



# **Securonix Multi-Tenant Documentation**

**13 January 2025**

# Contents

- Spotter. . . . . 3
  - Getting Started with Spotter Queries. . . . . 3
  - Spotter Query Optimization. . . . . 14
  - Spotter Search Examples. . . . . 28
  - Spotter Reports. . . . . 31
    - Executing Bulk Export Reports in Spotter. . . . . 31
    - Running Reports from Spotter. . . . . 32
  - Spotter Metrics. . . . . 36
- Spotter Search Help. . . . . 40
- Indicator Service (Beta). . . . . 98
  - Working with the Indicator Service (Beta). . . . . 100
- Data Masking in Spotter. . . . . 102
- Using Special Characters in Spotter. . . . . 106

# Spotter

Spotter is a lightning fast, natural language search engine that uses normalized search syntax and visualization techniques to provide threat hunters the tools they need to investigate current threats and trends, and track advanced persistent threats over long periods of time.

From the Spotter start screen, you can search for and view threats using various search filters. You can specify the report format to display information in tables, as bar charts, bubble charts, and time charts, or view a geographical map.

This guide is intended to help beginners get acquainted with the basics of Spotter search queries. It contains queries, functions, keywords, and the syntax available in Spotter, and provides you with a quick start to the Spotter search language. It includes the following topics:

[Getting Started with Spotter Queries](#)

[Spotter Query Optimization](#)

[Spotter Search Examples](#)

[Spotter Search Help](#)

[Indicator Service \(Beta\)](#)

[Data Masking in Spotter](#)

[Using Special Characters in Spotter](#)

## Getting Started with Spotter Queries

Spotter filters data by instructing Unified Defense SIEM to return a specified set of events. Enter this instruction in the form of a query. A query is a string of text that uses specific keywords and syntax to return targeted events.

The following sections will help you get started with Spotter queries:

- [Query Processing and Execution](#)
- [Query Terminology](#)
- [Query Syntax](#)
- [Search Indexes](#)
- [Common Fields](#)
- [Global Search](#)
- [Keyword Searches](#)

## Query Processing and Execution

When you execute a query, it goes through a step-by-step process to display results on the Spotter UI. The following steps occur during the Spotter query execution process:

1. The query converts to retrieve documents.
2. The query executes.
3. The query returns documentIDs to the query cache for each set of search terms.
4. The intersection (OR) or Join (AND) of the query caches.

### Note

Filter caches determine your result set.

- **OR Connections:** The DocumentID sets combine.
  - **AND Connections:** The DocumentID that appears in both sets become your result set.
5. The documents return to the documents cache based on the result set.

6. Any EVAL functions used in the query execute on each document and add modifications to the document cache.
7. Any Transforming Operator used in the query executes on the document set as a whole.
8. Any Data Processing Operators used in the query run against the remaining cache.
9. The final result set renders into a human-readable view on the Spotter UI.

## Query Terminology

The following table defines basic query terminology:

| Query Terminology   | Description   |
|---------------------|---|
| Search Terms        | <p>Forms a query and retrieves documents for further processing.</p> <ul style="list-style-type: none"><li>• Each Field-Value pair is a single search term.</li><li>• Logical operators link multiple search terms together.<ul style="list-style-type: none"><li>• <b>AND</b>: Ensures both search terms are matched within a document.</li><li>• <b>OR</b>: Ensures one of the search terms are presents within a document.</li></ul></li><li>• Parenthesis is used to link a grouping of search terms that must process together</li></ul> |
| Streaming Operators | <p>Executes an action on each document returned.</p>  |

| Query Terminology         | Description  |
|---------------------------|--|
|                           | <ul style="list-style-type: none"><li>• Includes all EVAL functions.</li><li>• Multiple streaming operators can be used in a single query, separated by a pipe ( ).</li></ul>  |
| Transforming Operators    | <p>Form the result set into structured data for visualizations.</p> <ul style="list-style-type: none"><li>• Typically, transforming operators are statistical and chart functions.</li><li>• Only one transforming operator is allowed.</li></ul>  |
| Data Processing Operators | <p>Performs action on the whole set, whether or not the data is transformed.</p> <ul style="list-style-type: none"><li>• Typically, data processing operators are <b>Order by</b> and <b>Where</b> type operators.</li><li>• Multiple data processing operators are used in a single query, separated by a pipe.</li></ul> |

When a search executes, the system returns a specific number of matches to the UI and continues to count all relevant matches.

- **HOT:** 1,000 events returned
- **Archive:** 300 events returned

### Note

The fewer the number of matches, the faster Unified Defense SIEM displays results on the UI.

## Query Syntax

Unified Defense SIEM uses the following key elements to filter data:

- **Field:** Terms used to identify the attribute to search within the Spotter indexes.
- **Operator:** Functions within the Spotter query language. The operators represent a comparison of value that are stored for a field compared to the user input value.
- **Value:** Pieces of data (a string, date, or number) used to compare against a field to get a list of events.

Combine these elements to create the search query. For example, the following query combines a field, operator, and value:

### Example

```
accountname = John.Doe
```

Use this query to find all the events in which John.Doe appears as the account name. Accountname is the field, = is the operator, and John.Doe is the value.

## Query Syntax Rules

Use the following query syntax in the search string to refine your results:

- Standard Classless Inter-Domain Routing (CIDR) is accepted for IP address field values.
- Search is not case-sensitive. For example, the following queries are all equivalent:
  - AdMin
  - admin
  - ADMIN
- Wildcards are special characters that stand in for unknown characters in a text value. The following rules apply to wildcards:

- To perform a single character wildcard search, use a question mark (?) in place of the single character you wish to replace.
- To perform a multiple-character wildcard search, use an asterisk (\*) to look for zero or more characters.
- Wildcards (?) and (\*) can be used in phrases.
- The wildcard character must be next to a letter or word.
- Standalone wildcard characters are not supported.
- Wildcard character support must be enabled in the administration console.
- Double quotation marks (") indicate that a document must contain the exact phrase within the double quotation marks for a match to occur. The following rules apply to double quotation marks:
  - A phrase or value that contains white spaces, commas (,), brackets ([]), pipes (|), punctuation marks, and special characters must be enclosed in double quotation marks.

**Example**

"phrasevalue"

- Use if you want to search for a value other than the default value, such as "AND", the Boolean operator, or the multiple character wildcard (\*).
- Numbers do not require double quotation marks.
- Each operator must be separated by a comma.

**Example**

"Operator 1", "Operator 2", "Operator 3",



- **Unknown** is a special keyword in the Spotter query language. The following rules apply to unknown keywords:
  - The **= unknown** attribute is the same as the null attribute.
  - The **!= unknown** attribute is the same as the not null attribute.
  - To search for an unknown string value, use double quotes before and after the string value.

**Example**

"unknown"

- Parenthesis ( ) and brackets [ ] are used interchangeably to group search terms together for processing.

**Example**

index = activity and accountname = john.doe and (deviceaction != blocked or deviceaction!= block)

**Note**

Curly brackets { } are not allowed and cause the system to return unexpected or 0 results.

## Search Indexes

Search indexing is the process by which the Spotter search engine organizes information before a search to enable fast responses to queries. When a search is performed, the Spotter search engine looks in the search index to find and return documents that match the query.

The following table describes each search index and its description:

| Index       | Description  |
|-------------|--|
| Activity    | Searches security log events from Windows, proxy devices, firewalls, DLP, IDS/IPS, etc. This is the default index.   |
| Activelist  | Search for entries stored in active lists.   |
| Archive     | <p>Searches for historical (warm) data.</p> <ul style="list-style-type: none"> <li>• Slower than searching hot data.</li> <li>• By default, results are in random order. To order results by eventtime, add a data process operator such as ORDERBY as follows:   orderby eventtime description</li> </ul> <div> <p><b>Note</b></p> <ul style="list-style-type: none"> <li>• Ordering results by eventtime delays the return of results, proportional to the amount of data scanned.</li> </ul> </div> |
| Asset       | Searches the ingested Entity/Asset Metadata for assets.  |
| Geolocation | Searches Geolocation correlated to IP addresses.   |
| Lookup      | Searches entries added to Lookup tables.   |
| Riskscore   | Searches current risk score for entities   |

| Index                          | Description  |
|--------------------------------|--|
| Riskscorehistory               | Searches for historical risk scores for entities                                     |
| Third-Party Intelligence (TPI) | Searches for threat intelligence ingested from third-party sources                   |
| Users                          | Searches user information ingested through files or Identity Access Management tools |
| Violation                      | Search for policy violations associated with an entity and a risk score              |
| Watchlist                      | Searches for entities added to a Watchlist.  |
| Whitelist                      | Searches for entities added to a Whitelist.  |

**Tip**

To prevent a user from searching a specific index in Unified Defense SIEM 6.4, apply granular Role-Based Access Control (RBAC).

**Common Fields**

The following table describes the common fields and their descriptions.

**Note**

The **indexedat** field is not available in Unified Defense SIEM. As an alternative, users must use **receivedtime/publishedtime**.

| Common Field   | Description   |
|----------------|---|
| eventid        | Unique identifier for each event ingested into the system. This is primarily used for counts in charts.   |
| eventtime      | The time the event occurs (reported by the resource).   |
| generationtime | <p>The time Unified Defense SIEM detects a violation. The <b>generationtime</b> field appears in the following indexes:</p> <ul style="list-style-type: none"><li>• Violation</li><li>• Risk score</li><li>• Risk score history</li></ul> |
| index          | The index to search. If an index is not specified, Spotter searches the Activity index.   |
| indexedat      | The time the event is indexed.  |
| policyname     | The policy name for which violations have been observed.  |
| policyuniqkey  | Unique identifier for each violation detected into the system. The <b>policyuniqkey</b> is primarily used for counts in charts.   |
| publishedtime  | The time the event was published to the RIN   |

| Common Field      | Description   |
|-------------------|---|
| receivedtime      | The time the event was received from the RIN.                           |
| resourcegroupname | The data source name that was provided during the import configuration. |

## Global Search

The global search bar appears at the top of every screen in Unified Defense SIEM. Enter a keyword in the global search bar to search for that value across Unified Defense SIEM. Global search executes a contained search and does not support the use of comparison operators.

An admin can configure the fields and the time range a search is executed against. **Last 1 Hour** is the default time range.

The following examples are valid searches in the global search bar:

- Logon - Search for string which contains a single word Logon
- "Logon Failure" - Search for string matching exactly to "Logon Failure"
- Logon Failure - Search string which contains 'Logon Failure'

The following examples are invalid searches in the global search bar:

- "Logon or "Logon Failure : unbalanced quotes
- Logon OR Failure : Either/or functionality is not supported at this time

## Keyword Searches

Enter a keyword or string of text into the search bar to search for that value with or without a field comparison.

Note the following when using keywords to search:

- Executes a contained search.
- Fields to be searched are configurable by an admin..
- **index = index\_name** specifies index to search. If not specified, activity index is searched.
- **AND** is not needed between index and keyword, as it will be read as a keyword and break the query

Examples:

- john
- index = violation OS1072
- Multiple keywords are not Supported
- Enclosing keyword in quotes will return incorrect results.

## Query Optimization

Query optimization makes your queries run as efficiently as possible. With query optimization, queries are sent through an optimizer before execution. The task of the optimizer is to create an initial execution plan for the query, look for optimization opportunities, and then apply them. As a result, the optimizer may produce multiple execution plans for a single query and will choose the plan that is considered to be the optimal plan, which is then executed.

This section covers the following information:

- [Guiding Principles for Query Optimization](#)
- [Indexes, Queries, and Resource Groups](#)
- [Frequently Used Queries vs. Optimized Queries](#)
- [Make Queries More Efficient](#)

- [Types of Searches](#)
- [Tips for Tuning Searches](#)
- [Tips to Limit Scanned Data](#)

## Guiding Principles for Query Optimization

If the query is not optimized, performance can become an issue. Having long-running queries not only consumes system resources causing the application to run slow, but may also cause the application to crash. The following are basic guiding principles for query optimization:

- Retrieve only the required data.
- Move as little data as possible.
- Set appropriate time windows.

The way to increase query performance is to use the simple search to filter as much as possible, perform joins and look-ups only on the required data, and perform evaluations on the minimum number of events.

## Indexes, Queries, and Resource Groups

The construction of a search has a significant impact on its performance. Generally, including a small number of events in your search processing results in the best performance.

When data is indexed, the data is normalized and enriched into usable events based on time. The processed data is stored as follows:

- Current On-Prem and Cloud:
  - Most Recent activity events are stored in hot storage, excluding the raw event (index = activity)
  - All activity events, including the raw events, are stored in warm storage (index=archive)
- Bring Your Own Snowflake (BYOS) and Unified Defense SIEM powered by Snowflake:

- All activity data is stored and searchable from a single index (index = activity).

On-Prem warm storage is processed once a day, so index=archive is typically 24 hours behind index=activity.

### Use Indexes and Resource Groups Effectively

When a search does not need to cross multiple resource groups, restrict your searches to specific resource group name.

## Frequently Used Queries vs. Optimized Queries

A frequently used query can consume a significant amount of system resources. This section describes the difference between frequently used queries and optimized queries.

### Frequently Used Queries

Often times, frequently used queries are excessive. For example, the following query contains many elements, followed by STATS:

```
ipaddress in CSV_GROUP_of_20_IPs and destinationport = 443 and categoryoutcome
```

Follow along with the diagram below to see what occurs when this query is used:



1. When the query is run, the activity index is accessed and the index is searched 20 times (once for each IP in the group).



