(RAW_TEXT CONTAINS "\x42\x61\x74\x6d\x61\x6e")`, because `ryft-server` uses hex encoding to avoid any possible escaping problems).

NOTE: The statement `query=Batman` only works for unstructured text search. It is not supported for structured search.

Complex Queries

You can also execute complex queries that contain several different search expressions at one time.

Here is an example that contains two search expressions: text search and date search:

```
(RECORD.id CONTAINS "100") AND (RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/15/2015))
```

The `ryft-server` command splits this expression into two separate queries. It then calls the Ryft hardware two times so that the results of the first call are used as the input for the second call, like this:

- `(RECORD.id CONTAINS "100")`
- `(RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/15/2015))`

Multiple AND and OR operators are supported by the `ryft-server` within complex search queries. An expression tree is built and each node is passed to the Ryft hardware, and then the results are properly combined.

NOTE: If the search query contains two or more expressions of the same type (text, date, time, numeric), then that query will not be split into subqueries because the Ryft hardware supports those types of queries directly.

It is also possible to use advanced text search queries to customize some parameters within search expression. Here is an example:

```
(RAW_TEXT CONTAINS FHS("555",CS=true,DIST=1,WIDTH=2)) AND (RAW_TEXT CONTAINS
FEDS("777",CS=true,DIST=1,WIDTH=4))
```

The `ryft-server` splits this expression into two Ryft calls:

- `(RAW_TEXT CONTAINS "555")` with FHS search mode, case sensitive turned on, fuzziness distance of 1 (`fuzziness=1`) and surrounding width of 2 (`surrounding=2`).
- `(RAW_TEXT CONTAINS "777")` with FEDS search mode, case sensitive turned on, fuzziness distance of 1 (`fuzziness=1`) and surrounding width of 4 (`surrounding=4`).

This advanced search query syntax overrides the following global parameters:

- search type: FHS or FEDS (exact search is used if fuzziness is zero)
- case sensitivity: CS=
- fuzziness distance: `DIST=`
- surrounding width: `WIDTH=`

If nothing is provided, the global options are used by default.

Any option may be omitted, like this example:

```
(RAW_TEXT CONTAINS FHS("555")) AND (RAW_TEXT CONTAINS FEDS("777",CS=false))
```

See General Search Syntax for more information about the search syntax.

Search Parameter: `file`

Use the `file` parameter to specify search input file(s). At least one file should be provided. Multiple files can be provided as a list or wildcard, like this:

- Single file: `file=1.txt`
- List of files: `file=1.txt&file=2.txt&file=3.txt`

- Wildcard: `file=*txt`

You can also specify a catalog name as a file parameter. The ryft-server automatically detects catalogs and performs the substitution. You can also use the `catalog= alias`.

The `files=` parameter (files vs. file) is also supported for backwards compatibility

Search Parameter: `mode`

The following search modes are supported by the `ryft-server`:

- `es` for exact search
- `fhs` for fuzzy hamming search (default, if no mode is specified)
- `feds` for fuzzy edit distance search
- `ds` for date search
- `ts` for time search
- `ns` for numeric search
- `cs` for currency search
- `ipv4` for IPv4 search
- `ipv6` for IPv6 search

NOTE: If no search mode is specified, then by default fuzzy hamming search is used for simple queries.

It is also possible to automatically detect search modes. If the search query contains the keyword "DATE" then the `DATE` search will be used. It's the same when the keyword "TIME" is used for `TIME` search, and "NUMBER" or "CURRENCY" keywords for `NUMERIC` search.

In case of complex search queries, the specified mode is used for text or structured search, only. Date, time and numeric search modes are automatically detected by their corresponding keywords.

Search Parameter: `surrounding`

Use the `surrounding` parameter with raw text search to specify the number of characters to return both before and after the match. You can specify up to a maximum of 64,000 bytes before and after the match. For anything other than raw text, this parameter is ignored.

By default, `surrounding=0`. To specify 20 characters before and after the search string, use `surrounding=20` in the query string.

If you want to return the entire surrounding line, which is especially useful for CSV files, then specify `surrounding=line`.

Search Parameter: `fuzziness`

The fuzziness of the search means different things, depending on the type of fuzzy search you perform. You can specify up to a maximum fuzziness of 255 when using either of these fuzzy search functions:

- Fuzzy Hamming Search (`fhs`): Fuzziness is measured as the maximum Hamming distance allowed to declare a match.

- Fuzzy Edit Distance Search (`feds`): Fuzziness is measured as the number of insertions, deletions or replacements required to declare a match.

By default, `fuzziness=0`.

Search Parameter: `format`

Use `format` to specify the input data format for structured search.

By default, structured search uses `format=raw`. That means that found data are reported as base-64 encoded raw bytes.

There are three other options: `format=xml`, `format=json`, and `format=utf8`.

XML Format

If the 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 Format

If the input file set contains JSON data, use the `format=json` query parameter to decode data to JSON

UTF8 Format

Use `format=utf8` to get human readable data (instead of base-64 encoded bytes). This parameter asks `ryft-server` to interpret found bytes as UTF-8 string:

```
curl "http://ryftone-777:8765/search?query=Joe&files=*.txt&format=utf8"
```

A `format=raw` - base-64 encoded raw bytes sample may look like this:

```
{
  "data": "LDMxMC01NTUtMzQyNQ==",
  "_index": {}
}
```

That same sample displayed as `format=utf8` would look like this:

```
{
  "data": ",310-555-3425",
  "_index": {}
}
```

If only the index information is relevant, you can specify `format=null` which tells ryft-server to ignore all the output data and only keep the indexes.

Search Parameter: `cs`

Use the `cs` parameter as a flag to specify the search text case-sensitivity. By default, `cs=false`.

For example, if the search is case-sensitive (`cs=true`), then searching for the string "John" will not find any occurrences of "JOHN". If the same search is done with `cs=false`, then case is ignored entirely and all possible capitalizations of the text will be found (e.g. "jOhn" or "JOHn").

Search Parameter: `reduce`

The `reduce` parameter flag is only used for edit distance search. By default, `reduce=true`, which tells the search engine to remove duplicates.

Search Parameter: `fields`

The comma-separated list of fields to use for structured search. If omitted, all fields are used.

This parameter is used to minimize structured search output or to get just subset of fields. For example, to get identifier and date from a `*.pcrime` file, pass `format=xml&fields=ID,Date`.

The same is true for JSON data: `format=json&fields=Name,AlterEgo`.

Search Parameter: `transform`

The `transform` query option defines a custom transformation. There may be a few transformations joined to the transformation chain - where the output of the first transformation goes to the input of the second transformation.

A single transformation can be one of:

- `transform=match("<expression>")`
- `transform=replace("<expression>", "<template>")`
- `transform=script("<script name>")`

NOTE: The output statistic contains initial number of matches. So we can check the number of dropped records as difference between the total number of matches and the actual number of records received. The same is true for indexes. Indexes contain initial data position and length without any transformations reflected.

A few transformations can be specified with several `transform` parameters. In this case, all transformations are combined into transformation chain.

For more information, see Post-Process Transformations

Search Parameters: `data` and `index`

By default, all search results are deleted from the Ryft server once they are delivered to the user. The `data` and `index` parameters enable you to preserve the results, giving you the ability to use the results as the input for a second, subsequent, search.

WARNING: The data or index files will be overridden.

Data Parameter

Using the `data=output.dat` parameter creates the search results file “output.dat” on the Ryft server under “/ryftone/output.dat.” It is then possible to use “output.dat” as the input file for the subsequent search call using `files=output.dat`.

NOTE: It is important to use consistent file extensions for the structured search to let Ryft use the appropriate RDF scheme.

Index Parameter

Using the second parameter `index=index.txt` keeps the search index file under the "/ryftone" directory.

NOTE: As noted in the Ryft API guide, an index file should always have a "*.txt" extension.

Search Parameter: `delimiter`

Use the string specified by this parameter to customize output format to separate found records in the output data file. By default, there is no delimiter.

To use the Windows newline function, use this value: `delimiter=%0D%0A`.

Search Parameter: `share-mode`

The ryft-server protects data files from simultaneous read and write. This parameter is used to customize sharing mode. By default, it's set to `share-mode=wait-0ms` which means to immediately report an error if the data file is busy.

The following sharing modes are supported:

- `share-mode=wait-up-to-10s` or `share-mode=wait-10s`. If data file is busy ryft-server waits up to specified timeout.
- `share-mode=skip-busy` or `share-mode=skip`. If data file is busy then it is removed from input fileset. Note, the input fileset might be empty - ryftprim reports error in this case.
- `share-mode=force-ignore` or `share-mode=ignore`. Forced to ignore any sharing rules. Even if file is busy, try to run the search. Note that the result might be undefined.

Search Parameter: `nodes`

Use the `nodes` parameter to specify the number of Ryft processing nodes that the algorithm should use. A minimally configured Ryft ONE ships from the factory with one processing node (`nodes=1`), and a maximally configured Ryft ONE ships with four processing nodes (`nodes=4`).

If you omit the parameter, the maximum of available nodes is used.

Search Parameter: `local`

The local/cluster flag. By default, `ryft-server` uses cluster mode (`local=false`) which means that the `ryft-server` asks all appropriate nodes in the cluster and then combines the results.

To execute a search on single node use `local=true`.

Search Parameter: `stats`

By default, statistics is not reported (`stats=false`). To check total number of matches and performance numbers, use `stats=true`.

Search Parameter: `performance`

By default, performance metrics are not reported (`performance=false`). To get performance metrics in the statistics, use `performance=true`.

Search Parameter: `limit`

Use the `limit` parameter to limit the total number of records reported. By default, there is no limit, or when you specify `limit=0`.

Search Parameter: `stream`

The `ryft-server` reports results in several formats. By default, the simple JSON object with "results" array and "stats" object is reported. That format is used by the Ryft Web-UI:

```
{
  "results": [
    {
      "Date": "04/15/2015 11:59:00 PM",
      "ID": "10034183",
      "_index": {}
    },
    {
      "Date": "04/15/2015 11:59:00 PM",
      "ID": "10034184",
      "_index": {}
    }
  ],
  "stats": {
    "matches": 2,
    "totalBytes": 6902619,
    "duration": 415,
    "dataRate": 15.862290255994683,
    "fabricDataRate": 15.86229
  }
}
```

But this format is not efficient for cluster nodes communication. We cannot decode a JSON object until all the data is received. Instead, use the `stream` parameter - a sequence of JSON "tag-object" pairs that enables `ryft-server` to decode input data, on the fly:

```
"rec"
{
  "Date": "04/15/2015 11:59:00 PM",
  "ID": "10034183",
  "_index": {}
}

"rec"
{
  "Date": "04/15/2015 11:59:00 PM",
  "ID": "10034184",
  "_index": {}
}

"stat"
{
  "matches": 2,
  "totalBytes": 6902619,
  "duration": 1456,
```

```
  "dataRate": 4.521188500163319,
  "fabricDataRate": 4.521189
}

"end"
```

Search Parameter: ep

The ep parameter is the helper "error prefix" flag for cluster mode. The default is `ep=false`.

To let the user know which cluster node reported the error, `ryft-server` adds the node's hostname to each error message. If `ep=true` (in cluster mode) then the bolded text is added to the error message:

```
{
    "message": "[ryftone-777]: ryftprim failed ...",
   "status": 500
}
```

Search Accept Header

Search endpoint supports three types of encoding:

- json - default
- msgpack - used internally in cluster mode
- csv

The output type can be specified by `Accept` HTTP header.

You can change the response format by specifying Accept with the appropriate encoding type. This is also supported in the Ryft ONE Swagger WebUI.

Accept: application/json

This is default content type. It reports records, errors and statistics in an appropriate JSON object.

Accept: application/msgpack

This content type is used internally for communication between nodes in cluster mode.

Accept: text/csv

Ryft-server supports csv encoding, using RFC 4180 as the foundation.

A csv file contains zero or more records of one or more fields per record. Each record is separated by the newline character, and a comma (‘,’) is the separator between each field, and may contain quotes and newline characters. The final record may optionally be followed by a newline character (‘\n’).

A csv record can be the following types:

- Data record
- Statistic
- Error
- End

Fields which start and stop with quotes (“ ”) are called quoted-fields. The beginning and ending quote are not part of the field.

Within a quoted-field a quote character followed by a second quote character is considered a single quote. Newlines and commas may be included in a quoted-field.

Data Record

The first field is "rec", then INDEX fields and data. Data is reported per selected format (utf8, json, etc).

```
rec,file,offset,length,fuzziness,host,data
```

Here is an example:

```
$ ryftrest -q hello -f test/foo/1.txt -w=10 --format=utf8 --accept=csv
rec,test/foo/1.txt,0,15,0,ryftone-313,"hello worldhel"
rec,test/foo/1.txt,2,25,0,ryftone-313,"llo worldhello worldhell"
rec,test/foo/1.txt,13,25,0,ryftone-313,ello worldhello from curl
rec,test/foo/1.txt,28,25,0,ryftone-313," from curlhello from curl"
rec,test/foo/1.txt,43,25,0,ryftone-313," from curlhello from curl"
rec,test/foo/1.txt,58,25,0,ryftone-313," from curlhello from curl"
rec,test/foo/1.txt,73,25,0,ryftone-313," from curlhello from curl"
stat,16,233,520,0.00042731945331280044,0,0,ryftone-313,null,{}
end
```

The output line is broken down, like this:

```
rec,test/foo/1.txt,2,25,0,ryftone-313,"llo worldhello worldhell"
```

- “rec” – the type
- “test/foo/1.txt” – the filename
- “2” – the offset
- “25” – the length of the search result
- “0” – the search fuzziness
- “ryftone-313” – the name of the host
- “llo worldhelloworldhell” – the matching data string

Statistic

The first field is "stat", then STAT fields:

```
stat,matches,totalBytes,duration,dataRate,fabricDuration,fabricDataRate,host,
details,extra
```

Here is an example:

```
$ ryftrest -q test -w 10 -f test/foo/1.txt --format=utf8 --search --address
localhost:9786 --accept csv
stat,0,252,294,0.0008174351283482144,0,0,ryftone-313,null,{}
end
```

The output is broken down, like this:

- “stat” – the type
- “0” – the total number of matches
- “252” – the total number of Bytes

- “193” – the duration to execute the query
- “0.00081743512834821 44” – the data rate
- “0” – the fabric duration
- “0” – the fabric data rate
- “ryftone-313” – the name of the host
- “null” – any additional details
- “{ }” – any extras such as performance metrics

Error

The first field is "err", then error message:

```
err,message
```

Here is an example:

```
$ ryftrest -q 'hkkkkkkkkkkkkk' -w 10 -f test/foo/1.txt --format=utf8 --search --no-
stats --address localhost:9786 --accept csv
err,"ryftprim failed with exit status 156
ERROR:  An error occurred - results are not valid!
"
err,"500 ryftprim failed with exit status 156
ERROR:  An error occurred - results are not valid!
"
end
```

The output shows the error and error text, followed by “err” and the “error message.”