- 20 \* size of index = 1,000,000 extracted

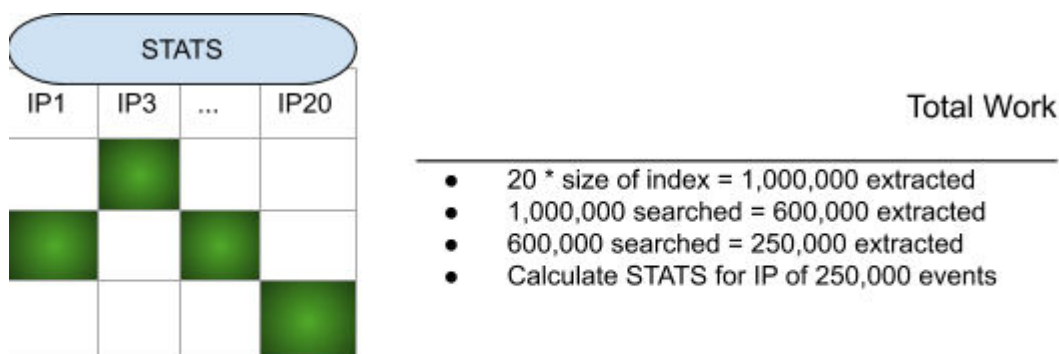
[illegible]

- 20 \* size of index = 1,000,000 extracted
- 1,000,000 searched = 600,000 extracted

[illegible]

- 20 \* size of index = 1,000,000 extracted
- 1,000,000 searched = 600,000 extracted
- 600,000 searched = 250,000 extracted

4. Finally, the STATS command is executed to output the results with a count of the event.



The query uses a lot of resources and is inefficient to complete.

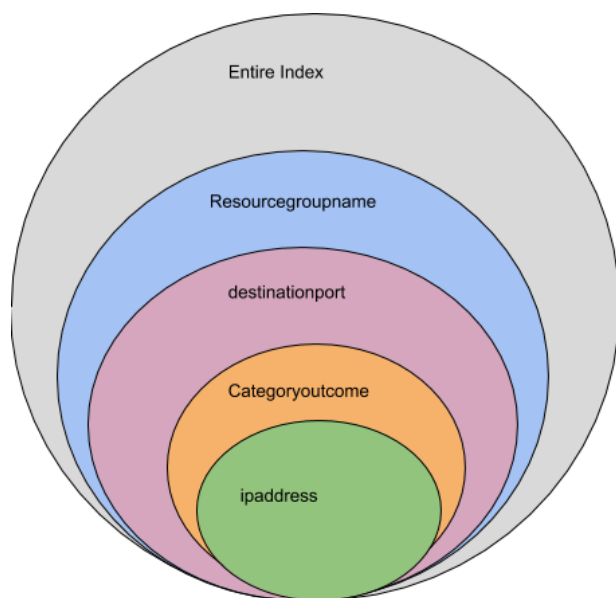
### Optimized Queries

You can optimize the search by adding a resource group and moving the terms to locations earlier in the query process. Adding a resource group to the front of the query will reduce the number of events that cross the index. Additionally, when you move a set with less elements forward, you reduce the number of events that each successive term must search through to locate results. This is important for **OR** and **IN** queries.

#### Note

Only add a resource group to the front of a query when you do not wish to search across multiple resource groups.

The following diagram shows identical results produced, while using less resources due to not searching across the index twenty times for the **IN** statement:



#### Total Work

- Query Entire Index for resourcegroupname = 750,000 extracted
- 750,000 queried for destinationport = 600,000 extracted
- 600,000 queried for categoryoutcome = 300,000 extracted
- 300,000 queried for ipaddress \* 20 = 250,000 extracted
- Calculate STATS for IP of 250,000 events

The optimized query:

```
resourcegroupname = firewall and destinationport = 443 and categoryoutcome
```

## Make Your Queries More Efficient

A query may be slow if it is too complex to retrieve events from the index. For example, if your query contains large **OR** or **IN** lists and types of phrase searches, it takes longer to process. To improve the performance of a query, limit the data to be scanned to an absolute minimum, and filter the data as early as possible in the query so that processing is done on the minimum amount of data necessary.

This section examines the causes of slow queries and includes guidelines to help you write queries to run more efficiently. Many factors can affect the speed of your query, including, but not limited to:

- The volume of data you are querying.
- The way searches are constructed.
- The number of concurrent queries.

## Types of Searches

The recommendations for optimizing queries depend on the type of query that you run and the characteristics of the data you are searching. Based on what you want to accomplish, queries fall into one of the following categories:

- Queries that retrieve events.
- Queries that generate a report that summarizes or organizes the data.

### Queries that Retrieve Events

Simple queries retrieve events from an index, typically activity, without additional processing of the retrieved events. Use field-value pairs that are unique to the events when events are retrieved from the index.

- **Dense query:** Retrieved events occur frequently in the data set.
- **Sparse query:** Retrieved events are rare in the data set. Sparse queries that run against large volumes of data take longer than dense queries against the same data set.

### Queries that Generate Reports

Report-generating queries, or transforming queries, perform additional processing on events after events are retrieved from an index. This processing can include filtering, transforming, and other operations using one or more statistical functions against the set of results. Because the processing occurs in memory, and the more restrictive and specific you are retrieving the events, the faster the search will run.

## Tips for Tuning Searches

In most cases, a query is slow due to the complexity of the query. For example, if your query contains extremely large **OR/IN** lists and types of phrase searches, it takes longer to process. This section includes tips to tune your search to be more efficient.

Performing statistics on a set of field values that have a high cardinality (uncommon or unique values) requires a lot of memory. Reducing the number of distinct values that must be processed is beneficial.

## Factors that Impact Search Performance

| Customer Controlled  | Uncontrollable   |
|--|--|
| <ul style="list-style-type: none"> <li>• Search type (use case)</li> <li>• Search complexity</li> <li>• Search period</li> <li>• Search load</li> <li>• Number of masked attributes</li> </ul> | <ul style="list-style-type: none"> <li>• EPS/GB ingestion rate</li> <li>• Frequency of matchable data</li> <li>• Masking compile time</li> </ul> |

### Customer Control

- **Search Type:**
  - Event retrieval does not need to scan all data to complete.
  - Aggregation queries must scan all data to get the correct results.
- **Search Complexity:** Has various effects on completion times.

| Comparison   | Efficiency Grade | Comments  |
|--|------------------|---|
| <ul style="list-style-type: none"> <li>• = [Equals]</li> <li>• In/Not In</li> <li>• Null/Not Null</li> </ul> | A+               | The most optimized comparisons. These comparisons use the exact match and existence checks.                             |
| <ul style="list-style-type: none"> <li>• != [Not Equals]</li> <li>• Not In</li> </ul>                        | A-               | Returns more data than the previously mentioned comparison, meaning more resources are required to complete the search. |

| Comparison   | Efficiency Grade | Comments   |
|--|------------------|--|
| <ul style="list-style-type: none"> <li>• Starts With</li> <li>• Ends With</li> </ul>   | B+               | A character-to-character match-up to the number of characters used in the string for search. This increases the time to process any search on the attribute. |
| <ul style="list-style-type: none"> <li>• Before</li> <li>• After</li> <li>• Between</li> <li>• CIDR Ranges [In]</li> <li>• &lt; [Less Than]</li> <li>• &lt;+ [Less Than or Equal To]</li> <li>• &gt; [Greater Than]</li> <li>• &gt;+ {Greater Than or Equal To}</li> <li>• Time Modifiers</li> </ul> | B-               | Forms a range on an attribute, meaning the execution doubles to ensure a match.  |
| <ul style="list-style-type: none"> <li>• CIDR Ranges [not in]</li> <li>• Not Between</li> </ul>  | C+               | Forms a range using an exclusion, often with more results returned. This leads to higher execution times.  |

| Comparison   | Efficiency Grade | Comments   |
|--|------------------|--|
| <ul style="list-style-type: none"> <li>• Not Starts With</li> <li>• Not Ends With</li> </ul> | C-               | Returns more results than their counterparts, which leads the query to require more resources for completion.  |
| Contains [wildcards]   | D                | Leverages a character-to-character match-up of the entire string for a significant increase in time to completion. (One of the least optimal comparisons). |
| Not Contains   | F                | Returns more results and is executed in a character match. (The least optimal comparison).   |

- **Search Period/Range selected**

- **Small Range:** Faster response times by reducing the data for the search to scan.

**Example**

Last 1 hour or last 7 days.

- **Large Range:** Slower response time by increasing the amount of data to scan.:

**Example**

Last 30 days, custom range > 30 days.

- Users have access to a larger search period.
- The system will continue to run until the feature limit is hit.

- **Masked Attributes:** The higher the number of masked attributes to process, the longer the load/generation time is for queries.
  - Privacy Master can see and search data unmasked, allowing them to see faster performance.
  - The Table Operator, or column selection in reports, allow users to reduce the number of masked columns for processing. This leads to faster performance.
- **Search Load (Resource Contention):** Split across concurrent queries on the same compute.
  - The Search Load is monitored by Securonix. If contention is observed, Securonix will split the compute as needed.
  - Unified Defense SIEM introduces auto-scaling of compute nodes.

#### Uncontrollable

- **Data Ingestion Rate:** During a given time period, the platform will experience spikes and lulls that can change the amount of data that is processed.
- **Frequency of Matchable Data:**
  - **High-Frequency:** Fast response times.
  - **Low-Frequency:** Low response times.
    - Needle-in-haystack searches are low-frequency searches.
    - Low EPS datasources are low-frequency searches.
- **Masking Compile Time:** The time it takes to process and mask the configured fields.

#### Place Lower Cardinality fields up front

A simple query structure should follow the least cardinality to most cardinality approach. This means the fields with the least possible values should be put up front. The lower cardinality of the possible values translates to faster completion times. As the cardinality increase, so to does the time it takes to complete the search term check.

- IP address = 4,294,967,296 possible



- countries = 195 possible
- day of year = 365 day (366 on Leap year)
- **Recommended order of search execution:** resourcegroupname > countries > dayofyear > ipaddress
- **Search:**

```
resourcegroupname = "firewall" and country in (Russia, Philippines, La
```

This also holds true for the number of values you are searching. When you want to apply an **OR/IN** comparison on of multiple fields, the smaller set of values should come first, and, where possible, apply the lower cardinality fields first.

- **countries:** UK, US, Latvia, Italy, Germany, Russia
- **Accountnames:** Bob, Joe, Sue, Jim
- **destinationports:** 22, 25, 445
- **Recommend order of search execution:** resourcegroupname > destinationport > accountname > countries
- **Search:**

```
resourcegroupname = "firewall" and destinationport IN (22, 25, 445) an
```

### The AND before OR Clause

To improve the logical evaluation of a query, place the **AND** clause before the **OR** clause. Each **AND** clause reduces the data set scanned by the **OR** clauses, which speeds up the return process.

- **Inefficient query:** This following query will search the entire firewall resource for each IP before checking for accountname:

```
resourcegroupname = "firewall" and (ipaddress = 10.10.1.1 or ipaddress
```

- **Efficient query:** The following query searches the entire firewall resource for each ip before then checking for accountname:

```
resourcegroupname = "firewall" and accountname = john.smith and (ipadd
```

This query focuses the search to only firewall events for **John.smith** before checking IPAddress.

### AND over OR with != and NOT CONTAINS

In addition to placing **AND** before **OR**, or when using **!= (Not Equals)** and **NOT CONTAINS** comparisons, you should use **and** between the conditions.

### Inefficient

The following query returns all results in the violation index and burns through a large percentage of resources available. This is do to the **OR** clauses canceling each other out instead of reducing the data set during search:

```
index=violation and policyname not contains "email" or policyname not cont
```

### Efficient

The following query removes both instances and reduces the data, using less resources:

```
index=violation and policyname not contains "email" and policyname not con
```

This query is a "not this one" and "not that one" scenario.

## Tips to Limit Scanned Data

The techniques to limit the amount of data scanned on the disk range from setting a narrow time frame to being as specific as possible and retrieving the smallest number of events necessary.

### Narrow the Time Frame

One of the most effective ways to limit the data that is scanned is to narrow the time frame. Use the time frame selector or specify time modifiers in the search to identify the smallest window of time necessary for your query.

### Specify the Index and/or Resource Group Name

Understanding how your data is organized is important to optimizing your searches. Take the time to learn which indexes contain what type of data and which resource group names contains a particular type of events. Knowing this will help you narrow down your searches.

Whenever possible, specify the index and/or resource group name in your query. When Unified Defense SIEM indexes data, it automatically tags each event with a number of fields. The index and resource group name are then added to each by default.

### Note

Searching across more than one data source is exponential in increasing the amount of data to be searched. As mentioned above, specify the resource group name where possible. If you cannot specify the resource group, exclude it from the search altogether for a slight performance improvement.

Forexample:

```
resourcegroupname = * and accountname = john.smith
```

or

```
resourcegroupname NOT NULL and accountname = john.smith
```

Both expend resources just to validate searching all resources. Instead, start with the first field: `accountname = john.smith`, which skips the validation and reduces the overhead used for the search.

### Be Specific

Use specific terms in your query. If possible, avoid using wildcard characters and `contains`. For example, instead of using a wildcard character: `eventoutcome = *success*`, use a specific string: `eventoutcome = "Audit success"`. Exact matches, starts with, and ends with are much faster to complete than wildcards and `contains`.