End

End of file. The first field is "end".

```
end
```

Search Examples

Example 1: Simple Example

This is an example of a simple search request to find “Joe” in all files that end in “*.txt”:

```
/search?query=Joe&files=*.txt
```

Example 2: Single Node Example

By default, cluster mode is used for all searches. To execute a search on a single node use the `local` query parameter:

```
/search?query=Joe&files=*.txt&local=true
```

Example 3: Unstructured Search Example

This is an example of an unstructured search request to find “10” in the file “passengers.txt.” The output will contain 12 bytes of data surrounding the search value, and output stats on the results. The request will be run on only on the local server vs. the cluster.

```
/search?query=10&files=passengers.txt&surrounding=12&fuzziness=0&stats=true&local=true
```

Example 4: Structured Example

This is an example of a structured search request to find "1003100" in the field "id" within all files that end in "*.pcrime" located in the "/" directory. It must be an exact match as fuzziness is set to 0, the format will be XML, and the output will contain the fields ID and Date. The output will also contain stats, and the query will run on the local server vs. the cluster.

```
/search?query=(RECORD.id CONTAINS
"1003100")&files=/*.pcrime&fuzziness=0&format=xml&fields=ID,Date&stats=true&local=true
```

Example 5: Fuzzy Edit Distance Example

The following command captures 5 bytes of data surrounding the search value, and executes a fuzzy edit distance search (`fuzziness=2`) instead of fuzzy hamming search, which is used by default:

```
curl "http://ryftone-
777:8765/search?query=Joe&files=*.txt&mode=feds&surrounding=5&fuzziness=2"
# - or -
curl --get --data-urlencode 'query=(RAW_TEXT CONTAINS "Joe")' \
  "http://ryftone-777:8765/search?files=*.txt&mode=feds&surrounding=5&fuzziness=2"
```

Count Endpoint - `/count`

The `GET /count` endpoint is also used to search data on Ryft boxes. It prints the number of matches and associated performance numbers. It does not transfer all found data.

NOTE: This endpoint is protected and user is prompted to provide valid credentials. See Authentication for more details.

Query Parameters

The supported parameters are listed in this table. All of the parameters are the same as the `/search` parameters. For more information, see Search Endpoint - /search.

Parameter	Type	Description
`query`	String	Required. The search expression.
`files`	String	Required. The set of files to search.
`mode`	String	The search mode.
`surrounding`	Uint16	The data surrounding width.
`fuzziness`	Uint8	The fuzziness distance.
`cs`	Boolean	Case sensitive flag.
`reduce`	Boolean	Reduce flag for FEDS.
`data`	String	Name of the data file to keep.
`index`	String	Name of index file to keep.
`delimiter`	String	The string used to separate found records.

Parameter	Type	Description
`share-mode`	String	Customize sharing mode, related to protecting data files from simultaneous read and write.
`nodes`	Integer	Number of processing nodes. Default is maximum number available on the Ryft server.
`local`	Boolean	True or False. Local/cluster search flag. By default, set to False/Cluster.
`performance`	Boolean	Flag to report performance metrics.

Example 1

Here is an example of counting the number of matches where the letter “a” or “b” is contained in all data files that end with *.pcrime, on the local server:

```
/count?query=(RECORD CONTAINS “a”) OR (RECORD CONTAINS “b”)&files=*.pcrime&local=true
```

Example 2

Here is an example of counting the number of matches where record contains “1003” in “id” and the date is > 04/14/2015 and the time is > 08:30:00 on the “mychicago.pcrime” file, on the local server:

```
/count?query=(RECORD.id CONTAINS "1003") AND (RECORD.date CONTAINS DATE(MM/DD/YYYY >
04/14/2015)) AND (RECORD.date CONTAINS TIME(HH:MM:SS >
08:30:00))&files=mychicago.pcrime&local=true
```

Example 3

Here is an example that uses AND and OR so that either the first criteria (record contains “1003” and date <= 04/12/2015) or the second criteria (time search) is met.

```
/count?query=(RECORD.id CONTAINS "1003") AND ((RECORD.date CONTAINS DATE(MM/DD/YYYY <=
04/12/2015)) OR (RECORD.date CONTAINS TIME(HH:MM:SS >
11:15:00)))&files=mychicago.pcrime&local=true
```

Current Server Version - `/version`

The `GET /version` endpoint is used to verify the current build version of `ryft-server`. This request prints the current server version and corresponding Git hash number. There are no parameters.

```
curl http://ryftone-777:8765/version
```

The resulting output may look like this:

```
{
  "git-hash": "35c358378f7c214069333004d01841f9066b8f15",
  "version": "1.2.3"
}
```

General Search Syntax

A match criteria `query` parameter is used to specify how the search should be performed. Search criteria are made up of one or more relational expressions, connected using logical operations. The Ryft Open API defines query language grammar as consisting of a relational expression that takes the following form:

```
(input_specifier relational_operator primitive(expression[, options]))
```

Let's break this down to its individual parts.

Input Specifier

The `input_specifier` specifies how the input data is arranged. The possible values are:

Specifier	Description
`RAW_TEXT`	The input is a sequence of raw bytes with no implicit formatting or grouping.
`RECORD`	The input is a series of records. Search all records.
`RECORD.<field_name>`	The input is a series of records. Search only the field called `<field_name>` in each record. **NOTE**: For JSON input records, multiple field names can be specified with '.' separators between them to specify a field hierarchy, or with '[]' separators to specify array hierarchy.

Relational Operator

The `relational_operator` specifies how the input relates to the expression. The possible values are:

Specifier	Description
`EQUALS`	The input must match the expression exactly for an exact search, or within the specified Hamming distance for a fuzzy search, with no additional leading or trailing data. This operator only has meaning for record- and field-based searches. If used with a raw text operation, an error is generated; use `CONTAINS` instead.
`NOT_EQUALS`	The input must be anything other than expression. This operator has meaning only for record- and field-based searches. If used with a raw text operation, an error is generated.
`CONTAINS`	The input must contain expression, and may contain additional leading or trailing data.
`NOT_CONTAINS`	The input must not contain expression. This operator has meaning only for record- and field-based searches. If used with a raw text operation, an error is generated.

Primitive

The `primitive` specifies the search primitive associated with the clause. The possible values are:

Specifier	Description
EXACT	Search for an exact match.
HAMMING	Perform a fuzzy search using the Hamming distance algorithm.
EDIT_DISTANCE	Perform a fuzzy search using the edit distance (Levenshtein) algorithm.
DATE	Search for a date or a range of dates.
TIME	Search for a time or a range of times.
NUMBER	Search for a number or a range of numbers.
CURRENCY	Search for a monetary value or a range of monetary values.
IPV4	Search for an IPv4 address or a range of IPv4 addresses.
IPV6	Search for an IPv6 address or a range of IPv6 addresses.

Each primitive is covered later in this chapter.

In addition, most of the primitives has aliases that may be used as a replacement. Details are available in each primitive's section. Here is a quick summary of each primitive and its aliases:

Primitive	Aliases
Exact Search / EXACT	**EXACT** ES
Fuzzy Hamming Search / HAMMING	**HAMMING** FHS
Fuzzy Edit Distance (Levenshtein) / EDIT_DISTANCE	**EDIT_DISTANCE** EDIT_DIST EDIT FEDS
Date / DATE	**DATE**
Time / TIME	**TIME**
Number / NUMBER	**NUMBER** NUMERIC
Currency / CURRENCY	**CURRENCY** MONEY
IPv4 / IPV4	**IPV4**
IPv6 / IPV6	**IPV6**

Expression

The `expression` specifies the expression to be matched. The possible values are:

Specifier	Description
Quoted String	Any valid C language string, including backslash-escaped characters. For example, look at the string "`match this text\n`. This can also include escaped hexadecimal characters, such as: • `match this text\x0A`, or • `\x48\x69\x20\x54\x68\x65\x72\x65\x21\x0A\x00`. • If a backslash needs to be placed in the quoted string for search query purposes, use the double backsplash escape sequence `"\\"` so that it is escaped properly.
Wildcard	A "?" character is used to denote that any single character will match. It can be inserted at any point(s) between quoted strings. For example, `"match th"?"s text\n"`.
Combination	Any combination of the above. For example, `"match\x20th"?"s text\x0A"`, or `"match\x20with a wildcard right here"?"and a null at the end\x00"`.

Options

The `options` specify a comma-separated list of options that can further qualify the request for certain primitives; the possible values are specific for each search type.

NOTE: You can include valid but extraneous options, but they will be ignored. For example, if a `DISTANCE` option is specified with an `EXACT` primitive, the `DISTANCE` option is ignored because it doesn't apply to an exact search, but the search will still run.

Some options are supported by each search primitive, such as `WIDTH` or `LINE`. Some are used internally by the Ryft REST server, such as `FILTER` as it applies to searching a catalog.

Here is a matrix of the available options and the primitives that use the options:

OPTION	Primitive								
	ES	FHS	FEDS	DATE	TIME	NUMBER	CURRENCY	IPV4	IPV6
WIDTH	X	X	X	X	X	X	X	X	X
LINE	X	X	X	X	X	X	X	X	X
FILTER	X	X	X	X	X	X	X	X	X
CASE	X	X	X						
DISTANCE		X	X						
REDUCE			X						
SEPARATOR						X	X		
DECIMAL						X	X		
SYMBOL							X		
OCTAL								X	

Examples of each option and its associated aliases are provided in the following sections.

WIDTH Option

✓	EXACT	✓	EDIT_DISTANCE	✓	TIME	✓	CURRENCY	✓	IPV6
✓	HAMMING	✓	DATE	✓	NUMBER	✓	IPV4		

The `WIDTH` option specifies a surrounding width as an integer value, where surrounding width means that results will contain the specified number of characters (value) to the left and to the right of the matching string/value. By default, `WIDTH=0`, unless you specify a value.

If the format of the input data is specified as raw text, the `surrounding` parameter specifies how many characters around the match will be returned as part of the matched data results. The `surrounding` width can be useful to assist a human analyst or a downstream machine learning tool to determine the contextual use of a specific matched term when searching unstructured raw text data.

If the input data is record-based, the surrounding width option is ignored.

NOTE: `WIDTH` and `LINE` are mutually exclusive. The option that is listed last affects the query. This behavior differs from `ryftprim` which gives an error if both are specified in the same query.

Aliases

You can use these aliases to specify the width. The example queries have the same effect:

Option/Alias	Example
WIDTH	(RAW_TEXT CONTAINS <primitive>("orange", WIDTH=5))
SURROUNDING	(RAW_TEXT CONTAINS <primitive>("orange", SURROUNDING=5))
W	(RAW_TEXT CONTAINS <primitive>("orange", W=5))

* replace <primitive> with a specific primitive like EXACT or HAMMING>

LINE Option

✓	EXACT	✓	EDIT_DISTANCE	✓	TIME	✓	CURRENCY	✓	IPV6
✓	HAMMING	✓	DATE	✓	NUMBER	✓	IPV4		

The `LINE=true` option enables you to return the line-feed delimited line on which the match occurs. This is useful when processing comma-separate value (CSV) files to return the entire CSV record on which the match is found.

By default, `LINE=false`.

Different results occur depending on where the match is found:

- If a match appears multiple times on a single line, only one match line is generated.
- If a nearby line feed cannot be found within a distance of one chunk on either side of the chunk containing the match, then the data results are undefined, and become implementation-specific.

The system defaults to use `WIDTH` mode, and `LINE` mode may be enabled on a per-query basis. If the input data is record-based, the line option is ignored

NOTE: `WIDTH` and `LINE` are mutually exclusive. The option that is listed last affects the query. This behavior differs from `ryftprim` which gives an error if both are specified in the same query.

Aliases

You can use these aliases to specify line return. The example queries have the same effect:

Option/Alias	Example
LINE	(RAW_TEXT CONTAINS <primitive>("orange", LINE=5))
L	(RAW_TEXT CONTAINS <primitive>("orange", L=5))

Please see Option Types to get the ways the `LINE` option can be set.

FILTER Option

✓	EXACT	✓	EDIT_DISTANCE	✓	TIME	✓	CURRENCY	✓	IPV6
✓	HAMMING	✓	DATE	✓	NUMBER	✓	IPV4		

The `FILTER` option specifies a regular expression as a string value. This option is only used with a catalog, by filtering the resulting catalog file parts, by name. By default, `FILTER=""` is used, which means that all file parts are relevant.

For example, if the catalog contains many "*.txt" and "*.bin" binary file parts, then you can narrow down the results by using the corresponding regular expression:

- `FILTER=".*\.txt"` – for text files
- `FILTER=".*\.bin"` – for binary files

Any regular expression can be used to specify complex filename filtering rules, such as date ranging.