### Specific and AND before Wildcards, CONTAINS, OR / IN, and NOT

Wildcards such as `*`, `CONTAINS`, `NOT CONTAINS`, `OR`, `IN`, and `NOT` clauses are slower to process within the system. To improve search performance, move these items to the end of your simple query so they process against the smallest data set.

Instead of putting these statements first: `(ipaddress contains 192.168 or ipaddress = 172.168.1.*) and accountname = john.smith and eventoutcome starts with Accept`

Place the specific AND first: `accountname = john.smith and eventoutcome starts with Accept and (ipaddress contains 192.168 or ipaddress = 172.168.1.*)`

**Exception:** when the **OR** / **IN** clauses are for the resourcegroupname, it should still come first, as this is the largest reducer of data scanned.

### Avoid Using NOT Expressions

More resources are used to track the **NOT** expression than if you specify what you are looking for. Where possible, avoid using **NOT** expressions. For example, instead of using **!=**, **NOT IN**, **NOT CONTAINS**, and others, such as: **ipaddress != a or ipaddress != b** or **ipaddress NOT IN a,b**.

Use the specific terms you are querying for: **ipaddress = c or ipaddress = e**

### Filter Unnecessary Fields from Query Results

You can remove unnecessary fields from the query results by using command **FIELDS**.

### Join Indexes only as Needed

The **FILTER** operator allows users to join multiple indexes. This doubles the time a search takes due to having to cross the index multiple times before combining the data for matching records, similar to using **OR** or **IN** clauses.

### Apply Limits with Transform Operations

Performing aggregation on high cardinality fields uses a large amount of resources to complete. To avoid overhead, **RARE** and **TOP** have a **LIMIT = #**, which can be applied. **HEAD #** is also available. Both restrict the result set to the number specified.

You should avoid combining **HEAD #** with **RARE** and **TOP** if using **LIMIT = #**, as it restricts to the first set of values found, whereas limit is based on count.

## Spotter Search Examples

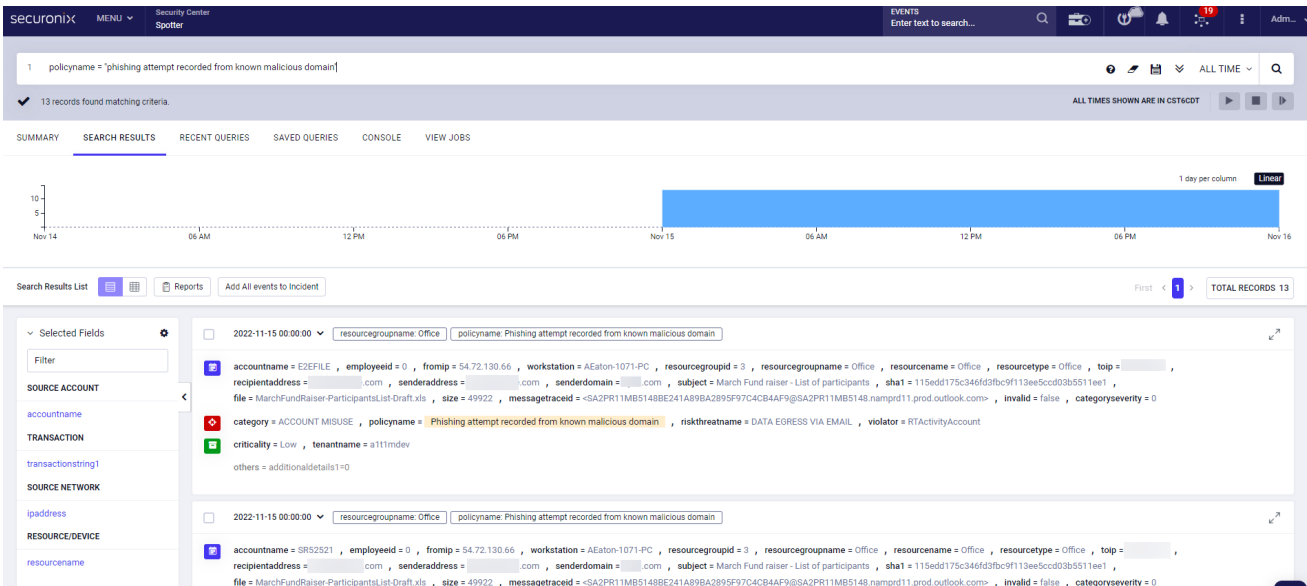
This section includes Spotter search examples.

### Note

All the images in this section are for illustrative purposes only and may differ from the actual product due to product enhancements.

## Example 1: Find Policy Violations

To view all the violations of a particular policy, use the syntax as in the following example: `polycname = "phishing attempt recorded from known malicious domain"`.



## Note

To query data using Securonix attributes, use the "@" prefix, then enter the Securonix attribute. The "@" indicates that you are using Securonix attribute to run a query.

### Example 2: Check if user has sent email to personal email address

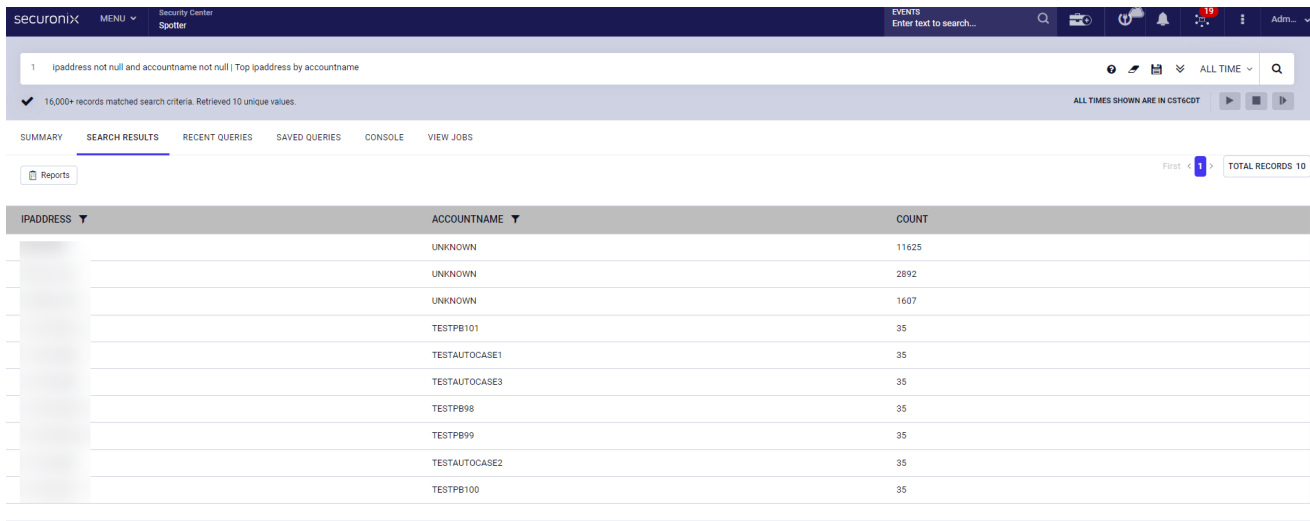
To view check if a user has sent email to a personal email address, the following query uses the CondensedStringMatcher: `workemail not null | EVAL matchPerc = emailtoself(accountname,workemail,0.02)`.

### Example 3: Find Asset Data

To view all the devices in the asset index, use the following query: `index = asset`.

### Example 4: View Top IP Address by Account Name

To view a table with the top IP addresses by account name for a datasource, use the following query: `ipaddress not null and accountname not null | Top ipaddress by accountname`.



| IPADDRESS | ACCOUNTNAME   | COUNT |
|-----------|---------------|-------|
|           | UNKNOWN       | 11625 |
|           | UNKNOWN       | 2892  |
|           | UNKNOWN       | 1607  |
|           | TESTPB101     | 35    |
|           | TESTAUTOCASE1 | 35    |
|           | TESTAUTOCASE3 | 35    |
|           | TESTPB98      | 35    |
|           | TESTPB99      | 35    |
|           | TESTAUTOCASE2 | 35    |
|           | TESTPB100     | 35    |

## Example 5: Search threat intelligence for top countries of origin

To find the top TPI domains from which threats originate in the Third Party Intelligence index, use the following query: `index = tpi | TOP tpi_domain`




| TPI_DOMAIN                                     | COUNT  |
|--|--------|
| null   | 421943 |
| bodyres.f3922.net                              | 2      |
| cnwx.58ad.cn                                   | 2      |
| inidown.com                                    | 2      |
| lyxyjaj.com                                    | 2      |
| -sharepoint.com                                | 1      |
| 000webhostapp.com                              | 1      |
| 004f67c79b428da67938dadc0a1e1a4                | 1      |
| 0129e158-a17-4900-99a6-90f4a49bd0a4.nordit.com | 1      |
| 020-danger.r1z.rocks                           | 1      |

## Executing Bulk Export Reports in Spotter

- The maximum output is one million events.
- The output is downloaded as a .zip file.
- The output contains multiple CSV files based on the configuration size.

- The file is only available for seven days.
- The current CSV size is configured to support uploads to Google Sheets or Office 365.

## To Execute Bulk Export Reports in Spotter

1. In Unified Defense SIEM, go to **Menu > Security Center > Spotter**.
2. In the Spotter search bar, execute a query.
3. Click **Reports > Bulk Export to Email**.
4. Select one or more event attributes.
5. Click  to move the event attributes to the right column.
6. In the **Choose Email Template** field, select **Bulk Data Report**.
7. Search and select an owner.
8. Click **Assign**.
9. Click **Export Records**.
10. To download the Bulk Export report, access the link in your email, or use the notification bell in Unified Defense SIEM.

## Running Reports from Spotter

You can run and export reports from the **Search Results** view in **Spotter**.

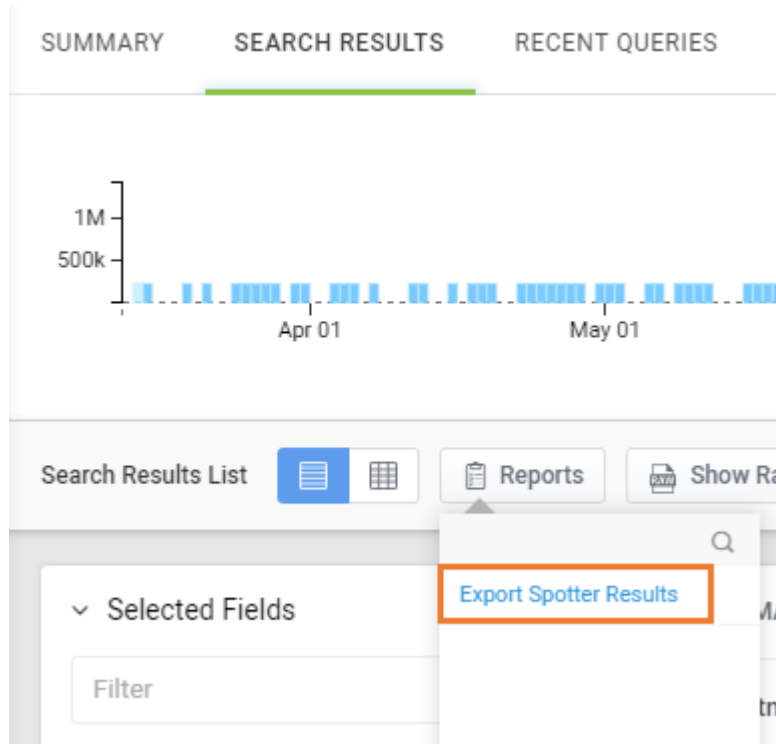
### To Run Reports from Spotter

1. Go to **Menu > Security Command Center > Spotter**.



2. Click the **Search Results** tab, then select **Reports > Export Spotter Results**.

The option to **Export Spotter Results** appears in the drop-down, along with the names of any Spotter reports configured from **Menu > Reports > Categorized Reports**. To learn how to configure Spotter reports, see [Reports](#).



The **Run Spotter Report** window appears.

3. Click **Select Report Format** drop-down and select a format for the report:
  - PDF
  - CSV
  - XLS
  - RTF
  - TXT

- DOCX

### Note

When disabled, the toggle below the **Select Report Format** only exports 1,000 events in the report.

4. Check the box next to each attribute to be included in the report, then click the right arrow highlighted in blue. Attributes that appear in the **User Attributes** column are included in the report.

The screenshot shows the 'SELECT ATTRIBUTE' interface. On the left, a list of attributes is shown with checkboxes: 'important' (checked), 'delay' (checked), 'index' (unchecked), 'tenantname' (unchecked), 'Other' (unchecked), 'agentfilename' (unchecked), and 'eventid' (unchecked). A blue right arrow button is highlighted. On the right, the 'Users Attributes (Securonix Attributes)' table shows the selected attributes mapped to report labels. The 'Report Label Column' header is visible. The table contains the following rows:

| Users Attributes (Securonix Attributes)    | Report Label Column |
|--|---------------------|
| <input type="checkbox"/> accountname       | accountname         |
| <input type="checkbox"/> sourceport        | sourceport          |
| <input type="checkbox"/> resourcegroupid   | resourcegroupid     |
| <input type="checkbox"/> resourcegroupname | resourcegroupname   |
| <input type="checkbox"/> rg_functionality  | rg_functionality    |
| <input type="checkbox"/> rg_vendor         | rg_vendor           |

### Note

By default, when the **STATS** or **TOP** table queries are used, the attributes display in the **User Attributes (Securonix Attributes)** box.

### Tip

To select all attributes, check the **Select All** box in the column header.

5. Click **Schedule** to save the label and include the attribute in the report.
6. Click **Run** to run the report and download the report from the **Notifications** menu when status is complete.

**EVENTS**  
Enter text to search...

**Notifications**

Enter min 3 characters to search

Last year

[Securonix-gitlab1.securonix.com:03/12/2021 03:40:05 -0500] Ingestor started  
11 HRS AGO

[Securonix-gitlab1.securonix.com:03/12/2021 03:36:28 -0500] Ingestor started  
11 HRS AGO

[Securonix-gitlab1.securonix.com:03/12/2021 01:18:54 -0500] Ingestor started  
13 HRS AGO

Policy with ID 845 deleted  
2 DAYS AGO

Policy with ID 114 deleted

[CLEAR ALL](#) [MORE](#)

## Executing Search-Based Reports

When executing search-based reports, performance and response times will may vary compared to the interface, based on the following:

- **Query Limit Varies:** The system is optimized to return a subset of results to the interface, with a limit based on the type of query and operators, plus a total count. The reporting system will set the limit as follows:
  - **Scheduled Reports:** Defaults to the maximum export amount (100,000) for the platform.
  - **On-Demand Reports:** On-Demand reports from Spotter, or the Violation Event page, will use the lowest number between the total count matched and the maximum export amount.

- **Re-execution for Data Retrieval:** The interface returns a subset of matches and a count. To get data to the maximum export, the report system must execute a new search to retrieve the data.
- **Increased Data Scans:** The reporting system is optimized to return the maximum results. The report may need to scan the maximum amount of data to achieve this goal, while the interface scans smaller segments.
- Searches with low data matches compared to the quantity of data crossed will have higher data scan rates and take longer to complete.

## Spotter Metrics

The screenshot shows the Securonix Spotter Metrics interface. On the left, a 'Category' sidebar lists various categories, with 'Spotter Metrics' at the bottom. The main content area displays a table of reports. The table has two columns: 'Report Name' and 'Actions'. The reports listed are 'Execution Details Report', 'Execution Time Per User', and 'Spotter Capacity Graphical Report'. Each report has a brief description and a set of action icons (edit, refresh, delete, download). The top navigation bar includes 'Reports' and 'Categorized Reports' tabs. The bottom of the main panel shows pagination controls: 'First < 1 > Last Show 10' and 'Total results : 3 | Total pages : 1'.

| Report Name   | Actions                              |
|---|--------------------------------------|
| Execution Details Report<br>This report presents essential details on executed queries, including query ID, user, execution time, start and end time... | [Edit] [Refresh] [Delete] [Download] |
| Execution Time Per User<br>This report summarizes user-wise query execution details, including the number of queries and total execution time.          | [Edit] [Refresh] [Delete] [Download] |
| Spotter Capacity Graphical Report<br>This report presents a graph of total execution time vs days   | [Edit] [Refresh] [Delete] [Download] |

First < 1 > Last Show 10 Total results : 3 | Total pages : 1

## How to Get There

1. In Unified Defense SIEM, got to **Menu > Reports > Categorized Reports**.
2. From the **Category** pane, select **Spotter Metrics**.

## Out-of-the-box Spotter Metrics Reports

This section explains the Spotter report metrics of the search system to gain more oversight of system quality.

- Search history is maintained for 90 days in the app.
- The Spotter Metric report will only show queries that pass.

### Execution Details Report

The Execution Details Report shows the historical search execution data. Each query shows the following information:

- The query ID.

#### Note

The query ID is provided in report tickets and performance/concern tickets to allow users and Securonix to focus on the same queries.

- The query.
- The user who executed the search.
- The end-to-end execution (completion time).
- The searched time range.
- The start and finish time for the search.

#### Note

Only admins can access the Execution Details report. The report contains the query, which can cause unauthorized access in the case of granular role-based access control/tenant access (depending on the configuration).

### Execution Time per User Report

Shows the aggregated end-to-end execution time and the count of queries that went into the aggregation.

### Spotter Capacity Report

Shows the end-to-end execution time aggregated by day-to-day usage of the search system.

### Search History Overview Report

The Search History Overview report shows an aggregated view of a users search history based on a specified date range. This can include success and failure query counts, workloads, and search ranges and response time distributions.

### Stats Chart

The Stats chart shows the total number of searches executed within a specified date range. The Stats chart categorizes queries into successful and failed executions, visually showing the ratio of success to failed query executions.

Table with format:

- Success | Count | Percentage
- Failure
- Total

### Search Workload by Index Chart

The Search Workload by Index is a pie chart that shows the distribution of queries across different indexes. It highlights the workload on each index, helping identify which indexes are most frequently queried.

### Chart: Total Execution Time by Index

The Total Execution Time by Index chart shows the execution times for queries segmented by index. This helps to identify the amount of time spent per index.

Table with format:

- Index | Execution time | Percentage

- Grand Total Execution Time

### **Chart: Distribution of Time Range Searched**

The Distribution of Time Range Searched is a pie graph that illustrates the distribution of searches based on the time range queried. This chart shows the time periods most frequently queried.

The Distribution of Time Range Searched chart has the following categorization:

- Less than 1 Hour
- 1 Hour to 24 Hours
- 24 hours to 7 days
- 7 days to 30 days
- Greater than 30 days

### **Chart: Distribution of End-to-End Execution Time**

The Distribution of End-to-End Execution Time pie chart displays the distribution of searches based on their end-to-end execution time (also known as completion time). This chart compares the number of queries that completed fast and the number of queries that completed slow.

The Distribution of End-to-End Execution Time chart has the following categorization:

- Less than 30 seconds
- 30 seconds to 1 minute
- 1 minute to 5 minutes
- 5 minutes to 10 minutes
- 10 minutes to 30 minutes
- Greater than 30 minutes

### **Chart: Query Execution Time Compared to Time Range Searched**

This data establishes a matrix of the percentage of searches executed for each time range searched based on the end-to-end execution time.

Columns have the following categorization:

- Less than 30 seconds
- 30 seconds to 1 minute
- 1 minute to 5 minutes
- 5 minutes to 10 minutes
- 10 minutes to 30 minutes
- Greater than 30 minutes

Rows have the following categorization:

- Less than 1 Hour
- 1 Hour to 24 Hours
- 24 hours to 7 days
- 7 days to 30 days
- Greater than 30 days

# Spotter Search Help

This section includes the common natural language search commands for Spotter, including search, reporting, and analytical operators.

## Spotter Query Structure

Spotter query structure includes the following elements:

### Search Terms

Search Terms are the simple search parameters used to form a query. This is also called Simple Search.

- Field-Value pair is a single search term.
- Logical operators are used to link multiple search terms together:
  - **AND:** Indicates both(or multiple) linked search terms must be present in the data set.



- **OR:** Indicates at least one of the linked search terms must be present in the data set.
- Parenthesis ( ) can be used to group search terms that must be processed together.

## Example

**Syntax:** <field> <comparator> <value>

- **Field:** The field, or attribute, within the data.
- **Comparator:** The comparison, or condition, against which to match the value to the field.
- **Value:** The value of the field within the data.

**Search Term:** accountname = John.Doe

## Streaming Operators

Streaming Operators execute an action on the search results returned from the Spotter query.

- All EVAL functions are steaming operators.
- Multiple Streaming operators can be used in a single query separated by a | (pipe).

## Transforming Operators

Transforming Operators display the search results into different visual formats.

- Typical transforming operators include Statistical and Chart functions.
- Only a single transforming operator is allowed for a query.

## Data Processing Operators

Data Processing Operators perform an action on the whole set of search results, whether or not a transforming operator has been used.

- Typical data processing operators are those such as Order by and Where.
- Multiple Data Processing operators can be used in a single query separated by a | (pipe).

## Query Syntax

Spotter queries use the following syntax:

- Case-insensitive numbers do not require quotes.
- Strings that contain special characters and/or white space characters require quotes.
- Multiple values for operators such as "IN" and "NOT IN" require quotes as needed and must be separated by commas. Example: `accountname IN "jdoe", "jsmith"`.

For specific meanings of special characters, see [Specific Meanings](#) in *Using Special Characters in Spotter*.

## Indexes

Spotter uses natural language to search within the indexed data. You can search within any index into which you have imported data. Search uses the following indexes to store data:

- **Activity:** Used to search for current (hot) security log events from Windows, Proxy devices, Firewalls, IDS/IPS, etc.
- **Activelist:** Used to locate entries on active lists stored in Redis.
- **Archive:** Used to search for historical (warm) data.

### Note

Queries are slower for this index than for the other (hot) indexes.

- **Asset:** Used to search metadata for assets such as servers, workstations, laptops, ATMs, POS devices, etc.
- **Geolocation:** Used to search Geolocation correlated to IP Addresses.
- **Lookup:** Used to search for entries in lookup tables such as Competitor Domains, Non-Business Domains, etc.
- **Riskscore:** Used to search current risk scores for entities.
- **Riskscorehistory:** Used to search for historical risk scores for entities.
- **TPI:** Used to search for threat intelligence ingested from third-party sources such as ThreatSteam.
- **Users:** Used to search user information ingested through Identity Access Management devices, HR systems, etc.
- **Violation:** Used to search for policy and threat violations associated with an entity and a risk score.
- **Watchlist:** Used to search for entities added to a watchlist.
- **Whitelist:** Used to search for entities added to a whitelist.

### Note

By default, Spotter searches the Activity index.

### Note

Securonix Unified Defense SIEM introduces a single search engine for activity data removing the need to utilize references for the Activity and Archive indexes

## Common Fields

The following fields are commonly used in Spotter search:

- **eventtime:** Time the event occurred on the resource (datasource).
- **generationtime:** Time Securonix Cloud SIEM detected a violation. Appears only in indexes: violation, risk score, and risk score history.
- **index:** Specifies the index in which to search.
- **indexedat:** Time the event was indexed into hot storage.
- **policyname:** Used to search a specific policy name for which violations have been observed.

- **publishedtime:** Time event was published to Kafka.
- **receivedtime:** Time the enrichment job processed the event.
- **resourcegroupname:** Used to search for a data source by the specific name the it was given when the connection was created to import data.

## Search Operators

Search Operators tell the system how to locate, format, manipulate, and display the data you want to see. These include the following types of commands:

### Basic Commands

The following operators are used to form search terms in a query.

#### indexedat

The indexedat command finds events that are indexed within the specified duration.

#### Supported Variables:

- **MINUTES:** [NOW-5MINUTES NOW]
- **HOURS:** [NOW-5HOURS NOW]
- **DAYS:** [NOW-5DAYS NOW]
- **Last 10 Minutes:**[NOW-15MINUTES NOW-5MINUTES]

**Syntax:** indexedat BETWEEN (value1) (value2)

#### Example

```
resourcegroupname = Google-FileDS AND indexedat BETWEEN  
2018-10-05T00:00:00.000Z 2018-10-17T23:59:59.999Z;  
resourcegroupname = Google-FileDS AND indexedat BETWEEN  
NOW-15DAYS NOW
```

#### Policy

The Policy command searches for a specific policy to find violations. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <policyname> <=> <value>

**Example**

```
polycyname = "Accounts visiting Algorithmically Generated Domains-1"; polycyname = Logon_Failure
```

**Data Sources**

Queries the activity core for specific data sources. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <resourcegroupname> <=> <value>

**Example**

```
resourcegroupname = BCP1
```

**Text**

Returns all results that include the specified text.

**Syntax:** <value>

**Example**

```
smith
```

**\***

Multiple character wild card searches looks for 0 or more characters.

**Syntax:** <field1 | \*> <field2 | \*> <field n | \*>

**Example**

```
MM*; With Field : firstname = Ma*
```

**?**

To perform a single character wild card search use the "?" symbol.

**Syntax:** <field1 | ?> <field2 | ?> <field n | ?>

**Example**

```
??2497
```

## Index Commands

The following operators are used to specify the index in which to search.

**Lookup**

Searches within lookup index for all items added in lookup tables.

**Syntax:** index= < lookup > <and | or> <field> | Report Commands | Field Commands

**Example**

```
index = lookup; index = lookup and lookupname = betaSpotter
```

**Activity**

Searches within the activity index for events. This is the default index for Spotter searches.

**Syntax:** index = < activity > <and | or> <field> = <field value>

**Example**

```
index = activity; index = activity and accountname = secure;  
index = activity and deviceaction = 26952 and  
transactionstring1 = THREAT
```

**Violation**

Searches within the index for policy violations.

**Syntax:** index = < violation > <and | or> <field> = <field value>

### Example

```
index = violation; index = violation and violator = Users;  
index = violation and sessionid = 1102
```

### Riskcore

Searches within the riskcore index that stores all violators and provides riskcore card information.

**Syntax:** Index = < riskcore > <and | or> <field> | Report Commands | Field Commands

### Example

```
index = riskcore; index = riskcore and accountname =  
WHITE.DAVID
```

### Archive

Searches historical data **in warm storage**. You must specify resourcegroupname, resourcetype, or rg\_functionality, and tenantname in the query.

**Syntax:** index = < archive > <and> <resourcegroupname> <=> <value> <and | or> <field> = <field value>

### Example

```
index = archive and resourcegroupname = Google_login; index =  
archive and resourcegroupname = Google_login and accountname =  
AJAIS"@SEC.COM
```

### Whitelist

Searches within the whitelist core for entities in a global or targeted whitelist.

**Syntax:** index = < whitelist > <and | or> <field> = <field value>

### Example

```
index = whitelist; index = whitelist and entityname = 1115
```

## TPI

Searches within the TPI index, which stores third-party threat intelligence.