Aliases

You can use these aliases to specify a filter. The example queries have the same effect:

Primitive/Alias	Example
FILTER	`(RECORD CONTAINS EXACT("orange", FILE_FILTER=".*txt"))`
FILE_FILTER	`(RECORD CONTAINS EXACT("orange", FILTER=".*txt"))`
FF	`(RECORD CONTAINS EXACT("orange", FF=".*txt"))`

CASE Option

✓	EXACT	✓	EDIT_DISTANCE		TIME		CURRENCY		IPV6
✓	HAMMING		DATE		NUMBER		IPV4		

When `CASE=false`, the query runs as case insensitive. By default, `CASE=true`, so searches are case sensitive.

Aliases

You can use these aliases to specify case sensitivity. The example queries have the same effect:

Option/Alias	Example
CASE	`(RECORD CONTAINS <primitive>("orange", CASE=false))`
CS	`(RECORD CONTAINS <primitive>("orange", CS=false))`

Please see Option Types to get the ways the CASE option can be set.

DISTANCE Option

	EXACT	✓	EDIT_DISTANCE		TIME		CURRENCY		IPV6
✓	HAMMING		DATE		NUMBER		IPV4		

DISTANCE specifies the fuzzy search distance as an integer value. By default, DISTANCE=0.

A match is determined if the distance between the match and the search expression is less than or equal to this DISTANCE option.

For unstructured RAW_TEXT queries, the resulting distance value for each match encountered is written to the output index file, if specified, which allows for downstream analytics tools to know how close the match was to the request.

For RECORD based searching (where input_specifier is RECORD or RECORD.<field_name>), the distance is not reported in the output index file, since the purpose of RECORD based searching is to return the entire matching RECORD which can often have more than one internal match, each of which could have a different distance.

Aliases

You can use these aliases to specify distance. The example queries have the same effect:

Option/Alias	Example
DISTANCE	(RECORD CONTAINS <primitive>("albatross ", DISTANCE=1))
FUZZINESS	(RECORD CONTAINS <primitive>("albatross ", FUZZINESS=1))
DIST	(RECORD CONTAINS <primitive>("albatross ", DIST=1))
D	(RECORD CONTAINS <primitive>("albatross ", D=1))

REDUCE Option

	EXACT	✓	EDIT_DISTANCE		TIME		CURRENCY		IPV6
	HAMMING		DATE		NUMBER		IPV4		

REDUCE determines whether the results will be reduced, thereby eliminating duplicate matches. It can be set to true or false. Default is REDUCE=false.

Duplicate matches are common given how the edit distance's Levenshtein algorithm works, where it counts insertions, deletions and replacements when calculating fuzziness for the match. For example, if you are searching for "giraffe" with distance less than or equal to one, and the input corpus is "That giraffe is tall," then three results would be generated:

1. " giraffe" - inserts one space before the word, so that is a match with distance one.
2. "giraffe" – an exact match with is a distance of zero (less than one)
3. "iraffe" – missing the leading "g" which is a distance of one.

All three are reported. With REDUCE=true, only one match will be reported: "giraffe".

Aliases

You can use these aliases to specify reduce. The example queries have the same effect:

Option/Alias	Example
REDUCE	(RECORD CONTAINS <primitive>("albatross ", D=1, REDUCE=true))
D	(RECORD CONTAINS <primitive>("albatross ", D=1, R=true))

Please see Option Types to get the ways the REDUCE option can be set.

SEPARATOR Option

	EXACT		EDIT_DISTANCE		TIME	✓	CURRENCY		IPV6
	HAMMING		DATE	✓	NUMBER		IPV4		

The required SEPARATOR is defined as the separating character to use. For example, for standard US numbers, a comma is specified for use between the thousand and hundred place. If other types of numbers are being searched, such as perhaps phone numbers, then a dash would be specified.

Note the following:

- The SEPARATOR does not need to appear in the data stream, but if it does appear, it will be considered part of the value being parsed.
- The SEPARATOR and DECIMAL must be different characters. If the same character is specified for both, an error message will be generated. The SEPARATOR="," is used by default, i.e. if nothing is provided. This behavior differs from ryftprim; there is no default so if it's not specified, then it errors out.

Aliases

You can use these aliases to specify the separator. The example queries have the same effect:

Option/Alias	Example
SEPARATOR	(RECORD.id CONTAINS NUMBER(1025<NUM<1050, SEPARATOR=",", DECIMAL="."))
SEP	(RECORD.id CONTAINS NUMBER(1025<NUM<1050, SEP=",", DECIMAL="."))

DECIMAL Option

	EXACT		EDIT_DISTANCE		TIME	✓	CURRENCY		IPV6
	HAMMING		DATE	✓	NUMBER		IPV4		

The required DECIMAL is defined as the character to use for the decimal specifier. For example, for standard US numbers, a period (decimal point) would be specified for use.

Note the following:

- The `DECIMAL` does not need to appear in the data stream, but if it does appear, it will be considered part of the value being parsed.
- The `DECIMAL` and `SEPARATOR` must be different. If the same character is specified for both, an error message will be generated. The `DECIMAL="."` is used by default, i.e. if nothing is provided. This behavior differs from `ryftprim`; there is no default so if it's not specified, then it errors out.

Aliases

You can use these aliases to specify the separator. The example queries have the same effect:

Option/Alias	Example
DECIMAL	`(RECORD.id CONTAINS NUMBER(1025<NUM<1050, SEP=",", DECIMAL="."))`
DEC	`(RECORD.id CONTAINS NUMBER(1025<NUM<1050, SEP=",", DEC="."))`

SYMBOL Option

	EXACT		EDIT_DISTANCE		TIME	✓	CURRENCY		IPV6
	HAMMING		DATE		NUMBER		IPV4		

The required SYMBOL is defined as the monetary symbol to be used in the state machine which starts a currency match state machine. For example, for standard US currency, a dollar sign would be specified as "$". The SYMBOL="$" is used by default, i.e. if nothing is provided. This behavior differs from `ryftprim`; there is no default so if it's not specified, then it errors out.

Aliases

You can use these aliases to specify the symbol. The example queries have the same effect:

Option/Alias	Example
SYMBOL	`(RECORD.price CONTAINS CURRENCY(450<CUR, SYMBOL="$", SEP=",", DEC="."))`
SYMB	`(RECORD.price CONTAINS CURRENCY(450<CUR, SYMB="$", SEP=",", DEC="."))`

Option/Alias	Example
SYM	`(RECORD.price CONTAINS CURRENCY(450<CUR, SYM="$", SEP=",", DEC="."))`

OCTAL Option

	EXACT		EDIT_DISTANCE		TIME		CURRENCY		IPV6
	HAMMING		DATE		NUMBER	✓	IPV4		

The default behavior of the IPv4 search primitive is to parse all octets encountered as a decimal. When option `OCTAL=true` is specified, any octets encountered with leading 0's will be parsed as octal quantities instead of decimal quantities, for comparison purposes. `OCTAL=0` is used by default.

This can be important in certain cyber analysis scenarios, where obfuscation attempts using octal notation can sometimes fool contemporary systems into decoding incorrect IP addresses if they do not properly parse octal quantities. This is also useful when it is known that certain input datasets may not be strictly adhering to a certain type of leading zero parsing rule.

Aliases

You can use these aliases to specify the octal. The example queries have the same effect:

Option/Alias	Example
USE_OCTAL	`(RAW_TEXT CONTAINS IPV4(IP > "10.11.12.13", USE_OCTAL))`
OCTAL	`(RAW_TEXT CONTAINS IPV4(IP > "10.11.12.13", OCTAL=true))`
OCT	`(RAW_TEXT CONTAINS IPV4(IP > "10.11.12.13", OCT=true))`

Option Types

There are three option types:

- **Integer** – Value can be in quotes or not, like `W=3`, or `W="50"`
- **String** – Value must be in quotes, like `SYMBOL="$"`
- **Boolean** – Values are flexible:
  - Values can be parsed. For example:
    - `true`: "1", "t", "T", "TRUE", and "True"
    - `false`: "0", "f", "F", "FALSE", and "False"
  - Double quotes are optional.
    For example: `L=true`, `L=1`, `L="T"`, and `L=False`
  - Can be defined as just names with optional negation.
    For example: `L, !CS` means `L=true, CS=false`

Logical Operator

The `logical_operator` allows for complex collections of relational expressions. The possible values are:

- **AND** - The logical expression (`a AND b`) evaluates to true only if both relational expressions "a" and "b" evaluate to true.
- **OR** - The logical expression (`a OR b`) evaluates to true if either the relational expression "a" or "b" evaluate.
- **XOR** - The logical expression (`a XOR b`) evaluates to true if either the relational expression "a" evaluates to true or the relational expression "b" evaluates to true, but not both.

Multiple relational expressions can be combined using the logical operators. For example:

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

There are additional ways to combine and control the precedence of operations, that conform to the query optimization rules, with a few exceptions.

Query Optimization Rules

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

- FEDS sub-queries are never combined for RECORD or RAW_TEXT searches.
- 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))`
  - Search type based exception list can be configured in the "ryft-server.conf" configuration file, using the `optimizer-do-not-combine` option, like this:
    `(RECORD.text CONTAINS FEDS("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**: Do NOT combine any RAW_TEXT-based sub-queries that use the "AND" logical operator; it can never be combined into one command call.

Here is a sample "/etc/ryft-server.conf" file that contains the ryft-server options set on the RyftONE. These can be customized on the server, but can also be individually overridden on the command line by using specific command line options.

```
### this configuration file contains ryft-server options.
### most of the options may be overriden by corresponding
### command line option (noted in parenthesis).

### main search engine and its options
search-backend: ryftprim
backend-options:
  ryftprim-legacy: true
#  instance-name: .rest-8765         # server's working directory (inside /ryftone)
  ryftprim-exec: /usr/bin/ryftprim
```

```
  ryftone-mount: /ryftone
  minimize-latency: false           # false - wait until ryftprim is finished before
start data processing, true - start immediately once ryftprim is launched
  open-poll: 100ms                  # open INDEX&DATA file poll timeout
  read-poll: 100ms                  # read INDEX&DATA file poll timeout
  read-limit: 1000                  # maximum number of read attempts

### query decomposition:
#  compat-mode: true                 # true - compatibility mode, false - generic
#  optimizer-limit: 10               # maximum number of subqueries to combine, 0
means do not combine at all, -1 means combine all (no limit)
  optimizer-do-not-combine: feds    # comma-separated search modes that should not be
combined
```

Precedence of Operations

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

Parenthesis "( )"

Parentheses can be used to control the precedence of operations. Additional whitespaces are allowable in your query string, 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 "{ }"

The curly braces can be used to break query optimization rules. All queries in the curly braces are combined into just one Ryft call. Curly braces can be used for manual optimization.

Here is an example of a query with two curly braces:

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

Ryft would split it into two Ryft 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 Ryft 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 Ryft 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 Ryft call.

Exact Search

The exact search operation will search `existing_data_set` using a match criterion. The match string can be up to 32 bytes in length. Only exact matches will be returned.

Exact searches extend the general relational expression defined, previously as follows:

```
(input_specifier relational_operator EXACT(expression[, options]))
```

A fully qualified EXACT clause looking for the term "orange" would look like this:

```
(RECORD CONTAINS EXACT("orange"))
```

Aliases

You can use these aliases to specify an exact search. The example queries have the same effect:

Primitive/Alias	Example
EXACT	`(RECORD CONTAINS EXACT("orange"))`
ES	`(RECORD CONTAINS ES("orange"))`

Compatible Syntax

For backwards compatibility, the term EXACT can be omitted:

```
(input_specifier relational_operator expression)
```

NOTE: There is no way to specify additional options in this syntax! The global options are used.

In addition, the syntax can be further simplified, whereby the standalone `expression` is converted to `(RAW_TEXT CONTAINS expression)`. For example, if the original exact search specified a match of "Apple" OR "Orange" it will be converted to

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

Options

There are four comma-separated options for the EXACT primitive:

- WIDTH
- LINE
- CASE
- FILTER

All options are covered in “Options” section on page 51.

Fuzzy Hamming Search

The Fuzzy Hamming search operation works similarly to exact search except that matches do not have to be exact. Instead, the DISTANCE option (the fuzziness parameter) allows the specification of a "close enough" value to indicate how close the input must be to match the search criteria. The match string can be up to 32 bytes in length.

A "close enough" match is specified as a Hamming distance. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. As provided to the fuzzy search operation, the Hamming distance specifies the maximum number of substitutions that can declare a match.

In addition, like exact search, the surrounding mechanism can aid in downstream analysis of contextual use of the fuzzy match results against unstructured raw text data.

Hamming searches extend the general relational expression defined previously as follows:

```
(input_specifier relational_operator HAMMING(expression, options))
```

A fully qualified HAMMING clause looking for the term "albatross" and return matching lines using a distance of 1, would be:

```
(RAW_TEXT CONTAINS HAMMING("albatross", DISTANCE="1", LINE="true"))
```

Aliases

You can use these aliases to specify HAMMING fuzziness on a fuzzy hamming search. The example queries have the same effect:

Primitive/Alias	Example
HAMMING	(RAW_TEXT CONTAINS HAMMING("albatross", DISTANCE="1", LINE="true"))
FHS	(RAW_TEXT CONTAINS FHS("albatross", DISTANCE="1", LINE="true"))

Hamming search algorithms are highly parallelizable, so they can run extremely quickly on accelerated hardware. Therefore, if search algorithms can make use of Hamming search, then they will typically perform at very high speed, often at the same speed as exact searches.

Fuzzy Hamming search algorithms are highly parallelizable, so they can run extremely quickly on accelerated hardware. Therefore, if fuzzy search algorithms can make use of fuzzy Hamming search, then they will typically perform at very high speed.

This means that if the search string is “united states” with a fuzziness of 2, then any match must first match the string length (13 characters) and up to 2 characters in the string may be different.

Options

There are four comma-separated options for the EXACT primitive:

- DISTANCE
- WIDTH
- LINE
- CASE
- FILTER

Note that the DISTANCE option is required, as it sets the Hamming distance that will be used.
If DISTANCE is specified as zero the search mode will be automatically changed to EXACT.