**Syntax:** Index = <tpi> <and | or> <field> | Report Commands | Field Commands

### Example

```
index = tpi; index = tpi and tpi_addr = zztxdown.com; index =  
tpi and tpi_srckey = zzshw.net_MalwareDomains
```

## Asset

Searches within the asset index, which stores device metadata.

**Syntax:** Index <asset> <and | or> <field> | Report Commands | Field Commands

### Example

```
index = asset; index = asset and entityname = resource98
```

## Watchlist

Searches within watchlist index for all watchlisted entities.

**Syntax:** Index = <watchlist> <and | or> <field> | Report Commands | Field Commands

### Example

```
index = watchlist; index = watchlist and watchlistitem_item2 =  
item2
```

## Users

Searches within the user index.

**Syntax:** index = < users > <and | or> <field> = <field value>



### Example

```
index = users; index = users and department = marketing
```

### Riskscorehistory

Searches within the riskscore card history index.

**Syntax:** Index = <riskscorehistory> <and | or> <field> | Report Commands | Field Commands

### Example

```
Index = riskscorehistory; index = riskscorehistory and  
accountname = SWIFT.JOHN
```

### Geolocation

Searches within the geolocation index for IP address.

**Syntax:** index = < geolocation > <and | or> <field> = <field value>

### Example

```
index = geolocation; index = geolocation and longitude = 9.491
```

### Activelist

Searches within the activelist index for entries found on active lists in Redis. The entries returned are historical in nature and due not status of the entry on the list.

**Syntax:** index = < activelist > <and | or> <field> = <field value>

### Example

```
index = activelist and activelistname =  
Suspicious_File_Download
```

## Comparators

The following operators compare a field to a value.

## CONTAINS

Returns results if a string field contains the specified value. Contains does not support Date attributes like hiredate, terminationdate, expirydate, etc. Contains is not case sensitive.

**Syntax:** <field> CONTAINS <value>

### Example

```
resourcegroupname = BCP1 and accountname contains securonix
```

## NOT CONTAINS

Returns results if a string field does not contain the specified value. This comparator does not support date attributes like hiredate, terminationdate, expirydate, etc. Contains is not case sensitive.

**Syntax:** <field> NOT CONTAINS <value>

### Example

```
resourcegroupname = BCP1 and accountname not contains  
securonix
```

## AND

Shows the result that fulfills both conditions.

**Syntax:** <field> <AND> <value>

### Example

```
resourcegroupname = BCP1 and accountname = securonix
```

## OR

Shows the result which fulfills either one of the specified conditions.

**Syntax:** <field> <OR> <value>

### Example

```
resourcegroupname = BCP1 OR accountname = TG2277
```

### BEFORE

Filters results for events before the specified date. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <field> BEFORE <value>

### Example

```
polycynname = test123 and createdate BEFORE 03/10/2016 06:21:31
```

### AFTER

Filters results for events after the specified date. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <field> AFTER <value>

### Example

```
polycynname = test123 and createdate AFTER 03/10/2016 06:21:31
```

### BETWEEN

Filters results for events between value1 and value2. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <field> BETWEEN <value1><,><value2>

### Example

```
polycynname = test123 and week BETWEEN 4,30
```

### NOT BETWEEN

Filters results for events, not between value1 and value2. The format supported for the date attributes to query is **MM/dd/yyyy HH:mm:ss.SSS**.

**Syntax:** <field> NOT BETWEEN <value1><,><value2>

#### Example

```
bytesout NOT BETWEEN 250,251
```

#### STARTS WITH

Returns results if the string field value starts with the specified value.

**Syntax:** <field> STARTS WITH <value>

#### Example

```
resourcegroupname = BCP1 and accountname STARTS WITH secur
```

#### NOT STARTS WITH

Returns results if the string field value does not start with the specified value.

**Syntax:** <field> NOT STARTS WITH <value>

#### Example

```
resourcegroupname = BCP1 and accountname NOT STARTS WITH secur
```

#### NULL

Returns the events if the field value is empty.

**Syntax:** <field> NULL

#### Example

```
accountname = securonix AND eventcountry NULL
```

#### NOT NULL

Returns the events if the field value is not empty.

**Syntax:** <field> NOT NULL

**Example**

```
accountname = securonix AND eventcountry NOT NULL
```

**IN**

Returns results if the string field value is present in the specified list of comma-separated values.

**Syntax:** <field> IN <value>

**Example**

```
resourcegroupname = BCP1 and accountname in TG2277,TG2207
```

**NOT IN**

Returns results if the string field value is present in the specified list of comma-separated values.

**Syntax:** <field> NOT IN <value>

**Example**

```
resourcegroupname = BCP1 and accountname not in TG2277,TG2207
```

**ENDS WITH**

Returns results if the string field value ends with the specified value.

**Syntax:** <field> ENDS WITH <value>

**Example**

```
resourcegroupname = BCP1 and accountname ENDS WITH curonix
```

**NOT ENDS WITH**

Returns results if the string field value does not end with the specified value.

**Syntax:** <field> NOT ENDS WITH <value>

**Example**

```
resourcegroupname = BCP1 and accountname NOT ENDS WITH curnix
```

**= (Equals)**

.Returns results if the file value matches the specified value exactly

**Syntax:** <field> <=> <value>

**Example**

```
resourcegroupname = BCP1
```

**!= (Not Equals)**

Returns results if the field value does not match the specified value exactly

**Syntax:** <field> <!=> <value>

**Example**

```
resourcegroupname != BCP1
```

**> (Greater Than)**

Returns results if a numerical field is greater than the specified value.

**Syntax:** <field> > <value>

**Example**

```
resourcegroupname = BCP1 and bytesOut > 200
```

**< (Less Than)**

Returns results if a numerical field is less than the specified value.

**Syntax:** <field> < <value>

### Example

```
resourcegroupname = BCP1 and bytesOut < 200
```

#### <= (Less than or Equal to)

Returns results if a numerical field is less than or equal to the specified value.

**Syntax:** <field> <= <value>

### Example

```
resourcegroupname = BCP1 AND year <= 2017
```

#### >= (Greater than or Equal to)

Return results if a numerical field is greater than or equal to the specified value.

**Syntax:** <field> >= <value>

### Example

```
resourcegroupname = BCP1 AND year >= 2017
```

## Time Modifiers

These operators are used to specify Relative Time or Snap Time using date time fields.

#### **+/-Time\_Offset\_IntegerTime\_Unit\_String**

Relative Time. Specifies a specific amount of time to be added or subtracted from current time.

- Can be used with any time field
- Syntax for Relative Time and Snap Time can be combined.
- When no time offset integer is specified 1 is used by default
- Comparators BETWEEN & NOT BETWEEN require proper quotation
- If used with eventtime (index=activity) or generationtime (index=violation), the time range selector on the right will be ignored.

- Only available in SNYP 6.2 CU4 SP2 or Newer.
- Not applicable to index=archive

**Syntax:** <field> <comparator> +/-Time\_Offset\_IntegerTime\_Unit\_String

### Example

```
eventtime after -2d (All events that occurred after Today -2 days or within the last 2 days)
```

### "@Time\_Unit\_String

Snap Time. Specifies the beginning of the time unit selected. For example, "@w is start of the week.

- Can be used with any time field
- Syntax for Relative Time and Snap Time can be combined.
- When no time offset integer is specified 1 is used by default
- Comparators BETWEEN & NOT BETWEEN require proper quotation
- If used with eventtime (index=activity) or generationtime (index=violation), the time range selector on the right will be ignored.
- Only available in SNYP 6.2 CU4 SP2 or Newer.
- Not applicable to index=archive

**Syntax:** <field> <comparator> @Time\_Unit\_String

### Example

```
eventtime between "am", "ad" (All events from the start of the month to the start of today)
```

### +/-Time\_Offset\_IntegerTime\_Unit\_String@Time\_Unit\_String

Combines relative time and snap time.

**Syntax:** <field> <comparator> +/-Time\_Offset\_IntegerTime\_Unit\_String@Time\_Unit\_String



### Example

eventtime between "-qtr@qtr+mon","-mon@mon" (All events from 3 months prior to the beginning of the current quarter plus 1 month to 1 month prior to current month)

### Supported Time Units

| Time Unit | Supported Abbreviations  |
|-----------|--|
| Second    | s, sec, secs, second, seconds  |
| Minute    | m, min, minute, minutes  |
| Hour      | h, hr, hrs, hour, hours  |
| Day       | d, day, days   |
| Week      | w, week, weeks<br>w0 (week0 = Sunday<br>w1 = Monday<br>w2 = Tuesday<br>w3 = Wednesday<br>w4 = Thursday<br>w5 = Friday<br>w6 = Saturday<br>w7 = Sunday (same as w0) |
| Month     | mon, month, months   |
| Quarter   | q, qtr, qtrs, quarter, quarters  |
| Year      | y, yr, yrs, year, years  |

## Filter Command

This command is used to run a query on multiple collections such as: activity, violation, watchlist, riskscore , riskscorehistory, users, lookup, geolocation, etc.

### **FILTER**

Performs an inner join on two indexes. This means that the results display the specified value contained in both indexes based on the comparator.

- Negative comparators are not valid, as this only performs an inner Join.
- Start with the larger index and filter to the smaller index.
- Any search terms before | (pipe) are applied to the first index only. The search terms that follow the | pipe are then used to further narrow and enrich your returned events within the second index.

**Syntax:** FILTER index = <indexname> AND <field> <comparator> <value>

#### **Example**

```
index = violation and accountname = john.doe | FILTER index =  
riskscore and violator = violator
```

### **Streaming Operators**

Streaming Operators execute actions on search results.

## Eval Commands

These streaming operators populate a new field based on an evaluation performed against a field value pair on previously entered search terms.

Capital letters, dashes (-), and spaces in the generated field name breaks the query when piped into other operators.

### **DEC**

Returns the decimal value for the specified

**Syntax:** EVAL (store-field) = (DEC) ( field )

**Example**

```
resourcegroupname = BCP1 | EVAL x = DEC ( bytesin );
resourcegroupname = Email_sent_to_Users | EVAL x = DEC
( bytesin ) | EVAL y = HEX(x)
```

**EQUALS**

Returns true if the value matches. Returns false if the value does not match.

**Syntax:** EVAL <store-field> = <EQUALS> <field> <field-value>

**Example**

```
resourcegroupname = BCP1 | EVAL x = EQUALS ( accountname ,
2029); LEN: EVAL x = LEN ( accountname ) | EVAL y = EQUALS
( x , 6); UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL
y = EQUALS ( x , TG2277); LOWERCASE : EVAL x = LOWERCASE
( accountname ) | EVAL y = EQUALS ( x , tg2277); REPLACE: EVAL
x = REPLACE ( accountname ,TG2277 , securonix) | EVAL y =
EQUALS ( x , securonix) ; SUBSTR: EVAL x = SUBSTR
( accountname , 0 , 2) | EVAL y = EQUALS ( x , TG); ISBOOLEAN:
EVAL x = ISBOOLEAN ( bytesout ) | EVAL y = EQUALS ( x , false);
ISNOTNULL: EVAL x = ISNOTNULL ( resourcegroupid ) | EVAL y =
EQUALS ( x , true); ISNULL: EVAL x = ISNULL ( accountname ) |
EVAL y = EQUALS ( x , false); ISSTRING : EVAL x = ISSTRING
( accountname ) | EVAL y = EQUALS ( x , true); ISNUM : EVAL x =
ISNUM ( accountname ) | EVAL y = EQUALS ( x , true); ISINT:
EVAL x = ISINT ( id ) | EVAL y = EQUALS ( x , true); SDIGIT:
EVAL x = ISDIGIT ( id ) | EVAL y = EQUALS ( x , true)
```

**ISDIGIT**

Returns true if the value is a digit. Returns false if the value is not a digit.

**Syntax:** EVAL <store-field> = <ISDIGIT> < field >

**Example**

```
resourcegroupname = BCP1 | EVAL x = ISDIGIT ( accountname);
LEN: EVAL x = LEN ( resourcegroupid ) | EVAL y = ISDIGIT ( x );
REPLACE: EVAL x = REPLACE ( accountname , - , 1) | EVAL y =
```

```
ISDIGIT ( x ); SUBSTR: EVAL x = SUBSTR ( accountname , 0 , 1) |
EVAL y = ISDIGIT ( x )
```

## MATCH

Populates the new field with true if its value matches the regular expression pattern and false if it does not.

**Syntax:** VAL (store-field) = MATCH (Field, REGEX)

### Example

```
EVAL x = MATCH (accountname, "JH*") • accountname = JH321
Result: x = true • accountname = SH321 Result: x = false
```

## ADD

Returns the sum of two or more fields/numbers.

**Syntax:** EVAL <store-field> = ADD (Field1, Field2, #) EVAL <store-field> = Field1 + Field2 + #

### Example

```
EVAL bandwidth = bytesout + bytesin • bytesout = 10 bytesin =
10 Result: bandwidth = 20
```

## SUBTRACT (DIFFERENCE)

Returns the difference of two or more fields/numbers.

**Syntax:** EVAL <store-field> = DIFFERENCE (Field1, Field2, #) EVAL <store-field> = Field1 - Field2 - #

### Example

```
EVAL freecache = customnumber1 - customnumber2 • customnumber1
= 200 customnumber2 = 80 Result:: freecache = 120
```

## MULTIPLY

Returns the product of two or more fields/numbers.

**Syntax:** EVAL <store-field> = MULTIPLY (Field1, Field2, #) EVAL <store-field> = Field = Field1 \* Field2 \* #

### Example

```
EVAL est_mon_bandwidth = (bytesout + bytesin) * 30 * bytesout = 10 bytesin = 10 Result: est_mon_bandwidth = 600
```

## DIVIDE

Returns the quotient of two or more fields/numbers.

**Syntax:** EVAL <store-field> = DIVIDE (Field1, Field2, #) EVAL <store-field> = Field1 / Field2 / #

### Example

```
EVAL bw_kb = bytesout + bytesin / 1000 * bytesout = 10 bytesin = 10 Result: est_mon_bandwidth = 0.02
```

## FROM\_UNIXTIME

Returns date string from an epoch time.

- Only converts 10-digit epoch (unix) time at this time.
- Combine with TO\_UNIXTIME to convert one human-readable form to another format.

The following date formats are supported:

- yyyy-MM-dd'T'HH:mm:ss'Z'
- yyyy-MM-dd'T'HH:mm:ssZ
- yyyy-MM-dd'T'HH:mm:ss
- yyyy-MM-dd'T'HH:mm:ss.SSS'Z'
- yyyy-MM-dd'T'HH:mm:ss.SSSZ

- yyyy-MM-dd HH:mm:ss
- yyyyMMdd
- MM/dd/yyyy
- MM/dd/yyyy HH:mm:ss
- MM/dd/yyyy'T'HH:mm:ss.SSS'Z'
- MM/dd/yyyy'T'HH:mm:ss.SSSZ
- MM/dd/yyyy'T'HH:mm:ss.SSS
- MM/dd/yyyy'T'HH:mm:ssZ
- MM/dd/yyyy'T'HH:mm:ss

The following formats do not show timezone: yyyy-MM-dd'T'HH:mm:ss'Z' yyyy-MM-dd'T'HH:mm:ss.SSS'Z' MM/dd/yyyy'T'HH:mm:ss.SSS'Z'

**Syntax:** EVAL <store-field> = <from\_unixtime> < field > < date format >

#### Example

```
EVAL x = from_unixtime (eventtime , MM/dd/yyyy HH:mm:ss)
```

#### to\_unixtime

Returns epoch time from a valid date string.

Only converts to 10-digit epoch (unix) time.

**Syntax:** EVAL <store-field> = <to\_unixtime> < field | Valid String >

#### Example

```
EVAL x = to_unixtime (04/27/2017 15:03:49); EVAL x =  
to_unixtime (dt_firstseen)
```

#### BASE64

Returns the base64 encoding value.

**Syntax:** EVAL (store-field) = (BASE64) ( field )

**Example**

```
requesturl = www.google.com Result: x = d3d3Lmdvb2dsZS5jb20=
```

**UNBASE64**

Returns the base64 decoding value.

**Syntax:** EVAL (store-field) = (UNBASE64) ( field )

**Example**

```
requesturl = d3d3Lmdvb2dsZS5jb20= Result: x = www.google.com
```

**CONCAT**

Populates new field with results by concatenating (joining) the values specified. Limited to 3 values.

**Syntax:** EVAL (store-field) = CONCAT (Field/"string", field/"string", field/"string")

**Example**

```
EVAL x = CONCAT (firstname, ".", lastname) firstname = John ,  
lastname = Doe Result: x = John.Doe
```

**ISINT**

Returns true if the value is an integer. Returns false if the value is not an integer.

**Syntax:** EVAL <store-field> = <ISINT> < field >

**Example**

```
resourcegroupname = BCP1 | EVAL x = ISINT ( accountname ); LEN:  
EVAL x = LEN ( accountname ) | EVAL y = ISINT ( x ); UPPERCASE:  
EVAL x = UPPERCASE ( accountname ) | EVAL y = ISINT ( x );  
LOWERCASE: EVAL x = LOWERCASE ( accountname ) | EVAL y = ISINT  
( x ); REPLACE: EVAL x = REPLACE ( accountname ,TG2277 ,
```

```
securonix) | EVAL y = ISINT ( x ); SUBSTR: EVAL x = SUBSTR
( accountname , 0 , 2) | EVAL y = ISINT ( x )
```

## ISNOTNULL

Returns true if the value is not null. Returns false if the value is null.

**Syntax:** EVAL <store-field> = <ISNOTNULL> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISNOTNULL ( accountname ) ;
LEN: EVAL x = LEN ( accountname ) | EVAL y = ISNOTNULL ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
ISNOTNULL ( x ); LOWERCASE: EVAL x = LOWERCASE ( accountname )
| EVAL y = ISNOTNULL ( x ); EQUALS: EVAL x = EQUALS
( accountname , - ) | EVAL y = ISNOTNULL ( x ); REPLACE: EVAL x
= REPLACE ( accountname , - , securonix) | EVAL y = ISNOTNULL
( x ); SUBSTR: EVAL x = SUBSTR ( accountname , 0 , 5) | EVAL y
= ISNOTNULL ( x ); ISBOOLEAN: EVAL x = ISBOOLEAN ( bytesout ) |
EVAL y = ISNOTNULL ( x ); ISSTRING: EVAL x = ISSTRING
( accountname ) | EVAL y = ISNOTNULL ( x ); ISNUM: EVAL x =
ISNUM ( accountname ) | EVAL y = ISNOTNULL ( x ); ISEMPY: EVAL
x = ISEMPY ( accountname ) | EVAL y = ISNOTNULL ( x )
```

## ISBOOLEAN

Returns true or false if the field is Boolean.

**Syntax:** EVAL <store-field> = <ISBOOLEAN> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISBOOLEAN ( accountname );
LEN: EVAL x = LEN ( accountname ) | EVAL y = ISBOOLEAN ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
ISBOOLEAN ( x ); LOWERCASE: EVAL x = LOWERCASE ( accountname )
| EVAL y = ISBOOLEAN ( x ); REPLACE: EVAL x = REPLACE
( accountname ,TG2277 , securonix) | EVAL y = ISBOOLEAN ( x );
SUBSTR: EVAL x = SUBSTR ( accountname , 0 , 2) | EVAL y =
ISBOOLEAN ( x ); ISNOTNULL: EVAL x = ISNOTNULL
( resourcegroupid ) | EVAL y = ISBOOLEAN ( x ); ISNULL: EVAL x
```



```
= ISNULL ( accountname ) | EVAL y = ISBOOLEAN ( x ); ISSTRING:
EVAL x = ISSTRING ( accountname ) | EVAL y = ISBOOLEAN ( x );
EQUALS: EVAL x = EQUALS ( accountname , securonix ) | EVAL y =
ISBOOLEAN ( x )
```

## LEN

Returns the length of the field value.

**Syntax:** EVAL <store-field> = <LEN> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = LEN ( accountname );
LOWERCASE: EVAL y = LOWERCASE ( accountname ) | EVAL x = LEN
( y ); UPPERCASE: EVAL y = UPPERCASE ( accountname ) | EVAL x =
LEN ( y ); ISEMPY: EVAL x = LEN ( accountname ) | EVAL y =
ISEMPY ( accountname ); REPLACE: EVAL y = REPLACE
( accountname , - , securonix ) | EVAL x = LEN ( y ); SUBSTR:
EVAL z = REPLACE ( accountname , - , securonix ) | EVAL y =
SUBSTR ( z , 0 , 5 ) | EVAL x = LEN ( y ); ISBOOLEAN: EVAL x =
LEN ( resourcegroupid ) | EVAL y = ISBOOLEAN ( x ); ISINT: EVAL
x = LEN ( resourcegroupid ) | EVAL y = ISINT ( x ); ISNOTNULL:
EVAL x = LEN ( resourcegroupid ) | EVAL y = ISNOTNULL ( x );
ISNULL: EVAL x = LEN ( resourcegroupid ) | EVAL x = ISNULL
( x ); ISDIGIT: EVAL x = LEN ( resourcegroupid ) | EVAL y =
ISDIGIT ( x ); EQUALS: EVAL x = LEN ( accountname ) | EVAL y =
EQUALS ( x , 5 )
```

## ISNUM

Returns true if the value is a number. Returns false if the value is not a number.

**Syntax:** EVAL <store-field> = <ISNUM> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISNUM ( accountname ); LEN:
EVAL x = LEN ( accountname ) | EVAL y = ISNUM ( x ); UPPERCASE:
EVAL x = UPPERCASE ( accountname ) | EVAL y = ISNUM ( x );
LOWERCASE: EVAL x = LOWERCASE ( accountname ) | EVAL y = ISNUM
( x ); EQUALS: VAL x = EQUALS ( accountname , - ) | EVAL y =
```

```
ISNUM ( x ); REPLACE: EVAL x = REPLACE ( accountname , - ,
securonix) | EVAL y = ISNUM ( x ); SUBSTR: EVAL x = SUBSTR
( accountname , 0 , 5) | EVAL y = ISNUM ( x )
```

## UPPERCASE

Converts all characters to uppercase.

**Syntax:** EVAL <store-field> = <UPPERCASE> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = UPPERCASE ( accountname );
LEN: EVAL x = UPPERCASE ( accountname ) | EVAL y = LEN ( x );
LOWERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
LOWERCASE ( x ); ISEMPTY: EVAL x = UPPERCASE ( accountname ) |
EVAL y = ISEMPTY ( x ); EQUALS: EVAL y = UPPERCASE
( accountname ) | EVAL x = EQUALS ( y , - ); REPLACE : EVAL x =
UPPERCASE ( accountname ) | EVAL y = REPLACE ( x , - ,
securonix); SUBSTR: EVAL y = SUBSTR ( accountname , 0 , 5) |
EVAL x = UPPERCASE ( y ); ISBOOLEAN: EVAL x = UPPERCASE
( resourcegroupid ) | EVAL y = LEN ( x ) | EVAL x = ISBOOLEAN
( y ); ISNOTNULL : EVAL x = UPPERCASE ( resourcegroupid ) |
EVAL y = ISNOTNULL ( x ); ISNULL : EVAL x = UPPERCASE
( accountname ) | EVAL y = ISNULL ( x ); ISSTRING : EVAL x =
UPPERCASE ( accountname ) | EVAL y = ISSTRING ( x )
```

## ISSTRING

Returns true if the value is a string. Returns false if the value is not a string.

**Syntax:** EVAL <store-field> = <ISSTRING> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISSTRING ( accountname );
LEN: EVAL x = LEN ( accountname ) | EVAL y = ISSTRING ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
ISSTRING ( x ); LOWERCASE: EVAL x = LOWERCASE ( accountname ) |
EVAL y = ISSTRING ( x ); REPLACE: EVAL x = REPLACE
( accountname , - , securonix) | EVAL y = ISSTRING ( x );
```

```
SUBSTR: EVAL x = SUBSTR ( accountname , 0 , 5) | EVAL y =
ISSTRING ( x )
```

## ISNULL

Returns true if the value is null. Returns false if the value is not null.

**Syntax:** EVAL <store-field> = <ISNULL> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISNULL ( accountname );
LEN: EVAL x = LEN ( accountname ) | EVAL y = ISNULL ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y = ISNULL
( x ); LOWERCASE: EVAL x = LOWERCASE ( accountname ) | EVAL y =
ISNULL ( x ); EQUALS: EVAL x = EQUALS ( accountname , - ) |
EVAL y = ISNULL ( x ); REPLACE: EVAL x = REPLACE
( accountname , - , securonix) | EVAL y = ISNULL ( x ); SUBSTR:
EVAL x = SUBSTR ( accountname , 0 , 5) | EVAL y = ISNULL ( x );
ISBOOLEAN: EVAL x = ISBOOLEAN ( bytesout ) | EVAL y = ISNULL
( x ); ISSTRING: EVAL x = ISSTRING ( accountname ) | EVAL y =
ISNULL ( x ); ISNUM: EVAL x = ISNUM ( accountname ) | EVAL y =
ISNULL ( x ); ISEMPY: EVAL x = ISEMPY ( accountname ) | EVAL
y = ISNULL ( x )
```

## ISEMPY

Returns true if the value is empty. Returns false if the value is not empty.

**Syntax:** EVAL <store-field> = <ISEMPY> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = ISEMPY ( accountname );
LEN: EVAL x = LEN ( accountname ) | EVAL y = ISEMPY ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
ISEMPY ( x ); LOWERCASE: EVAL x = LOWERCASE ( accountname ) |
EVAL y = ISEMPY ( x ); EQUALS: EVAL x = EQUALS ( accountname ,
- ) | EVAL y = ISEMPY ( x ); REPLACE: EVAL x = REPLACE
( accountname , - , securonix) | EVAL y = ISEMPY ( x ); SUBSTR:
EVAL x = SUBSTR ( accountname , 0 , 5) | EVAL y = ISEMPY
( x ); ISBOOLEAN: EVAL x = ISBOOLEAN ( bytesout ) | EVAL y =
```

```
ISEMPTY ( x ); ISNOTNULL: EVAL x = ISNOTNULL
( resourcegroupid ) | EVAL y = ISEMPTY ( x ); ISNULL: EVAL x =
ISNULL ( accountname ) | EVAL y = ISEMPTY ( x ); ISSTRING: EVAL
x = ISSTRING ( accountname ) | EVAL y = ISEMPTY ( x ); ISNUM:
EVAL x = ISNUM ( accountname ) | EVAL y = ISEMPTY ( x )
```

## HEX

Returns the hexadecimal value.

**Syntax:** EVAL (store-field) = (HEX) ( field )

### Example

```
resourcegroupname = BCP1 | EVAL x = HEX ( bytesin ); Example 1:
resourcegroupname = Email_sent_to_Users | EVAL x = DEC
( bytesin ) | EVAL y = HEX(x)
```

## LOWERCASE

Converts all characters to lowercase.