All options are covered in “Options” section on page 51.

Fuzzy Edit Distance search

Fuzzy Edit Distance Search performs a search that does not require two strings to be of equal length to obtain a match. Instead of considering individual symbol differences, fuzzy edit distance search counts the minimum number of insertions, deletions and replacements required to transform one string into another. This can make it much more powerful than Fuzzy Hamming search for certain applications.

Let’s compare searching for the string “Michelle” using Fuzzy Hamming vs. Fuzzy Edit Distance, with an edit distance = 1.

String	Fuzzy Hamming, Edit Distance = 1	Fuzzy Edit Distance, Edit Distance = 1
Michelle	Yes. Exact match.	Yes. Exact match.
Mishelle	Yes. “c” changed to “s”.	Yes. “c” changed to “s”.
Mischelle	No. The string “chelle” does not appear in the same position as in the original search term “Michelle”. This makes it an edit distance of 6, which is greater than the specified distance.	Yes. Can insert the single character “s”.
Michele	No. Deleting one “l” shortens the string by one character, and the match string must be of equal length.	Yes. One “l” deleted.
Mischele	No. Although of equal length, more than one change is required to match.	No. Requires 2 changes: add an “s” and remove an “l”.

String	Fuzzy Hamming, Edit Distance = 1	Fuzzy Edit Distance, Edit Distance = 1
		NOTE: If the edit distance = 2, then it would match since the calculated edit distance between the 2 strings is less than or equal to the desired edit distance (2).

Fuzzy edit distance is an extremely powerful search tool for a variety of data sources, including names, addresses, medical records searching, genomic and disease research data, common misspellings, and more. Unlike fuzzy Hamming search, fuzzy edit distance is a more natural fuzzy search paradigm for many algorithms, since it does not require string matches to be of the same size.

The tradeoff is that fuzzy edit distance is not as amenable to full hardware parallelization, so algorithms reliant on it typically run slower than those that implement fuzzy Hamming search. In addition, the match string length added to the distance value must not exceed 32 bytes.

Edit distance searches extend the general relational expression defined previously as follows:

```
(input_specifier relational_operator EDIT_DISTANCE(expression, options))
```

A fully qualified EDIT_DISTANCE clause looking for the term "albatross" and return 20 bytes on each side of every match encountered, while using a distance of 1 looks like this:

```
(RAW_TEXT CONTAINS EDIT_DISTANCE("albatross", DISTANCE="1", WIDTH="20"))
```

Aliases

You can use these aliases to specify EDIT_DISTANCE on a fuzzy edit distance (Levenshtein) search. The example queries have the same effect:

Primitive/Alias	Example
EDIT_DISTANCE	(RAW_TEXT CONTAINS EDIT_DISTANCE("albatross", DIST="1", W=20))
EDIT_DIST	(RAW_TEXT CONTAINS EDIT_DIST("albatross", DIST="1", W=20))
EDIT	(RAW_TEXT CONTAINS EDIT ("albatross", DISTANCE="1", W=20))
FEDS	(RAW_TEXT CONTAINS FEDS("albatross", DISTANCE="1", W=20))

Options

There are six comma-separated options for the EDIT_DISTANCE primitive:

- DISTANCE
- REDUCE
- WIDTH
- LINE
- CASE
- FILTER

Note that the DISTANCE option is required, as it sets the Hamming distance that will be used.
If DISTANCE="0" the search mode is automatically changed to EXACT.

All options are covered in "Options" section on page 51.

Date Search

The Date Search operation allows for exact date searches and for searching for dates within a range, for both structured and unstructured data, using the following date formats:

- YYYY/MM/DD
- YY/MM/DD
- DD/MM/YYYY
- DD/MM/YY
- MM/DD/YYYY
- MM/DD/YY

NOTE: The "/" character in the above list of formats can be replaced by any other single character delimiter. For example, YYYY-MM-DD and MM_DD_YYYY are both acceptable date formats.

Date searches extend the general relational expression defined previously, as follows:

```
(input_specifier relational_operator DATE(expression))
```

Different date ranges can be searched for by modifying the expression provided. There are two general expression types supported:

- DATE(DateFormat operator ValueB)
- DATE(ValueA operator DateFormat operator ValueB)

This is a full list of the supported expressions. DateFormat represents the format of the dates to search for and ValueA and ValueB represent dates to compare input data against.

- DateFormat = ValueB
- DateFormat != ValueB (Not equals operator)
- DateFormat >= ValueB
- DateFormat > ValueB
- DateFormat <= ValueB
- DateFormat < ValueB
- ValueA <= DateFormat <= ValueB
- ValueA < DateFormat < ValueB
- ValueA < DateFormat <= ValueB
- ValueA <= DateFormat < ValueB

For example, to find all dates after "02/28/12" in unstructured raw text, use the following search query criteria:

```
(RAW_TEXT CONTAINS DATE(MM/DD/YY > 02/28/12))
```

To find all matching dates between "02/28/12" and "01/19/15" but not including those two dates, in a record/field construct, where the field tag is date, use this:

```
(RECORD.date CONTAINS DATE(02/28/12 < MM/DD/YY < 01/19/15))
```

Please note, the ValueA and ValueB can be quoted or unquoted. In addition to ryftprim the following formats are supported:

- `DateFormat == ValueB`
- `ValueA >= DateFormat >= ValueB`
- `ValueA > DateFormat > ValueB`
- `ValueA > DateFormat >= ValueB`
- `ValueA >= DateFormat > ValueB`

Options

There are three comma-separated options for the DATE primitive:

- WIDTH
- LINE
- FILTER

All options are covered in “Options” section on page 51.

Time Search

Like DATE search, the TIME search can be used to search for exact times or for times within a particular range, for both structured and unstructured data, using the following time formats:

- `HH:MM:SS`
- `HH:MM:SS:ss`

NOTE: The `'ss'` in the second format indicates hundredths of a second, and if specified should always be two digits.

Time searches extend the general relational expression defined previously as follows:

```
(input_specifier relational_operator TIME(expression))
```

Similar to the Date Search, different time ranges can be searched for by modifying the expression in the relational expression above. Again, there are two general expression types supported:

- `TIME(TimeFormat operator ValueB)`
- `TIME(ValueA operator TimeFormat operator ValueB)`

This is a list of the supported expressions. `TimeFormat` represents the format of the times to search for and `ValueA` and `ValueB` represent times to compare input data against.

- `TimeFormat = ValueB`
- `TimeFormat != ValueB` (Not equals operator)
- `TimeFormat >= ValueB`

- `TimeFormat > ValueB`
- `TimeFormat <= ValueB`
- `TimeFormat < ValueB`
- `ValueA <= TimeFormat <= ValueB`
- `ValueA < TimeFormat < ValueB`
- `ValueA < TimeFormat <= ValueB`
- `ValueA <= TimeFormat < ValueB`

For example, to find all times after "09:15:00", use the following search query criteria:

```
(RAW_TEXT CONTAINS TIME(HH:MM:SS > 09:15:00))
```

To find all matching times between "11:15:00" and "13:15:00" but not including those times in a record/field construct where the field tag is time, use:

```
(RECORD.time CONTAINS TIME(11:15:00 < HH:MM:SS < 13:15:00))
```

Please note, the ValueA and ValueB can be quoted or unquoted. In addition to `ryftprim` the following formats are supported:

- `TimeFormat == ValueB`
- `ValueA >= TimeFormat >= ValueB`
- `ValueA > TimeFormat > ValueB`
- `ValueA > TimeFormat >= ValueB`
- `ValueA >= TimeFormat > ValueB`

Options

There are three comma-separated options for the `TIME` primitive:

- `WIDTH`
- `LINE`
- `FILTER`

All options are covered in “Options” section on page 51.

Number Search

The Number Search operation can be used to search for exact numbers or numbers in a particular range for both structured and unstructured input data.

The protocol rules are:

- No line breaks anywhere.
- The maximum number of characters is 64.
- Whitespace is not permitted within the number.
- The maximum number of consecutive digits is 16 base-10 characters. This means that numeric accuracy is compromised on numbers outside the range of approximately -253 and +253.
- The exponent value ranges when scientific notation is used are -512 and +512.

- Numeric separators are not permitted within a specified exponent.
- Infinity and "not a number" (NaN) are not accepted as numbers.

When a series of characters violates these parsing rules, the result will be the last valid sequence within the series, which allows for partial matches, and can be an excellent tool for analyzing potentially dirty data.

This general solution allows for numbers to be represented in arbitrary forms, including:

- Configurable separators, such as perhaps specifying a comma representing a thousands separator for the US number system (e.g., "7,000"), or specifying a dash separating fields in a phone number or a social security number (e.g., "1-800-555-1212" or "123-45-6789").
- Configurable decimals, such as perhaps specifying a period to represent the decimal for the US number system (e.g., "7.2").
- A minus sign to specify a negative value, such as "-7.2".
- Scientific notation, such as "-2e3", "-2e-3", "-2.2E+2", etc.
- Data results returned for a particular number matching specified criteria will be truncated after a total of 64 characters.

Number Searches extend the general relational expression defined previously as follows:

```
(input_specifier relational_operator NUMBER(expression, separator, decimal))
```

Different number ranges can be searched for by modifying the expression provided. There are two general expression types supported:

- `NUM operator1 "ValueA"`
- `"ValueA" operator1 NUM operator2 "ValueB"`

This is a full list of the supported expressions. `ValueA` and `ValueB` represent the numbers to compare the input data against.

- `NUM = "ValueA"`
- `NUM != "ValueA"` (Not equals operator)
- `NUM >= "ValueA"`
- `NUM > "ValueA"`
- `NUM <= "ValueA"`
- `NUM < "ValueA"`
- `"ValueA" <= NUM <= "ValueB"``
- `"ValueA" < NUM < "ValueB"`
- `"ValueA" < NUM <= "ValueB"`
- `"ValueA" <= NUM < "ValueB"`

The separator is defined as the separating character to use. For example, for standard US numbers, a comma would be specified. If other types of numbers are being searched, such as perhaps phone numbers, then a dash would be specified.

The decimal is defined as the decimal specifier to use. For example, for standard US numbers, a period would be specified.

Note that the separator and decimal must be different. If the same character is specified for both, an error message will be generated.

Please note, the ValueA and ValueB can be quoted or unquoted. In addition to ryftprim the following formats are supported:

- `NUM == "ValueA"`
- `ValueA >= NUM >= ValueB`
- `ValueA > NUM > ValueB`
- `ValueA > NUM >= ValueB`
- `ValueA >= NUM > ValueB`

As an example of a fully qualified number search relational expression, to find all matching numbers using the US number system between but not including "1025" and "1050" in a record/field construct where the field tag is id, use:

```
(RECORD.id CONTAINS NUMBER("1025" < NUM < "1050", ",", "."))
```

The results will contain all numbers that are encountered in the input data that match the specified range. For example, if a scientific notation number "1.026e3" appears in the input data (which expands to "1,026"), then it will be reported as a match since it falls within the specified range. Similarly, the numbers 1049 and 1,049.9 would also match. However, the number -1,234 would not be a match, as it does not fall within the requested range, nor would the number +1,050.00001.

Aliases

You can use these aliases to specify NUMBER. The example queries have the same effect:

Primitive/Alias	Example
NUMBER	`(RECORD.id CONTAINS NUMBER("1025" < NUM < "1050", SEPARATOR=",", DECIMAL="."))`
NUMERIC	`(RECORD.id CONTAINS NUMERIC("1025" < NUM < "1050", SEPARATOR=",", DECIMAL="."))`

Options

There are six comma-separated options for the EDIT_DISTANCE primitive:

- SEPARATOR
- DECIMAL
- WIDTH
- LINE
- FILTER

All options are covered in “Options” section on page 51.

Currency Search

The Currency Search operation follows largely the same rules as the Number Search operation. The parsing protocol diagram changes slightly to allow for configurable currency identifiers, such as "$" for US currency. In addition, for currency, negative amounts are parsed using either the minus sign or parenthetical "()" notation.

Currency Searches are a type of number searches and extend the general relational expression defined previously as follows:

```
(input_specifier relational_operator CURRENCY(expression, currency, separator,
decimal))
```

Different currency ranges can be searched for by modifying the expression provided. There are two general expression types supported:

- `CUR operator1 "ValueA"`
- `"ValueA" operator1 CUR operator2 "ValueB"`

This is a list of the supported expressions. `ValueA` and `ValueB` represent the currency values to compare the input data against.

- `CUR = "ValueA"`
- `CUR != "ValueA"` (Not equals operator)
- `CUR >= "ValueA"`
- `CUR > "ValueA"`
- `CUR <= "ValueA"`
- `CUR < "ValueA"`
- `"ValueA" <= CUR <= "ValueB"`
- `"ValueA" < CUR < "ValueB"`
- `"ValueA" < CUR <= "ValueB"`
- `"ValueA" <= CUR < "ValueB"`

For currency searches, the separator is defined as the optional separating character that may be encountered when parsing currency. For example, for standard US currency, a comma would be specified. If other types of currencies are being searched, such as perhaps certain types of European currencies that use a period as the separator, a period would be specified.

The decimal is defined as the decimal specifier to use. For example, for standard US numbers, a period would be specified. For other currency types, an appropriate character would be specified.

Note that the separator and decimal specifier must be different. If the same character is specified for both, an error message will be generated.

As an example of a fully qualified currency search relational expression, to find all values using the US currency system between but not including "$450" and "$10,100.50" in a record/field construct where the field tag is price, use:

```
(RECORD.price CONTAINS CURRENCY("$450" < CUR < "$10,100.50", "$", ",", "."))
```

The results will contain all prices encountered in the input data that match the specified range. For example, if a price "$692.01" appears in the input data, then it will be reported as a match since it falls within the specified range. But currency values like -$123.00, $449.99 and $100,000 would not match.