**Syntax:** EVAL <store-field> = <LOWERCASE> < field >

### Example

```
resourcegroupname = BCP1 | EVAL x = LOWERCASE ( accountname );
LEN: EVAL x = LOWERCASE ( accountname ) | EVAL y = LEN ( x );
UPPERCASE: EVAL x = UPPERCASE ( accountname ) | EVAL y =
LOWERCASE ( x ); ISEMPTY: EVAL x = LOWERCASE ( accountname ) |
EVAL y = ISEMPTY ( x ); EQUALS: EVAL y = LOWERCASE
( accountname ) | EVAL x = EQUALS ( y , - ); REPLACE: EVAL x =
LOWERCASE ( accountname ) | EVAL y = REPLACE ( x , - ,
securonix); SUBSTR: EVAL y = SUBSTR ( accountname , 0 , 5) |
EVAL x = LOWERCASE ( y ); ISBOOLEAN: EVAL x = LOWERCASE
( resourcegroupid ) | EVAL y = LEN ( x ) | EVAL x = ISBOOLEAN
( y ); ISNOTNULL: EVAL x = LOWERCASE ( resourcegroupid ) | EVAL
y = ISNOTNULL ( x ); ISNULL: EVAL x = LOWERCASE ( accountname )
| EVAL y = ISNULL ( x ); ISSTRING: EVAL x = LOWERCASE
( accountname ) | EVAL y = ISSTRING ( x )
```

## REPLACE

Returns a string after replacing all occurrences.

**Syntax:** EVAL <store-field> = <REPLACE> < field > < fieldvalue > <replace-value>

### Example

```
resourcegroupname = BCP1 | EVAL x = REPLACE
( accountname ,TG2277 , securonix); LEN: EVAL x = REPLACE
( accountname ,TG2277 , securonix) | EVAL y = LEN ( x );
UPPERCASE: EVAL x = REPLACE ( accountname ,TG2277 , securonix)
| EVAL y = UPPERCASE ( x ); LOWERCASE: EVAL x = REPLACE
( accountname ,TG2277 , SECURONIX) | EVAL y = LOWERCASE ( x );
EQUALS: EVAL x = REPLACE ( accountname ,TG2277 , securonix) |
EVAL y = EQUALS ( x , securonix); SUBSTR: EVAL x = REPLACE
( accountname ,TG2277 , securonix) | EVAL y = SUBSTR ( x , 0 ,
2); ISBOOLEAN: EVAL x = REPLACE ( accountname ,TG2277 ,
securonix) | EVAL y = ISBOOLEAN ( x ); ISNOTNULL: EVAL x =
REPLACE ( accountname ,TG2277 , securonix) | EVAL y = ISNOTNULL
( x ); ISNULL: EVAL x = REPLACE ( accountname ,TG2277 ,
securonix) | EVAL y = ISNULL ( x ); ISSTRING: EVAL x = REPLACE
( accountname ,TG2277 , securonix) | EVAL y = ISSTRING ( x );
ISNUM: EVAL x = REPLACE ( accountname ,TG2277 , 123 ) | EVAL y
= ISNUM ( x ); ISINT: EVAL x = REPLACE ( accountname ,TG2277 ,
123 ) | EVAL y = ISINT ( x ); ISDIGIT: EVAL x = REPLACE
( accountname ,TG2277 , 7 ) | EVAL y = ISDIGIT ( x )
```

## SUBSTR

Returns a substring of the actual field value.

**Syntax:** EVAL <store-field> = <SUBSTR> < field > < start-position > <endposition>

### Example

```
EVAL x = SUBSTR ( accountname , 0 , 5 ); REPLACE: EVAL x =
REPLACE ( accountname ,TG2277 , securonix) | EVAL y = SUBSTR
( x , 0 , 3 ); LEN: EVAL x = SUBSTR ( accountname , 0 , 3 ) |
EVAL y = LEN ( x ); UPPERCASE: EVAL x = SUBSTR ( accountname ,
0 , 3 ) | EVAL y = UPPERCASE ( x ); LOWERCASE: EVAL x = SUBSTR
( accountname , 0 , 3 ) | EVAL y = LOWERCASE ( x ); EQUALS:
EVAL x = SUBSTR ( accountname , 0 , 3 ) | EVAL y = EQUALS ( x ,
```

```
TG2); ISBOOLEAN: EVAL x = SUBSTR ( accountname , 0 , 3 ) | EVAL
y = EQUALS ( x , TG2) | EVAL z = ISBOOLEAN ( y ); ISNOTNULL:
EVAL x = SUBSTR ( accountname , 0 , 3 ) | EVAL y = ISNOTNULL
( x ); ISNULL: EVAL x = SUBSTR ( accountname , 0 , 3 ) | EVAL y
= ISNULL ( x ); ISSTRING: EVAL x = SUBSTR ( accountname , 0 ,
3 ) | EVAL y = ISSTRING ( x ); ISNUM: EVAL x = SUBSTR
( accountname , 0 , 3 ) | EVAL y = ISNUM ( x ); ISINT: EVAL x =
SUBSTR ( accountname , 0 , 3 ) | EVAL y = ISINT ( x ); ISDIGIT:
EVAL x = SUBSTR ( accountname , 0 , 3 ) | EVAL y = ISDIGIT
( x );
```

## SUBSTRBYINDEX

Returns sub-string of actual field value by index.

**Syntax:** EVAL <store-field> = <SUBSTRBYINDEX> < field > < delimiter > <An integer indicating the number of occurrences of delimiter>

### Example

```
EVAL x = SUBSTRBYINDEX (workemail , "a", 1 ); REPLACE: EVAL x =
REPLACE ( workemail ,TG2277 , securonix) | EVAL y =
SUBSTRBYINDEX (x , "a", 1 ); LEN: EVAL x = SUBSTRBYINDEX
(workemail , "a", 1 ) | EVAL y = LEN ( x ); UPPERCASE: EVAL x =
SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y = UPPERCASE ( x );
LOWERCASE: EVAL x = SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL
y = LOWERCASE ( x ); EQUALS: EVAL x = SUBSTRBYINDEX
(workemail , "a", 1 ) | EVAL y = EQUALS ( x , TG2); ISBOOLEAN:
EVAL x = SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y = EQUALS
( x , TG2) | EVAL z = ISBOOLEAN ( y ); ISNOTNULL: EVAL x =
SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y = ISNOTNULL ( x );
ISNULL: EVAL x = SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y =
ISNULL ( x ); ISSTRING: EVAL x = SUBSTRBYINDEX (workemail ,
"a", 1 ) | EVAL y = ISSTRING ( x ); ISNUM: EVAL x =
SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y = ISNUM ( x );
ISINT: EVAL x = SUBSTRBYINDEX (workemail , "a", 1 ) | EVAL y =
ISINT ( x ); ISDIGIT: EVAL x = SUBSTRBYINDEX (workemail , "a",
1 ) | EVAL y = ISDIGIT ( x );
```

## VISUALCOMPARATOR

Provides the the visual comparator value.

**Syntax:** EVAL (store-field) = VISUALCOMPARATOR( field , field value , (> | < | <= | >= | = ) (threshold value))

EVAL (store-field) = VISUALCOMPARATOR( field , field value , (> | < | <= | >= | = ) (threshold value)); EVAL (store-field) = VISUALCOMPARATOR( field , field value , (> | < | <= | >= | = ) (threshold value)); EVAL (store-field) = VISUALCOMPARATOR( field , field value , (> | < | <= | >= | = ) (threshold value))

Relation Operators to Use: > , < , <= , >= , =

## Nested Queries

You can nest queries that use EVAL commands.

**Syntax:** (resourcegroupname | policyname) | (EVAL) (=) (EVAL COMMANDS (EVAL COMMANDS)( (fiel1) .. )field 2)))

### Example

```
EVAL x = len ( uppercase ( lowercase ( substr
( concat(u_firstname,u_lastname), 0,4) ) ) ); EVAL x =
deviceaction | eval y = equals ( firstname , Kiran); isnum
( len ( uppercase ( lowercase ( substr
( concat(u_firstname,u_lastname), 0,4) ) ) ) ); replace (substr
( concat(u_firstname,u_lastname), 0,10),Secure, Securonix);
EVAL x = len ( substr ( concat(u_firstname,u_lastname), 0,4) )
```

## Field Commands

These operators perform specific functions on a field.

### RENAME

Rename the source field in search results.

**Syntax:** RENAME < field1> <as> <field2>

### Example

```
Resourcegroupname = BCP1 | RENAME ipaddress as hostaddress
```

## FIELDS

Display or remove the specified field(s) from the Results. Use "+" to display only the specified fields. Use "-" to remove the specified fields from the results.

**Syntax:** FIELDS < + or - > <field1><,><field2><,>...<field N>

### Example

```
resourcegroupname = BCP1 | FIELDS + ipaddress; +: FIELDS +  
ipaddress , accountname; -: FIELDS - ipaddress , accountname
```

## DELETE

**Delete specific events from the results displayed on the UI. Note: Does not delete events from the system**

**Syntax:** DELETE <field1 = value> ...<field N = value>

### Example

```
Resourcegroupname = BCP1 | DELETE ipaddress = 182.74.60.19 ...  
| DELETE ipaddress = 182.74.60.19 accountname = TG2277
```

## GEOLOOKUP

Extract location information based on IP address, such as city, country, latitude, and longitude.

**Syntax:** GEOLOOKUP <field>

### Example

```
Resourcegroupname = BCP1 | GEOLOOKUP ipaddress
```

## REX

Extracts and creates fields based on regular expression groupings matched in the specified key field.

- Field name is the same as Regex group name



- Performs extraction only, not filtering
- Multiple groupings (fields) can be formed from a single use as long as all the data is in the specified field

**Syntax:** REX key=field "Regex\_grouping"

### Example

```
REX key=destinationprocessname "(?<dstproc>[^\/*<>|\r\n]+)$" •
Extracts the process name and extension to new field dstproc
from the specified file path contained in
destinationprocessname attribute • destinationprocessname = c:\
\program files (x86)\google\chrome\application\chrome.exe
Results: dstproc = chrome.exe
```

## Transforming Operators

These operators display search results in various visualization formats.

## Reporting Commands

These operators transform search results into visual reports.

### GEOMAP

Displays the events as points on a GEOMAP.

- Field1 sets latitude
- Field2 sets the longitude
- Field3 values are used in the map legend

**Syntax:** GEOMAP <field1> <field2> <field-n>

### Example

```
GEOMAP latitude longitude addr; Activity : resourcegroupname =
BCP1 | GEOMAP latitude longitude ipaddress; Violation :
policyname = Logon_Failure | GEOMAP eventlatitude
eventlongitude ipaddress; Index : index = tpi | GEOMAP
```

```
tpi_latitude tpi_longitude tpi_addr; GEOLOOKUP:
resourcegroupname = BCP1 | GEOLOOKUP ipaddress | GEOMAP
latitude longitude ipaddress; Group By: resourcegroupname =
BCP1 | GEOMAP eventlatitude eventlongitude ipaddress by
eventregion
```

## BUBBLECHART

Shows a type of chart that displays three dimensions of data (x, y, z).

- Chart does a count on field1 by default unless an aggregated operator is applied to the field
- Count of field1 determines the size of the bubble
- Field 2 is the values observed on the Y-axis
- Field3 is the values observed on the X-axis

**Syntax:** BUBBLECHART <field1> <count> <by> <field2> .... <field N>

### Example

```
resourcegroupname = BCP1 | BUBBLECHART ipaddress; STACKED:
BUBBLECHART ipaddress by accountid; COUNT: BUBBLECHART count by
ipaddress; STACKED with COUNT: BUBBLECHART ipaddress by
accountid
```

## BARCHART

Represents grouped data with rectangular bars with lengths proportionate to the values they represent.

- Chart does a count on field1 by default unless an aggregated operator is applied to the field

**Syntax:** BARCHART <field1> <count> <by> <field2> .... <field N>

### Example

```
resourcegroupname = BCP1 | BARCHART ipaddress; STACKED:
BARCHART ipaddress by accountname; GROUP: BARCHART ipaddress
accountname; COUNT: BARCHART count by ipaddress; STACKED with
COUNT: BARCHART count by ipaddress accountname
```

## TIMECHART

Displays the data for field(s) in a time series based on a specified increment

- Maximum of 2 fields
- Available Periods:
  - Hours, Hourly
  - Days, Daily
  - Weeks, Weekly
  - Months, Monthly
- The unique count of Field1, as observed for field2 value is the y-value plotted
- Each unique value of field2 is a separate line on the chart

**Syntax:** TIMECHART <# increment> <period> <count by> <field1> <by> <field2> ... <field N>

### Example

```
TIMECHART weekly policyuniqkey by policynome; TIMECHART weekly
policyuniqkey policynome
```

## SPAN

Displays the data in a table format by grouping the specified fields around a duration period or by calculating the average event count of a time period.

**Duration Periods:** dur =

- Seconds - s, sec, second, seconds.
- Minutes - m, min, minute, minutes
- Hours - h, hr, hour, hours
- Days - d, day, days
- Month - mon, month, months

**Average Periods** avg=

- seconds
- minutes
- hourly
- daily
- weekly
- monthly
- yearly

**Syntax:** SPAN dur=period Field1, Field2, fieldN

Syntax: SPAN avg=period Field1, Field2, fieldN

### Example

```
SPAN dur=5min ipaddress accountname; SPAN dur=weekly  
accountname
```

## RARE

Displays the least common values of a field(s). Use this limit to restrict the number of displayed events.