Please note, the ValueA and ValueB can be quoted or unquoted. In addition to ryftprim the following formats are supported:

- `CUR == "ValueA"`
- `ValueA >= CUR >= ValueB`
- `ValueA > CUR > ValueB`
- `ValueA > CUR >= ValueB`
- `ValueA >= CUR > ValueB`

Aliases

You can use these aliases to specify NUMBER. The example queries have the same effect:

Primitive/Alias	Example
CURRENCY	`(RECORD.price CONTAINS CURRENCY("$450" < CUR < "$10,100.50", SYMBOL="$", SEPARATOR=",", DECIMAL="."))`
MONEY	`(RECORD.price CONTAINS MONEY("$450" < CUR < "$10,100.50", SYMBOL="$", SEPARATOR=",", DECIMAL="."))`

Options

There are six comma-separated options for the EDIT_DISTANCE primitive:

- SEPARATOR
- DECIMAL
- SYMBOL
- WIDTH
- LINE
- FILTER

All options are covered in “Options” section on page 51.

IPv4 Search

The IPv4 Search operation can be used to search for exact IPv4 addresses or IPv4 addresses in a particular range, in both structured and unstructured text, using the standard "a.b.c.d" format for IPv4 addresses.

IPv4 Search extends the general relational expression defined previously as follows:

```
(input_specifier relational_operator IPV4(expression))
```

Different ranges can be searched for by modifying the expression in the relational expression above. There are two general expression types supported:

- `IP operator "ValueB"`
- `"ValueA" operator IP operator "ValueB"`

In addition, the following formats are supported:

- `IP == "ValueB"`
- `ValueA >= IP >= ValueB`
- `ValueA > IP > ValueB`
- `ValueA > IP >= ValueB`
- `ValueA >= IP > ValueB`

This is a list of supported expressions. `ValueA` and `ValueB` represent the IP addresses to compare the input data against.

- `IP = "ValueB"`
- `IP != "ValueB"` (Not equals operator)
- `IP >= "ValueB"`
- `IP > "ValueB"`
- `IP <= "ValueB"`
- `IP < "ValueB"`
- `"ValueA" <= IP <= "ValueB"`
- `"ValueA" < IP < "ValueB"`
- `"ValueA" < IP <= "ValueB"`
- `"ValueA" <= IP < "ValueB"`

For example, to find all IP addresses greater than 10.11.12.13, use the following search query criteria:

```
(RAW_TEXT CONTAINS IPV4(IP > "10.11.12.13"))
```

To find all matching IPv4 addresses adhering to 10.10.0.0/16 (that is, all IP addresses from 10.10.0.0 through 10.10.255.255 inclusive) in a record/field construct where the field tag is ipaddr, use:

```
(RECORD.ipaddr CONTAINS IPV4("10.10.0.0" <= IP <= "10.10.255.255"))
```

Options

There are four comma-separated options for the `IPv4` primitive:

- `OCTAL`
- `WIDTH`
- `LINE`
- `FILTER`

All options are covered in “Options” section on page 51.

IPv6 Search

The IPv6 Search operation can be used to search for exact IPv6 addresses or IPv6 addresses in a particular range in both structured and unstructured text using the standard “a:b:c:d:e:f:g:h” format for IPv6 addresses. The double colon (::) is also supported, per RFC guidelines.

IPv6 searches extend the general relational expression defined previously, as follows:

```
(input_specifier relational_operator IPV6(expression))
```

Different ranges can be searched for by modifying the expression in the relational expression above. There are two general expression types supported:

- `IP operator "ValueB"`
- `"ValueA" operator IP operator "ValueB"`

This is list of supported expressions. `ValueA` and `ValueB` represent the IP addresses to compare the input data against.

- `IP = "ValueB"`
- `IP != "ValueB"` (Not equals operator)
- `IP >= "ValueB"`
- `IP > "ValueB"`
- `IP <= "ValueB"`
- `IP < "ValueB"`
- `"ValueA" <= IP <= "ValueB"`
- `"ValueA" < IP < "ValueB"`
- `"ValueA" < IP <= "ValueB"`
- `"ValueA" <= IP < "ValueB"`

In addition, the following formats are supported:

- `IP == "ValueB"`
- `ValueA >= IP >= ValueB`
- `ValueA > IP > ValueB`
- `ValueA > IP >= ValueB`
- `ValueA >= IP > ValueB`

For example, to find all IP addresses greater than `1abc:2::8`, use the following search query criteria:

```
(RAW_TEXT CONTAINS IPV6(IP > "1abc:2::8"))
```

To find all matching IPv6 addresses between “10::1” and “10::1:1”, inclusive, in a record/field construct where the field tag is “ipaddr6” use this:

```
(RECORD.ipaddr6 CONTAINS IPV6("10::1" <= IP <= "10::1:1"))
```

Options

There are three comma-separated options for the `IPv6` primitive:

- `WIDTH`
- `LINE`
- `FILTER`

All options are covered in “Options” section on page 51.

4

4: Command Line Tool

The `ryftrest` command line tool is a simple bash script with syntax that is very similar to the native `ryftprim` tool. It uses the `ryft-server` as a backend. The install package places the `ryftrest` tool in the "/usr/bin" directory.

There are distinct advantages to using `ryftrest` instead of `ryftprim` at the command line. With the `ryftrest` tool, you can:

- Execute one or more subqueries, with different search expressions connected by logical operators, with one command line. It can be used for both unstructured and structured searches, and you can create complex search queries. For example, you can combine a fuzzy edit search with a date range and a time range in a single command.
- Seamlessly execute queries on a single server or a cluster of Ryft servers, thereby allowing for searches to be distributed across a cluster.
- Send requests to remote Ryft servers using the `--address` option on the command line.
- Execute queries on remote/distributed servers. The end user does not have to be logged on to the Ryft server directly.
- Change output to JSON, on the fly which is very useful when used in conjunction with jq, the JSON command line processor
- Translate simple pattern match queries to hex encoding, avoiding any possible escaping problems.
- Can print found data records.
- Can be used to perform requests that `ryftprim` does and more, such as combining primitives in one command line.

NOTE: The data files used in the example already have an associated record definition file (RDF) defined on the Ryft ONE.

Parameters

The parameters for `ryftrest` serve two purposes. One set is specific to performing searches, and the second set are REST specific. Both are covered in the following tables. Use the `ryftrest --help` on the command line to see the syntax:

Parameter	Type	Description
Search Specific Parameters		
`-c | --catalog=<path>`	String	Specify an input catalog name.
`-d | --fuzziness=<D>`	Integer	Specify the search distance Default is 0. Used with fuzzy hamming and edit distance.
`-e | --delimiter=<E>`	String	Specify the delimiter used between found records. For example, use Windows new line: `-e $'\r\n'`
`-f | --file=<path>`	String Required	Specify an input filename. More than one -f <filename> parameter can be used to specify multiple filenames.
`-help | --help`		Print the help message and detailed syntax.
`-i`	String	Specify case-insensitive analysis for supported primitives. Default is case-sensitive.
`--line`		Specify the surrounding whole line (same was `-w=line`)
`-n | --nodes=<N>`	Integer	Specify 1-4 RCAB processing nodes to use. Default uses all available nodes.
`--no-reduce`		Do not reduce the duplicates for FEDS.
`-od | --data=<file>`	String	Specify a data results file.
`-oi | --index=<file>`	String	Specify an index results file.
`-p | --mode=<mode>`	String	Specifies the search mode to run, which can be one of: • `exact_search` (or `es`); DEFAULT • `fuzzy_hamming_search` (or `fhs`) • `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`)
`-r | --reduce`		Reduce the duplicates for FEDS (by default).
`-s | -q=<query> | --query=<query>`	String	Required. Specify the search/query expression to use with `*_search` primitives. A detailed description of all possible modes and options is contained in the Ryft Open API Library User Guide..

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td colspan="3">Search Specific Parameters</td></tr>
<tr><td></td><td></td><td>Exact Search: exact_search
-s '(RAW_TEXT CONTAINS "Babe Ruth",CS=true)'

Fuzzy Hamming Search: fuzzy_hamming_search | fhs
-s '(RAW_TEXT CONTAINS FHS("Babe Ruth",CS=true,DIST=2,WIDTH=20))'

Fuzzy Edit Distance Search: fuzzy_edit_distance_search | feds
-s '(RAW_TEXT CONTAINS FEDS("Babe Ruth",CS=true,DIST=2,WIDTH=20))'

For Fuzzy Hamming or Fuzzy Edit search type: FHS or FEDS exact search is used if fuzziness is zero. Optional parameters:
• case sensitivity: CS=true|false, Default=true
• fuzziness distance: DIST=<int>, Default=0
• surrounding width: WIDTH=<int>, Default=0

Date Search: date_search | ds
-s '(RAW_TEXT CONTAINS DATE(04-01-1927 < MM-DD-YYYY < 09-30-1927))'

Time Search: time_search | ts
-s '(RAW_TEXT CONTAINS TIME(11:00:00 <= HH:MM:SS < 11:53:00))'

Numeric and Currency Search: numeric_search | ns
-s '(RAW_TEXT CONTAINS NUMBER(NUM = "1927",",","."))'
-s '(RAW_TEXT CONTAINS CURRENCY("$190CUR <= "$2000","$",",","."))'

IPV4 and IPv6 Search: IPv4 | IPv6
-s '(RAW_TEXT CONTAINS IPV4("127.0.0.0" <= IP < "127.255.255.255"))'
-s '(RAW_TEXT CONTAINS IPV6("2000::" <= IP < "2200::"))'

For structured data files, like XML & JSON, the search type can be for any specific tag in the record or for all tags in the record as specified in the rdf for this file:
• Specific tag: -s '(RECORD.Name CONTAINS "Babe Ruth",CS=true)'
All tags in record: -s '(RECORD CONTAINS "Babe Ruth",CS=true)'</td></tr>
<tr><td>-w | --width=<W></td><td>Integer</td><td>Specify the surrounding width to return with match. Can specify integer or use -w=line to specify that the entire line be returned when a match is found. Used with unstructured *_search primitives.
Default is width=0</td></tr>
</table>

Parameter	Type	Description
REST Specific Parameters		
`-a | --address=<addr>`	String	Specify the ryft-server address. Default is `http://localhost:8765`.
`--accept=<fmt>`	String	Accept format can be “json” or “csv”.
`--cluster`		Specify a cluster search. Opposite to `--local`. Default for search.
`--count`		Default. The `/count` endpoint; print just statistics. 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. Do not call Ryft hardware.
`--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 • `utf8` - to get human readable data (instead of base-64 encoded bytes). This parameter asks `ryft-server` 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.
`--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.

Parameter	Type	Description
REST Specific Parameters		
`--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 | --user<cred> | --auth`	String	Specify user credentials, “username:password”.
`-v | --verbose`		Tell curl to be verbose.
`-vv | --pretty`		Specify properly indented formatting with jq tool.

Here are some examples of the command and parameters:

- Search and print all occurrences of “Joe” in text files:

```
/usr/bin/ryftrest -q=Joe -f=*.txt -vv
```

  Notice that this example shows “`-q=Joe`” instead of “`-q ‘Joe’`” as the search value (as shown in the next 2 examples). The `ryftrest` command supports both methods.

- Print the number of matches and some performance numbers:

```
/usr/bin/ryftrest -q 'Joe' -f '*.txt' -vv --count
```

- Launch a structured search in “*.pcrime” files:

```
/usr/bin/ryftrest -q '(RECORD.id CONTAINS "100310")' -f '*.pcrime' --format=xml
--fields=ID,Date -vv
```

Search Mode Parameter

The `ryft-server` supports the following parameters for a new `mode`:

- Exact Search: `exact_search` (or `es`); DEFAULT
- Fuzzy Hamming Search: `fuzzy_hamming_search` (or `fhs`)
- Fuzzy Edit Distance Search: `fuzzy_edit_distance_search` (or `feds`),
- Date Search: `date_search` (or `ds`),
- Time Search: `time_search` (or `ts`)
- Numeric Search: `numeric_search` (or `ns`), also used for currency
- IPV4 Search: `ipv4_search` (or `ipv4`)

- IPV6 Search: `ipv6_search` (or `ipv6`)

If the search mode is empty or missing, it defaults to `fhs`, fuzzy hamming search, with one exception. If the query contains the keywords "DATE" or "TIME" then `ds` or `ts` mode will be used, respectively.

Here are examples that show the similarities of using `ryftprim` vs. `ryftrest` on the command line.

Fuzzy Hamming Search Example

```
ryftprim -p fhs -q '(RECORD.desc CONTAINS "Jones")' -f mychicagoJones.pcrime -d 1 –v
```

The same command using `ryftrest`, with the additional options of running on local sever and selection of fields to return.

```
ryftrest -p fhs -q '(RECORD.desc CONTAINS "Jones")' -f mychicagoJones.pcrime -d 1 -vv
--local --format=xml --fields=ID,Description
```

Fuzzy Edit Distance Search Example

```
ryftprim -p feds -q '(RECORD.desc CONTAINS "Jones")' -f mychicagoJones.pcrime -d 1 –v
```

The same command using `ryftrest` with the additional options of running on the cluster and selection of fields to return.

```
ryftrest -p feds -q '(RECORD.desc CONTAINS "Jones")' -f mychicagoJones.pcrime -d 1 -vv
--cluster --format=xml --fields=ID,Description
```

Date Search Example

```
ryftprim -p ds -q '(RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/13/2015))' -f
mychicago.pcrime –v
```

The same command using `ryftrest` with the additional options of running on local sever and executing the command to a different server.

```
ryftrest -p ds -q '(RECORD.date CONTAINS DATE(MM/DD/YYYY > 04/13/2015))' -f
mychicago.pcrime -vv --local --format=xml --address "http://ryftone-0008:8765 "
```

Time Search Example

```
ryftprim -p ts -q '(RECORD.date CONTAINS TIME(HH:MM:SS > 10:20:00))' -f
mychicago.pcrime –v
```

The same command using `ryftrest` with the additional options of running on local sever and format the results as XML.

```
ryftrest -p ts -q '(RECORD.date CONTAINS TIME(HH:MM:SS > 10:20:00))' -f
mychicago.pcrime -vv --local --format=xml
```

Complex Query Decomposition

One advantage of using `ryftrest` is the added benefit of query decomposition, which is not supported by `ryftprim`. Queries follow the query optimization rules, and the Query Decomposition configuration set within the "/etc/ryft-server.conf" configuration file.

Query decomposition combines a few subqueries of the same type to minimize Ryft hardware calls. Here's the basic query structure of three query statements combined using two AND operators:

```
(RECORD CONTAINS DATE(...)) AND (RECORD CONTAINS DATE(...)) AND (RECORD CONTAINS
"sometext")
```

This is translated to just two calls, because the first two subqueries have the same date search type.

- `(RECORD CONTAINS DATE(...)) AND (RECORD CONTAINS DATE(...))`
- `(RECORD CONTAINS "sometext")`

Here is an example of the original search that looks for any record where the ID contains “1003”.

```
ryftrest -q '(RECORD.id CONTAINS "1003")' -f mychicago.pcrime -vv --local --format=xml
--fields=ID,Date,Description
```

And here is how it can be decomposed using the output of the first query as the input of the second query:

- Take the same search and decompose it to narrow down the results so that it only returns data where the date also contains 04/14/2015 :

  ```
  ryftrest -q '(RECORD.id CONTAINS "1003") AND (RECORD.date CONTAINS
  DATE(MM/DD/YYYY > 04/14/2015))' -f mychicago.pcrime -vv --local --format=xml --
  fields=ID,Date,Description
  ```

- And then narrow it down even more by taking those results and using it as input to only return data where date also contains a time greater than 08:30:00:

  ```
  ryftrest -q '(RECORD.id CONTAINS "1003") AND (RECORD.date CONTAINS
  DATE(MM/DD/YYYY > 04/14/2015)) AND (RECORD.date CONTAINS TIME(HH:MM:SS >
  08:30:00))' -f mychicago.pcrime -vv --local --format=xml --
  fields=ID,Date,Description
  ```

The OR Operator

The OR operator is supported, with the following caveats:

- Duplicate records are possible because duplicates are not discarded.
- Large data files may be copied - two or more intermediate result files are copied into one.

Using the OR operator with complex queries or with multiple OR operators requires additional steps.

Let’s look at a sample input file and some example queries.

Example Input File

Here is an example input file “or.pcrime” that looks like this (beginning and end of each record is marked with <rec> and <rec/>). It has 10 records, and is a subset of the Chicago Crimes dataset.

```
<rec><ID>10034183</ID><CaseNumber>HY223673</CaseNumber><Date>04/10/2015 10:15:00
PM</Date><Block>002XX</Block><IUCR>0486</IUCR><PrimaryType>BATTERY</PrimaryType><Descriptio
n>John</Description><LocationDescription>STREET</LocationDescription><Arrest>false</Arrest>
<Domestic>true</Domestic><Beat>0313</Beat><District>003</District><Ward>20</Ward><Community
Area>42</CommunityArea><FBICode>08B</FBICode><XCoordinate>1181263</XCoordinate><YCoordinate>
1863965</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.781961688</Latitude><Longitude>-
87.610984705</Longitude><Location>"(41.781961688, -87.610984705)"</Location></rec>
<rec><ID>10034188</ID><CaseNumber>HY223687</CaseNumber><Date>04/10/2015 10:30:00
PM</Date><Block>003XX</Block><IUCR>0820</IUCR><PrimaryType>THEFT</PrimaryType><Description>
Jonny</Description><LocationDescription>SIDEWALK</LocationDescription><Arrest>false</Arrest
><Domestic>true</Domestic><Beat>1123</Beat><District>011</District><Ward>28</Ward><Communit
```

yArea>27</CommunityArea><FBICode>06</FBICode><XCoordinate>1152292</XCoordinate><YCoordinate>
1901795</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.886390821</Latitude><Longitude>-
87.716204071</Longitude><Location>"(41.886390821, -87.716204071)"</Location></rec>
<rec><ID>10034213</ID><CaseNumber>HY223716</CaseNumber><Date>04/11/2015 10:45:00
PM</Date><Block>001XX</Block><IUCR>0470</IUCR><PrimaryType>PUBLIC PEACE
VIOLATION</PrimaryType><Description>Jenny</Description><LocationDescription>ALLEY</Location
Description><Arrest>true</Arrest><Domestic>false</Domestic><Beat>0522</Beat><District>005</
District><Ward>9</Ward><CommunityArea>53</CommunityArea><FBICode>24</FBICode><XCoordinate>11
77304</XCoordinate><YCoordinate>1825999</YCoordinate><Year>2015</Year><UpdatedOn>04/22/201
5 12:47:10 PM</UpdatedOn><Latitude>41.67786846</Latitude><Longitude>-
87.626642113</Longitude><Location>"(41.67786846, -87.626642113)"</Location></rec>
<rec><ID>10034327</ID><CaseNumber>HY223684</CaseNumber><Date>04/11/2015 10:53:00
PM</Date><Block>075XX</Block><IUCR>0486</IUCR><PrimaryType>BATTERY</PrimaryType><Descriptio
n>Lenny</Description><LocationDescription>RESIDENTIAL YARD
(FRONT/BACK)</LocationDescription><Arrest>false</Arrest><Domestic>true</Domestic><Beat>062
3</Beat><District>006</District><Ward>6</Ward><CommunityArea>69</CommunityArea><FBICode>08B
</FBICode><XCoordinate>1178866</XCoordinate><YCoordinate>1854896</YCoordinate><Year>2015</Y
ear><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.757130314</Latitude><Longitude>-
87.620048394</Longitude><Location>"(41.757130314, -87.620048394)"</Location></rec>
<rec><ID>10034247</ID><CaseNumber>HY223708</CaseNumber><Date>04/12/2015 10:52:00
PM</Date><Block>081XX</Block><IUCR>0486</IUCR><PrimaryType>BATTERY</PrimaryType><Descriptio
n>Manny</Description><LocationDescription>VEHICLE NON-
COMMERCIAL</LocationDescription><Arrest>false</Arrest><Domestic>true</Domestic><Beat>0414<
/Beat><District>004</District><Ward>8</Ward><CommunityArea>46</CommunityArea><FBICode>08B</
FBICode><XCoordinate>1190898</XCoordinate><YCoordinate>1851594</YCoordinate><Year>2015</Yea
r><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.747787125</Latitude><Longitude>-
87.57606016</Longitude><Location>"(41.747787125, -87.57606016)"</Location></rec>
<rec><ID>10034197</ID><CaseNumber>HY223707</CaseNumber><Date>04/12/2015 11:18:00
PM</Date><Block>001XX</Block><IUCR>1811</IUCR><PrimaryType>NARCOTICS</PrimaryType><Descript
ion>More</Description><LocationDescription>STREET</LocationDescription><Arrest>true</Arrest>
<Domestic>false</Domestic><Beat>1523</Beat><District>015</District><Ward>28</Ward><Communit
yArea>25</CommunityArea><FBICode>18</FBICode><XCoordinate>1141655</XCoordinate><YCoordinate>
1900379</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.882708414</Latitude><Longitude>-
87.75530118</Longitude><Location>"(41.882708414, -87.75530118)"</Location></rec>
<rec><ID>10034248</ID><CaseNumber>HY223738</CaseNumber><Date>04/13/2015 11:48:00
PM</Date><Block>015XX</Block><IUCR>1121</IUCR><PrimaryType>DECEPTIVE
PRACTICE</PrimaryType><Description>Less</Description><LocationDescription>RESTAURANT</Locat
ionDescription><Arrest>false</Arrest><Domestic>false</Domestic><Beat>1012</Beat><District>0
10</District><Ward>24</Ward><CommunityArea>29</CommunityArea><FBICode>10</FBICode><XCoordina
te>1149938</XCoordinate><YCoordinate>1891833</YCoordinate><Year>2015</Year><UpdatedOn>04/22
/2015 12:47:10 PM</UpdatedOn><Latitude>41.859100084</Latitude><Longitude>-
87.725107817</Longitude><Location>"(41.859100084, -87.725107817)"</Location></rec>
<rec><ID>10037110</ID><CaseNumber>HY224060</CaseNumber><Date>04/13/2015 11:35:00
PM</Date><Block>011XX</Block><IUCR>0910</IUCR><PrimaryType>MOTOR VEHICLE
THEFT</PrimaryType><Description>No</Description><LocationDescription>STREET</LocationDescrip
tion><Arrest>false</Arrest><Domestic>false</Domestic><Beat>1211</Beat><District>012</Distri
ct><Ward>26</Ward><CommunityArea>24</CommunityArea><FBICode>07</FBICode><XCoordinate>115612
8</XCoordinate><YCoordinate>1907708</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015
12:47:10 PM</UpdatedOn><Latitude>41.902540073</Latitude><Longitude>-
87.701957503</Longitude><Location>"(41.902540073, -87.701957503)"</Location></rec>
<rec><ID>10034200</ID><CaseNumber>HY223668</CaseNumber><Date>04/14/2015 11:40:00
PM</Date><Block>054XX</Block><IUCR>0486</IUCR><PrimaryType>BATTERY</PrimaryType><Descriptio
n>John</Description><LocationDescription>RESIDENCE</LocationDescription><Arrest>false</Arre
st><Domestic>true</Domestic><Beat>1522</Beat><District>015</District><Ward>29</Ward><Commun

```
ityArea>25</CommunityArea><FBICode>08B</FBICode><XCoordinate>1140152</XCoordinate><YCoordina
te>1897108</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.873760014</Latitude><Longitude>-
87.760900431</Longitude><Location>"(41.873760014, -87.760900431)"</Location></rec>
<rec><ID>10034234</ID><CaseNumber>HY223685</CaseNumber><Date>04/15/2015 11:30:00
PM</Date><Block>011XX</Block><IUCR>0320</IUCR><PrimaryType>ROBBERY</PrimaryType><Descriptio
n>Job</Description><LocationDescription>SIDEWALK</LocationDescription><Arrest>false</Arrest
><Domestic>false</Domestic><Beat>1824</Beat><District>018</District><Ward>42</Ward><Communi
tyArea>8</CommunityArea><FBICode>03</FBICode><XCoordinate>1175283</XCoordinate><YCoordinate>
1908223</YCoordinate><Year>2015</Year><UpdatedOn>04/22/2015 12:47:10
PM</UpdatedOn><Latitude>41.903544846</Latitude><Longitude>-
87.631582982</Longitude><Location>"(41.903544846, -87.631582982)"</Location></rec>
```


Example Queries

Using the "or.pcrime" data file, above, consider these three sample queries:

1. To get all records:
   ```
   (RECORD.id CONTAINS "1003")
   ```
2. To get first 6 records:
   ```
   (RECORD.date CONTAINS DATE(MM/DD/YYYY <= 04/12/2015))
   ```
3. To get last 5 records:
   ```
   (RECORD.date CONTAINS TIME(HH:MM:SS > 11:15:00))
   ```

Now, let's look at combining two of the queries into one:

1. Combine queries number 2 and 3 with the OR operator, and the system returns 11 records (ID=10034197 will be included twice):
   ```
   ((RECORD.date CONTAINS DATE(MM/DD/YYYY <= 04/12/2015)) OR (RECORD.date CONTAINS
   TIME(HH:MM:SS > 11:15:00)))
   ```
2. Combine queries number 1 and 3 with the OR operator, and the system returns 15 records (the last 5 will have duplicates):
   ```
   (RECORD.id CONTAINS "1003") OR (RECORD.date CONTAINS TIME(HH:MM:SS > 11:15:00))
   ```
3. Use both the AND and OR operators together, like this, and the system returns 11 matches.
   ```
   (RECORD.id CONTAINS "1003") AND ((RECORD.date CONTAINS DATE(MM/DD/YYYY <=
   04/12/2015)) OR (RECORD.date CONTAINS TIME(HH:MM:SS > 11:15:00)))
   ```

NOTE: When using both the AND and OR operators in the same command line, the OR operator has lower priority. This means that so using "`a OR b AND c`" is equivalent to using "`a OR (b AND c)`":

```
(RECORD.date CONTAINS DATE(MM/DD/YYYY <= 04/12/2015)) OR (RECORD.date CONTAINS
TIME(HH:MM:SS > 11:15:00)) AND (RECORD.id CONTAINS "1003")
```

Minimize Output

To minimize output, use the `--count` option. Passing this flag makes the tool use the `/count` endpoint instead of `/search` so that only the search statistics are printed, not the search results.

```
ryftrest -q '(RECORD.id CONTAINS "100310")' -f '*.pcrime' --local --format=xml --
fields=ID,Date --count
```

```
{"matches":24,"totalBytes":1415539,"duration":1140,"dataRate":1.1841782352380585,"fabr
icDataRate":0}
```

An alternative option is to use `jq` command line JSON processor:

```
ryftrest -q '(RECORD.id CONTAINS "100310")' -f '*.pcrime' --local --format=xml --
fields=ID,Date | jq ".stats"
{
  "fabricDataRate": 6573.359375,
  "dataRate": 9.723904570178872,
  "duration": 676,
  "totalBytes": 6892667,
  "matches": 81
}
```

This command is almost equal to the previous one. Without the `--count` option, all data is still transferred. All processing is done on the client side.

Using `jq` it's possible to do advanced data processing. This command prints a list of matching date strings:

```
ryftrest -q '(RECORD.id CONTAINS "100310")' -f '*.pcrime' --local --format=xml --
fields=ID,Date | jq ".results[].Date"
"04/13/2015 11:45:00 PM"
"04/13/2015 11:40:00 PM"
"04/13/2015 11:30:00 PM"
"04/13/2015 11:25:00 PM"
"04/13/2015 11:20:00 PM"
"04/13/2015 11:18:00 PM"
"04/13/2015 11:10:00 PM"
"04/13/2015 11:00:00 PM"
"04/13/2015 11:35:00 AM"
"04/13/2015 11:00:00 AM"
"04/13/2015 11:53:00 PM"
"04/13/2015 11:45:00 PM"
"04/13/2015 11:40:00 PM"
"04/13/2015 11:30:00 PM"
"04/13/2015 11:25:00 PM"
"04/13/2015 11:20:00 PM"
"04/13/2015 11:18:00 PM"
"04/13/2015 11:10:00 PM"
"04/13/2015 11:00:00 PM"
```

Preserve Search Results

By default, all search results are deleted from the Ryft server once they are delivered to user. However, you can preserver your search results so that the output from the first search becomes the input for the second search.

Enable the "search in the previous results" feature using two query parameters:

- Output File: The `data=output.dat` or `–od output.dat` parameter keeps the search results on the Ryft server in the file /ryftone/output.dat. It is possible to use that file as an input for the subsequent search call `files=output.dat`.

  NOTE: It is important to use consistent file extension for the structured search in order to let Ryft use appropriate RDF scheme!

- Index File: The `index=index.txt` or `-oi index.txt` parameter keeps the search index under /ryftone/index.txt.

Here is an example:

```
ryftrest -q '(RECORD.id CONTAINS "100310")' -f '*.pcrime' --local --format=xml --
fields=ID,Date –od mytest.pcrime –oi mytest.txt
cat /ryftone/mytest.txt
```

The `data=` and `index=` query parameters are supported by both `/search` and `/count` endpoints.

In case of complex search expression, query decomposition saves data and index of the top subquery.

WARNING: If data or index file with the same name already exist, it will be overridden!

JSON Format Support

If the input file set contains JSON data, the found records could be decoded using the `format=json` query parameter.

```
ryftrest -q '(RECORD.Name CONTAINS "Bruce")' -f 'CitizensOfGotham.json' --local
--format=json
```

This is the output:


```
{"results":[{"Actors":[{"Name":"Adam West"},{"Name":"Michael Keaton"},{"Name":"Val
Kilmer"},{"Name":"George Clooney"},{"Name":"Christian Bale"}],"AlterEgo":"The
Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":8,"length":1108,"fuzziness":
0,"host":"ryftone-310"}}
,{"Actors":[{"Name":"Adam West"},{"Name":"Michael Keaton"},{"Name":"Val
Kilmer"},{"Name":"George Clooney"},{"Name":"Christian Bale"}],"AlterEgo":"The
Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":5688,"length":1108,"fuzzines
s":0,"host":"ryftone-310"}}
,{"Actors":[{"Name":"Adam West"},{"Name":"Michael Keaton"},{"Name":"Val
Kilmer"},{"Name":"George Clooney"},{"Name":"Christian Bale"}],"AlterEgo":"The
Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":11368,"length":1108,"fuzzine
ss":0,"host":"ryftone-310"}}
],"stats":{"matches":3,"totalBytes":17040,"duration":652,"dataRate":0.0249242490054639
55,"fabricDataRate":0}
```


To minimize output or to get just a subset of fields, use the `fields=` query parameter. For example, to get AlterEgo and Name fields from the "CitizensOfGotham.json" file, pass this parameter: `format=json&fields=AlterEgo,Name`.

```
ryftrest -q '(RECORD.Name CONTAINS "Bruce")' -f 'CitizensOfGotham.json' --local --
format=json --fields=AlterEgo,Name
```

Here is the output:


```
{"results":[{"AlterEgo":"The Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":8,"length":1108,"fuzziness":
0,"host":"ryftone-310"}}
,{"AlterEgo":"The Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":5688,"length":1108,"fuzzines
s":0,"host":"ryftone-310"}}
,{"AlterEgo":"The Batman","Name":"Bruce
Wayne","_index":{"file":"/CitizensOfGotham.json","offset":11368,"length":1108,"fuzzine
ss":0,"host":"ryftone-310"}}
```

```
],"stats":{"matches":3,"totalBytes":17040,"duration":734,"dataRate":0.0221397961192949
6,"fabricDataRate":0}
```

NOTE: JSON format (just like XML) can only be used with structured search.

Post-Process Transformations

The output data can be transformed on the server side just before it is reported to the client. A regular expression or custom application/script can be used to perform transformations. The `transform` query option supports three custom transformations:

- `Match` – a filter to only display matches to the regular expression.
- `Replace` – a filter to find a match to the regular expression, and then replace the matching string with the specified string, and display the results.
- `Script` – define a custom application/script to use for transformation.

NOTE: the output statistic contains initial number of matches so that you can check the number of dropped records as the difference between matches and the actual number of records received. The same is true for indexes. Indexes contain initial data position and length without any transformations reflected.

You can also create a transformation chain where the output of the first transformation becomes the input for the second transformation.

A simple JSON file called "test.json" is used in the following examples. If no transformation is applied to the output, it would look like this:

```
$ ryftrest -q "hello" -f test.json -w=5 --format=utf8 --search -u admin:admin | jq -c
.results[].data
"1111-hello-1111"
"2222-hello-2222"
"3333-hello-3333"
"4444-hello-4444"
"5555-hello-5555"
```

Match Transformation

The regular expression `match` transformation is used as a filter. If a record does not match the regular expression, then the record is just dropped.

This transformation is defined as `transform=match("expression")` where `expression` is a valid regular expression applied to the found record. For example, the following transformation will report only records containing `markX` where `X` is a digit:

```
transform=match("^.*mark[0-9].*$")
```

In this example, it performs a post-processing match for odd numbers:

```
$ ryftrest -q "hello" -f test.json -w=5 --format=utf8 --search -u admin:admin \
  --transform 'match("^[13579].*$")' \
  | jq -c .results[].data
"1111-hello-1111"
"3333-hello-3333"
"5555-hello-5555"
```

Replace Transformation

The regular expression `replace` transformation is used as a simple "match and replace." If a record does match the regular expression, then it is replaced with a template. If a record does not match, then it remains "as is" with no replacement.

This transformation is defined as `transform=replace("expression", "template")` where `expression` is a valid regular expression and the `template` is replacement text. The template can use special variables `$1`, `$2`, etc. to refer to the matched text.

For example, the following transformation will replace all "apples" with "oranges":

```
transform=replace("^(.*)apple(.*)$", "${1}orange${2}")
```

In this example, let's match "hello" and replace it with "bye".

```
$ ryftrest -q 'RAW_TEXT CONTAINS "hello"' -f test.json -w=5 --format=utf8 --search -u
admin:admin \
  --transform 'replace("^(.*)hello(.*)$", "${1}bye${2}")' \
  | jq -c .results[].data
"1111-bye-1111"
"2222-bye-2222"
"3333-bye-3333"
"4444-bye-4444"
"5555-bye-5555"
```

Script Transformation

The `script` transformation uses an external application or script to transform a record. A matching record is written to the `STDIN` of the stated script and the transformed record is read from the `STDOUT`. If the exit status of the script is non-zero, then the record is dropped.

This transformation is defined as `transform=script("name")` where name is a predefined script name.

Configure valid script names using the "/etc/ryft-server.conf" configuration file. In general, any script can be used:

```
post-processing-scripts:
  false:
    path: [/bin/false]
  cat:
    path: [/bin/cat]
  my_test1:
    path: [/usr/bin/jq, -c, "{lat: .lat, lon: .lon}"]
```

Each item is a script name and includes the full path to the application/script, along with a set of additional command line options.

For example, this is the script "jq_ab" that adds two fields a+b:

```
jq_ab:
  path: [/usr/bin/jq, -c, "{\"a+b\": (.a + .b), \"a\": .a, \"b\": .b}"]
```

Let's use RECORD-based search and use the "jq_ab" script – this is the result:

```
ryftrest -q 'RECORD.text CONTAINS "hello"' -f test.json --format=utf8 --search -u
admin:admin \
```

```
  --transform 'script("jq_ab")' \
  | jq -r -c .results[].data | jq -c .
{"a+b":11,"a":10,"b":1}
{"a+b":22,"a":20,"b":2}
{"a+b":33,"a":30,"b":3}
{"a+b":44,"a":40,"b":4}
{"a+b":55,"a":50,"b":5}
```

Transformation Chain

A few transformations can be combined into the transformation chain. This examples uses both match and replace with the same data:

```
$ ryftrest -q 'RAW_TEXT CONTAINS "hello"' -f test.json -w=5 --format=utf8 --search -u
admin:admin \
  --transform 'match("^[13579].*$")' \
  --transform 'replace("^(.*)hello(.*)$", "${1}bye${2}")' \
  | jq -c .results[].data
"1111-bye-1111"
"3333-bye-3333"
"5555-bye-5555"
```

In this case, each found record is processed by two transformations:

- If the first transformation matches, then the second transformation is applied.
- If the first transformation does not match, then the record is dropped, therefore the second transformation cannot be considered.

It first matches records with odd numbers, and for the found records, replaces “hello” with “bye” and then outputs the results.

You can also call a script with a transformation and include it as part of the transformation chain, or individually.

5

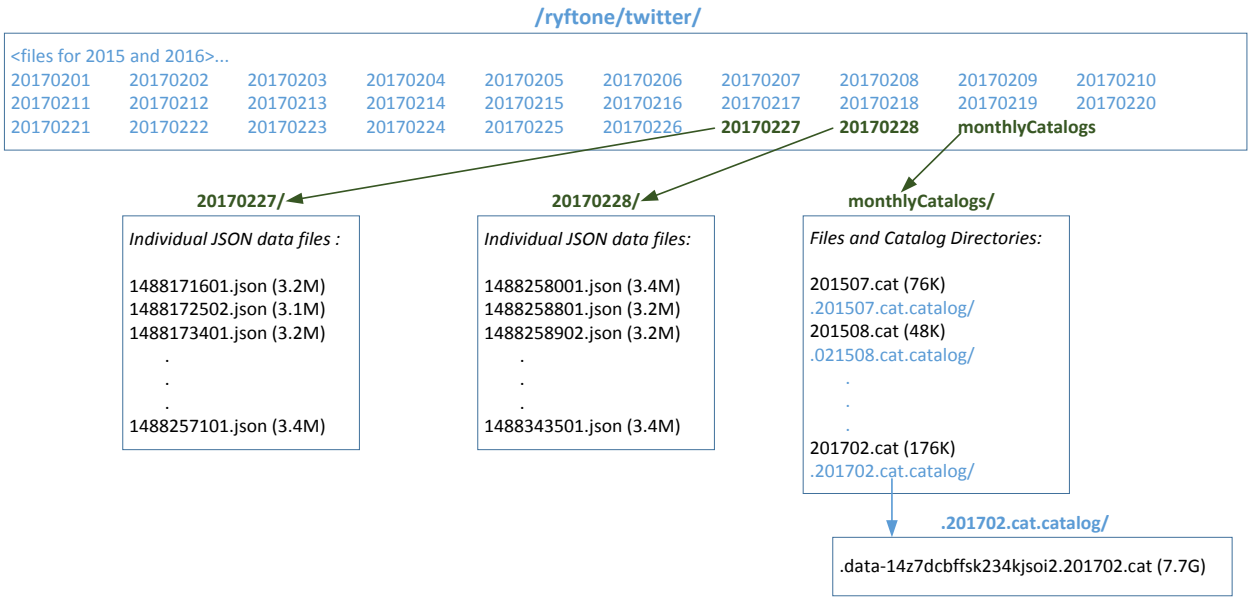
5: Create Catalog for Large Data Sets

The number of files in a search dataset, the size of each file, and the chunk size defined in an RDF can all impact search performance. The Ryft ONE and the `ryftrest` primitive work best with a smaller number of large data files instead of lots of smaller files; it's just more efficient. However, if you only have an abundance of small files, how do you efficiently use Ryft ONE and its processing capabilities? You can, by combining the small files into one or a few large "catalog" files without affecting the original files.

What is a Catalog?

A catalog is an abstraction that collects a user-specified set of smaller files into one large file. When you search the catalog, the resulting index file contains references to the initial small files. The catalog looks like a normal sub-directory that contains a set of small files, all with the same name followed by a unique file extension. It's implemented as an index file and a set of big data files with a configurable size limit.

Here's how it looks for the daily Twitter datafiles that Ryft collects. The top shows that each day is a separate directory and under each day's directory are numerous JSON files that are approximately 3MB, each.



So here, the “20170227” and “2017228” are sub directories that contain many small JSON files of tweets for that day and each file is approximately 3MB. The “monthlyCatalogs” directory contains the catalogs that combine a month’s worth of smaller files into one large file, like the “201702.cat” file. Its related subdirectory is “.201702.cat.catlog” which has just one very large file (7.7G) – and this is the combination of all the smaller JSON files contained in the “201702*” directories.

Upload Files

In this scenario, Twitter data files were pulled from Twitter at 15 minute intervals, on February 28, 2017, and uploaded to the RyftONE. Each file is just over 3MB. This Twitter sub-directory contains all the data files for just that one date.

```
ryftuser@ryftone-313:/ryftone/twitter/20170228$ ls
1488258001.json  1488268801.json  1488279601.json  1488290402.json  1488301201.json  1488312001.json  1488322801.json  1488333601.json
1488258901.json  1488269701.json  1488280501.json  1488291301.json  1488302101.json  1488312901.json  1488323701.json  1488334501.json
1488259802.json  1488270601.json  1488281402.json  1488292201.json  1488303001.json  1488313801.json  1488324601.json  1488335402.json
1488260701.json  1488271502.json  1488282302.json  1488293102.json  1488303901.json  1488314701.json  1488325501.json  1488336301.json
1488261601.json  1488272402.json  1488283201.json  1488294001.json  1488304801.json  1488315602.json  1488326401.json  1488337201.json
1488262502.json  1488273301.json  1488284101.json  1488294901.json  1488305701.json  1488316501.json  1488327301.json  1488338101.json
1488263401.json  1488274201.json  1488285001.json  1488295801.json  1488306601.json  1488317401.json  1488328201.json  1488339001.json
1488264302.json  1488275102.json  1488285901.json  1488296701.json  1488307501.json  1488318302.json  1488329101.json  1488339901.json
1488265201.json  1488276001.json  1488286801.json  1488297601.json  1488308401.json  1488319202.json  1488330001.json  1488340801.json
1488266101.json  1488276901.json  1488287701.json  1488298501.json  1488309301.json  1488320101.json  1488330901.json  1488341702.json
1488267001.json  1488277802.json  1488288602.json  1488299401.json  1488310201.json  1488321001.json  1488331801.json  1488342601.json
1488267901.json  1488278701.json  1488289501.json  1488300302.json  1488311101.json  1488321901.json  1488332701.json  1488343501.json
ryftuser@ryftone-313:/ryftone/twitter/20170228$ ls -alh
total 303M
drwxrwxr-x 0 ryftuser ryftuser    0 Feb 28 00:00 .
drwxr-xr-x 0 ryftuser ryftuser    0 Feb  2 16:59 ..
-rw-rw-r-- 0 ryftuser ryftuser 3.4M Feb 28 00:00 1488258001.json
-rw-rw-r-- 0 ryftuser ryftuser 3.3M Feb 28 00:15 1488258901.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 00:30 1488259802.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 00:45 1488260701.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 01:00 1488261601.json
-rw-rw-r-- 0 ryftuser ryftuser 3.3M Feb 28 01:15 1488262502.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 01:30 1488263401.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 01:45 1488264302.json
-rw-rw-r-- 0 ryftuser ryftuser 3.3M Feb 28 02:00 1488265201.json
-rw-rw-r-- 0 ryftuser ryftuser 3.3M Feb 28 02:15 1488266101.json
-rw-rw-r-- 0 ryftuser ryftuser 3.2M Feb 28 02:30 1488267001.json
-rw-rw-r-- 0 ryftuser ryftuser 3.3M Feb 28 02:45 1488267901.json
```

Create the Catalog

Use the `ryftutil` command to create the catalog. It’s a useful command that gives you command line access to the RyftREST API and can perform many actions. Here is the help text for the command:

```
$ ryftutil -help
usage: /usr/bin/ryftutil

command line access to RyftAPI file list, copy, delete features

Revision: 0.0.1

-help                       shows usage
-vers                       shows the version of this script (also shown in help)

-ls|-dir                    uses the server to list contents of -f directory/ on RyftONE
-cp|-copy <filename>        Copies the <filename> files or directory tree to the target
                            RyftONE (no wildcards are allowed in the directory path)
    -lifetime <#>h              sets file/catalog lifetime to seconds, minutes or hours
(#s|#m|#h)
    -maxdepth <depth>           sets directory traversal depth (default is 1, 0 for full
recursion)
    -stripPath|-sp              strips the source directory path from the destination
file compressing
                                multi-level directories into a single RyftONE directory
    -append                     append to exsting files when copying
    -replace                    replace existing files when copying (default)
```

```
-rm|-del                  Deletes specified files (-f file), directories (-f
directory/), or
                          catalogs (-c catalogs) on the target RyftONE

-f <filename>             sets target RyftONE file <filename>.
-c <filename>             sets target RyftONE catalog <filename>
     -cache-drop-timeout <#>   defaults 10 seconds ( from /etc/ryft-server.conf )
     -delimiter <delimiter>    defaults to RyftONE's /etc/ryft-server.conf:default-
data-delim,
                               typically "%0A%0C%0A"
     -stamp                    appends current _YYYYMMDD-HHMMSS to catalog name


-a <address>              defaults to localhost:8765
-u user:password          defaults to admin:admin
-local                    operation applies to a single RyftONE (default)
-cluster                  operation applies to a RyftONE Cluster

                          See CCStrategy.txt for a sample strategy.
-verbose                  turn on intermediate status statements
-debug <level>            turn on debugging output where <level> is an integer
-show                     show ryftrest/api commands

Examples:
  # list file on remote host's /ryftone filesystem
        ryftutil -a hostname:8765 -ls -f /

  # list files on remote host's /ryftone/directory
        ryftutil -a hostname:8765 -ls -f /directory

  # copy localfile to remote host's /ryftone/remotefile
  ryftutil -a hostname:8765 -cp localfile -f /remotefile

  # copy localfile to remote host's catalog
  ryftutil -a hostname:8765 -cp localfile -c /remoteCatalog

  # append file to remote host's catalog
  ryftutil -a hostname:8765 -cp anotherfile -c /remoteCatalog -append

  # delete file/directory on remote host
  ryftutil -a hostname:8765 -del -f /remotefile
  ryftutil -a hostname:8765 -del -f /remotedirectory
```

Using `ryftutil`, create a catalog named “twitter_Feb_Catalog” of all the files from the “/ryftone/twitter/20170228” directory, and put it in the “/ryftone/training” directory, like this:

```
ryftuser@ryftone-313:/ryftone/twitter$ ryftutil -copy /ryftone/twitter/20170228 -c
training/twitter_Feb_Catalog

Pushed 96 file(s), 317178411 bytes.
Elapsed time: 115 secs
```

It combined 96 files into a catalog in almost 2 minutes.

Lets see the catalog it created in the “/ryftone/training” directory:

```
ryftuser@ryftone-313:/ryftone/twitter$ cd /ryftone/training
ryftuser@ryftone-313:/ryftone/training$ ls -al
total 20
drwxrwxr-x 0 ryftuser ryftuser     0 Jan 15 13:04 .
drwxrwxr-x 0 ryftone  ryftone      0 Nov 10 10:48 ..
-rw-r--r-- 0 ryftuser ryftuser 20480 Mar  9 13:12 twitter_Feb_Catalog
drwxr-xr-x 0 ryftuser ryftuser     0 Mar  7 10:44 .twitter_Feb_Catalog.catalog
ryftuser@ryftone-313:/ryftone/training$ cd .twitter_Feb_Catalog.catalog/
ryftuser@ryftone-313:/ryftone/training/.twitter_Feb_Catalog.catalog$ ls -al
total 309743
drwxr-xr-x 0 ryftuser ryftuser        0 Mar  7 10:44 .
drwxrwxr-x 0 ryftuser ryftuser        0 Jan 15 13:04 ..
-rw-r--r-- 0 ryftuser ryftuser 64215977 Mar  7 10:45 .data-14a9a400fa5ecc14.twitter_Feb_Catalog
-rw-r--r-- 0 ryftuser ryftuser 67100083 Mar  7 10:45 .data-14a9a40aa2d85472.twitter_Feb_Catalog
-rw-r--r-- 0 ryftuser ryftuser 66100459 Mar  7 10:45 .data-14a9a40d78aaabf2.twitter_Feb_Catalog
-rw-r--r-- 0 ryftuser ryftuser 64145164 Mar  7 10:45 .data-14a9a410c3b007b1.twitter_Feb_Catalog
-rw-r--r-- 0 ryftuser ryftuser 55617016 Mar  7 10:46 .data-14a9a4130f3bef74.twitter_Feb_Catalog
```

Query the Catalog

To search tweets for February 28 2017, you use the “twitter_Feb_Catalog” file, which is just 20MB, like this:

```
ryftuser@ryftone-313:/ryftone/twitter/monthlyCatalogs$ ryftrest -f
training/twitter_Feb_Catalog -q '(RAW_TEXT CONTAINS "Korea")' -vv -p es -od CATTEST
-oi CATTESTIndex -w 30
{
  "Duration(sec)          ": 2.3,
  "Total Bytes(MB)        ": 7.63,
  "Matches                ": 13054,
  "Fabric Data Rate(GB/s)": 18.86,
  "Fabric Duration(msec) ": 404
}
```

It took just 2.3 second to run the query on the catalog.

Compare that with running the same query on the entire directory of the original *.json files in directories that begin “/ryftone/twitter/201702*” to encompass all of February 2017. This search took 9.5 seconds and found the same number of results:

```
ryftuser@ryftone-313:/ryftone/twitter/monthlyCatalogs$ ryftrest -f
twitter/201702*/*.json -q '(RAW_TEXT CONTAINS "Korea")' -vv -p es -od CATTEST -oi
CATTESTIndex -w 30
{
  "Duration(msec)         ": 9.5,
  "Total Bytes(MB)        ": 7.63,
  "Matches                ": 13054,
  "Fabric Data Rate(GB/s)": 16.18,
  "Fabric Duration(msec) ": 471
}
```

Catalog Output

The output data will be the same, whether you search the catalog or the original JSON files:

```
ryftuser@ryftone-313:/ryftone$ head -10 CATTEST
gAng Media Network (JMnet) is Korea's leading comprehensive media
"lang":"ko","location":"South Korea","default_profile_image":fals
":"Journalist writing on both Koreas for Radio France Internation
lse,"location":"SEOUL , South Korea","time_zone":"Seoul","name":"
ime_zone":"Seoul","location":"Korea","profile_background_image_ur
:31 +0000 2017","text":"South Korean lawmakers move to curb presi
},"text":"RT @business: South Korean lawmakers move to curb presi
gAng Media Network (JMnet) is Korea's leading comprehensive media
slator":false,"location":"USA/Korea","profile_sidebar_fill_color"
ded_url":"https://twitter.com/Koreandogs/status/80220494054653952
```

Looking at the resulting index file, notice that the index begins “twitter/20170201” – it points to the original daily sub-directory under “twitter” and the specific JSON data file, in addition to the normal index information.

Here is the beginning of the “CATTESTIndex” file:

```
ryftuser@ryftone-313:/ryftone$ head -10 CATTESTIndex
/ryftone/twitter/20170201/1485925201.json,856433,65,0
/ryftone/twitter/20170201/1485925201.json,864540,65,0
/ryftone/twitter/20170201/1485926102.json,542638,65,0
/ryftone/twitter/20170201/1485926102.json,643910,65,0
/ryftone/twitter/20170201/1485926102.json,951349,65,0
/ryftone/twitter/20170201/1485927001.json,607700,65,0
/ryftone/twitter/20170201/1485927001.json,612360,65,0
/ryftone/twitter/20170201/1485927001.json,946795,65,0
/ryftone/twitter/20170201/1485927901.json,391390,65,0
/ryftone/twitter/20170201/1485927901.json,584251,65,0
```

And the end of the “CATTESTIndex” file:

```
ryftuser@ryftone-308:/ryftone$ tail -10 CATTESTIndex
/ryftone/twitter/20170228/1488343501.json,585213,65,0
/ryftone/twitter/20170228/1488343501.json,586306,65,0
/ryftone/twitter/20170228/1488343501.json,586336,65,0
/ryftone/twitter/20170228/1488343501.json,586420,65,0
/ryftone/twitter/20170228/1488343501.json,586929,65,0
/ryftone/twitter/20170228/1488343501.json,587155,65,0
/ryftone/twitter/20170228/1488343501.json,588101,65,0
/ryftone/twitter/20170228/1488343501.json,794072,65,0
/ryftone/twitter/20170228/1488343501.json,2627891,65,0
/ryftone/twitter/20170228/1488343501.json,2630575,65,0
```

Append Data File to Existing Catalog

You can append files to the end of an existing catalog using the `ryftutil` command, if it is of the same data type and uses the same RDF as the original files.

In this example, we have a twitter catalog called “2017_Mar_Catalog” and we need to append yesterday’s data files to the catalog. Here’s a partial list of the files in the 20170325 directory:

```
ryftuser@ryftone-313:/ryftone/twitter/20170325$ ls -alh
total 5.3M
drwxr-xr-x 0 ryftuser ryftuser    0 Mar 25 17:00 .
drwxr-xr-x 0 ryftuser ryftuser    0 Mar 25 16:59 ..
-rwxr-xr-x 0 ryftuser ryftuser 345K Mar 25 17:00 1436440058.json
-rwxr-xr-x 0 ryftuser ryftuser 335K Mar 25 17:00 1436440071.json
-rwxr-xr-x 0 ryftuser ryftuser 353K Mar 25 17:00 1436441258.json
-rwxr-xr-x 0 ryftuser ryftuser 360K Mar 25 17:00 1436441341.json
-rwxr-xr-x 0 ryftuser ryftuser 320K Mar 25 17:00 1436442398.json
```

Here is the current catalog file and its size:

```
ryftuser@ryftone-313:/ryftone/training$ ls -al
total 20
drwxrwxr-x 0 ryftuser ryftuser     0 Jan 15 13:04 .
drwxrwxr-x 0 ryftone  ryftone      0 Nov 10 10:48 ..
-rw-r--r-- 0 ryftuser ryftuser 20480 Mar 20 17:19 2017_Mar_Catalog
drwxr-xr-x 0 ryftuser ryftuser     0 Mar  7 10:44 .2017_Mar_Catalog.catalog
```

To append the files, execute the `ryftutil` command using the `-append` flag:

```
ryftuser@ryftone-313:/ryftone/twitter$ ryftutil -copy /ryftone/twitter/20170325 -c
training/2017_Mar_Catalog -append

Pushed 96 file(s), 326049804 bytes.
Elapsed time: 50 secs
```

List the catalog file again and notice that the file grew by 8K from 20480 to 28672:

```
ryftuser@ryftone-313:/ryftone/training$ ls -al
total 28
drwxrwxr-x 0 ryftuser ryftuser     0 Jan 15 13:04 .
drwxrwxr-x 0 ryftone  ryftone      0 Nov 10 10:48 ..
-rw-r--r-- 0 ryftuser ryftuser 28672 Mar 28 14:55 2017_Mar_Catalog
drwxr-xr-x 0 ryftuser ryftuser     0 Mar  7 10:44 .2017_Mar_Catalog.catalog
ryftuser@ryftone-313:/ryftone/training$
```

Append Data File to Existing Data File

You can also append additional data files to an existing data file on your "/ryftone" filesystem. Use the same `ryftutil` command, with a "-f" parameter vs. "-c":

```
ryftuser@ryftone-313:/ryftone/twitter$ ryftutil -copy
/ryftone/reddit/2017/March/20170325.json -f reddit/2017_Mar_Reddit.json -append
```