- The query will show up to ten results if a limit is not specified.
- Each field refers to the faceting of the previous field. Field1 is L1, Field2 is L2, etc.
- Best Practice: This operator generally produces a large number of 1 off events, if you are looking for something a little more common but still rare, append the query with a where clause: | where count > #

**Syntax:** RARE <limit = constant> <field1> <by> <field 2> .... <field N>

### Example

```
resourcegroupname = BCP1 | RARE ipaddress; STACKED: RARE
ipaddress by accountname; GROUP: RARE ipaddress accountname;
LIMIT: RARE limit =5 ipaddress; STACKED with LIMIT: RARE limit
=5 ipaddress accountid
```

## TOP

Displays the most common values with the count of events for each value based on the specified field

- The query will show up to ten results if a limit is not specified.
- Each field refers to the faceting of the previous field. Field1 is L1, Field2 is L2, etc.

**Syntax:** TOP <limit = constant> <field1> <by> <field 2> .... <field N>

### Example

```
resourcegroupname = BCP1 | TOP ipaddress; STACKED: TOP
ipaddress by accountname; GROUP: TOP ipaddress accountname;
LIMIT: TOP limit =5 ipaddress; STACKED with LIMIT: TOP limit =5
ipaddress accountid
```

## STATS

Displays the values as observed with count of events for each value based on the specified attributes.

- Each field refers to faceting of the previous field. Field1 is L1, Field2 is L2, etc.

**Syntax:** STATS < field | count> <by> <field> STATS LIMIT=# Field1 Field2 field3

### Example

```
STATS ipaddress accountname
```

## LINK

Displays a graphical view of the connections of 2 field values based on the middle linking field values

- Field1 and Field3 are the fields to be linked
- Field2 represents the link between the other 2 fields.
- Best Practice: Put the fields with larger unique values on the outside with the smallest cardinality field in the middle.
- Limitation: If a field has a value of NULL then the link will not form as it needs an actual value. To circumvent this where you are trying to find the link for NULL values pipe the field into eval to generate a value and then form the link using that field. EX: emailsubject NULL | EVAL e\_subject = ISNULL(emailsubject) | LINK emailsender e\_subject emailrecipient

**Syntax:** LINK < field1> <field2> <field-n>

### Example

```
LINK emailsender filename emailrecipient; Activity:
resourcegroupname = BCP1 | LINK ipaddress accountname filename;
Violation: policyname = Logon_Failure | LINK ipaddress
accountname filename
```

## TABLE

Displays the data in a table format based on the fields specified in a comma or space separated list.

**Syntax:** TABLE <field1><,><field2><,>...<field N>

### Example

```
resourcegroupname = BCP1 | TABLE ipaddress; Multiple  
Attributes: TABLE ipaddress , accountname, accountstatus
```

## GEOLINK

Plots the geographical connection between two fields on a world map.

- Field1 sets the starting point
- Field3 sets the ending point
- Applicable to the following fields:
  - ipaddress
  - sourceaddress
  - sourcehostname
  - destinationaddress
  - destinationhostname
  - resourcehostname
  - devicehostname

**Syntax:** GEOLINK Field1 Field3

### Example

```
GEOLINK ipaddress destinationaddress
```

## HEATMAP

Forms a matrix based on selected attributes where values are represented by color.

- The chart does a count on field1 by default unless an aggregated operator is applied to the attribute
- The count of field1 determines the color (and size of the block if only one attribute is given)
- Field 2 is the values observed on the Y-axis
- Field3 is the values observed on the X-axis

**Syntax:** HEATMAP Field1 Field2 field3

### Example

```
LINK emailsender filename emailrecipient; Activity:
resourcegroupname = BCP1 | LINK ipaddress accountname filename;
Violation: policyname = Logon_Failure | LINK ipaddress
accountname filename
```

## INDEXEDVOLUME

Displays the number of events indexed based on a duration period in the form of a table or bar chart.

- **Duration periods:**
  - Seconds
  - Minutes
  - Hours
  - Days
  - Months
- Supported Start / End periods Now-#Timemodifier
- **Supported Types:**
  - Table
  - Chart

**Syntax:** INDEXEDVOLUME start=NOW/Time\_Modifier-#Time\_Modifier end=NOW/  
Time\_Modifier-#Time\_Modifier span=#duration type=type



### Example

```
INDEXEDVOLUME start=NOW/days-14days end=NOW span=1hours
type=table; INDEXEDVOLUME start=NOW/days-14days end=NOW
span=1hours type=chart
```

## Aggregation Operators

These operators function on the output of transforming functions.

| Command   |   | Syntax       |
|---|---|--------------|
| <b>MIN</b><br><b>Examples:</b> STATS<br>MIN(bytesout) by<br>ipaddress account-<br>name; resource-<br>groupname =<br>Email_sent_to_Users<br>  BUBBLECHART<br>MIN(bytesout) ipad-<br>dress accountname;<br>resourcegroupname =<br>Email_sent_to_Users<br>  TOP MIN(bytesout)<br>ipaddress employ-<br>eeid; resource-<br>groupname =<br>Email_sent_to_Users<br>  RARE MIN(bytes-<br>out) ipaddress em-<br>ployeeid | Provides the MIN value<br>for specified field<br><br><div> <b>Note</b><br/><br/> MIN Operator should<br/> be used with follow-<br/> ing commands: TOP,<br/> RARE, STATS and BUB-<br/> BLECHART </div> | MIN(<field>) |
| <b>MAX</b><br><b>Examples:</b> STATS<br>MAX(bytesout) by<br>ipaddress account-<br>name; resource-<br>groupname =<br>Email_sent_to_Users<br>  BUBBLECHART<br>MAX(bytesout) ipad-   | Provides the MAX value<br>for specified field<br><br><div> <b>Note</b><br/><br/> MAX Operator should<br/> be used with follow-<br/> ing commands: TOP, </div>   | MAX(<field>) |

| Command   |  | Syntax                    |
|---|--|---------------------------|
| <pre>dress accountname; resourcegroupname = Email_sent_to_Users   TOP MAX(bytesout) ipaddress employ- eeid; resource- groupname = Email_sent_to_Users   RARE MAX(bytes- out) ipaddress em- ployeeid</pre>   | <p>RARE, STATS and BUB-<br/>BLECHART</p>   |                           |
| <p><b>SUM</b><br/><b>Examples:</b> STATS<br/>MAX(bytesout) by<br/>ipaddress account-<br/>name; resource-<br/>groupname =<br/>Email_sent_to_Users<br/>  BUBBLECHART<br/>MAX(bytesout) ipad-<br/>dress accountname;<br/>resourcegroupname =<br/>Email_sent_to_Users<br/>  TOP MAX(bytesout)<br/>ipaddress employ-<br/>eeid; resource-<br/>groupname =<br/>Email_sent_to_Users<br/>  RARE MAX(bytes-<br/>out) ipaddress em-<br/>ployeeid</p> | <p>Provides the aggregated<br/>SUM value for specified<br/>field</p> <p><b>Note</b><br/><br/>SUM Operator should<br/>be used with follow-<br/>ing commands: TOP,<br/>RARE, STATS and BUB-<br/>BLECHART</p> | <p>SUM(&lt;field&gt;)</p> |
| <p><b>AVG</b><br/><b>Examples:</b> STATS<br/>AVG(bytesout) by<br/>ipaddress account-<br/>name; resource-<br/>groupname =<br/>Email_sent_to_Users</p>  | <p>Provides the AVG value<br/>for specified field</p> <p><b>Note</b><br/><br/>AVG Operator should<br/>be used with follow-</p>   | <p>AVG(&lt;field&gt;)</p> |

| Command   |   | Syntax                    |
|---|---|---------------------------|
| <pre>  BUBBLECHART AVG(bytesout) ipaddress accountname; resourcegroupname = Email_sent_to_Users   TOP AVG(bytesout) ipaddress employeeid; resource- groupname = Email_sent_to_Users   RARE AVG(bytes- out) ipaddress em- ployeeid</pre>   | <p>ing commands: TOP, RARE, STATS and BUBBLECHART</p>   |                           |
| <p><b>PCR</b><br/> <b>Examples:</b> TOP<br/> PCR( bytesin, bytesout) ipaddress<br/> accountname; re-<br/> sourcegroupname =<br/> Email_sent_to_Users<br/>   BUBBLECHART<br/> PCR( bytesin, bytesout) ipaddress<br/> accountname; re-<br/> sourcegroupname =<br/> Email_sent_to_Users<br/>   BARCHART<br/> PCR( bytesin, bytesout) ipaddress<br/> accountname; re-<br/> sourcegroupname =<br/> Email_sent_to_Users<br/>   TIMECHART weekly<br/> PCR( bytesin, bytesout) ipaddress<br/> accountname</p> | <p>Finds changes in traffic flows that indicate exfiltration.</p> <p><b>Note</b></p> <p><b>Analysis Techniques:</b><br/> Identify changes in host roles, and investigate. PCR is a normalized metric of traffic ratios and from a host ranging from -1 to 1. <b>PCR</b> = <math display="block">\frac{(\text{bytesin} - \text{bytesout})}{(\text{bytesin} + \text{bytesout})}</math> <b>PCR</b><br/> <b>host role:</b> 1.0 pure push - FTP upload, multicast, beaconing<br/> 0.4 70:30 export - Sending Email<br/> 0.0 Balanced Exchange - NTP, ARP probe<br/> -0.5 3:1 import - HTTP Browsing<br/> -1.0 pure</p> | <p>PCR(field1,field2)</p> |

| Command       |  | Syntax               |
|---------------|--|----------------------|
|               | <p data-bbox="621 279 954 583">pull - HTTP Download<br/>DNS is less noisy than HTTP for this metric, and is a possible exfil channel. A positive shift in PCR for DNS traffic may indicate DNS Exfil.</p> <p data-bbox="581 646 930 762"><math>PCR = (bytes_{in} - bytes_{out}) / (bytes_{in} + bytes_{out})</math></p> <p data-bbox="581 798 803 835"><b>PCR host role:</b></p> <ul data-bbox="634 892 967 1428" style="list-style-type: none"> <li>• 1.0 pure push - FTP upload, multicast, beaconing</li> <li>• 0.4 70:30 export - Sending Email</li> <li>• 0.0 Balanced Exchange - NTP, ARP probe</li> <li>• -0.5 3:1 import - HTTP Browsing</li> <li>• -1.0 pure pull - HTTP Download</li> </ul> <p data-bbox="581 1465 967 1696">DNS is less noisy than HTTP for this metric, and is a possible exfil channel. A positive shift in PCR for DNS traffic may indicate DNS Exfil.</p> |                      |
| <b>STDDEV</b> | Calculates the Standard Deviation observed for   | <b>STDDEV(Field)</b> |

| Command   |  | Syntax                       |
|---|--|------------------------------|
| <b>Exam-<br/>ples:</b> <code>STDDEV(bytesin)</code>   | the specified numeric attribute. values.   |                              |
| <b>SUMSQ</b><br><b>Exam-<br/>ples:</b> <code>SUM(bytesin)</code> •<br>event 1 bytesin = 3<br>and event 2 bytesin = 2<br>Result: 18<br>Derived from math:<br>$3^2 + 2^2 = 9 + 4$ | Calculates the sum of squares for all values for the specified numeric attribute.<br><br><b>Note</b><br>This function squares each value observed before perform addition. | <code>SUM(Field)</code>      |
| <b>VARIANCE</b><br><b>Examples:</b> <code>VAR-<br/>IANCE(bytesin)</code>  | Calculates the variance observed for the specified numeric attribute.  | <code>VARIANCE(Field)</code> |

## Data Processing Operators

These operators are used to perform actions on set of search results.

### Operators

| Command   | Description   | Syntax                           |
|---|---|----------------------------------|
| <b>HEAD</b><br><b>Examples:</b> <code>HEAD 10;</code><br><code>With Top: resource-<br/>groupname = OKTA  <br/>top accountname  <br/>HEAD 10;</code> <code>With<br/>STATS: resource-<br/>groupname = OKTA  <br/>STATS accountname  <br/>HEAD 10;</code> <code>With<br/>BARChart: resource-<br/>groupname = OKTA  </code> | Returns filtered results based on the condition<br><br><b>Note</b><br>MHEAD Operator should be used with following commands: TOP, RARE, STATS and BUBBLECHART | <code>HEAD &lt;number&gt;</code> |

| Command  | Description  | Syntax  |
|--|--|---|
| <code>BARChart account-name   HEAD 10</code>   |  |   |
| <b>WHERE</b><br><b>Examples:</b> <code>where count &gt; 10; With Top - resource-groupname = OKTA   top accountname   WHERE count &gt; 35; With Top &amp; ORDERBY - resourcegroupname = OKTA   top accountname   WHERE count &gt; 35   ORDERBY asc; With STATS: resourcegroupname = OKTA   stats accountname   WHERE count &gt; 35; With STATS &amp; ORDERBY: resourcegroupname = OKTA   STATS accountname transactionstring1   WHERE count &gt; 0   ORDERBY desc; With BARChart: resource-groupname = OKTA   BARChart account-name ipaddress   WHERE count &gt; 5</code> | Returns filtered results based on the condition<br><br><div> <b>Note</b><br/> WHERE command should be used with the following Operators: &gt; Greater than, &gt;= Greater than or equal to, &lt; Less than, &lt;= Less than or equal to </div> | <code>WHERE &lt;count&gt; &lt; = &gt; &lt;number&gt;</code>           |
| <b>ORDERBY</b><br><b>Examples:</b> <code>ORDERBY asc; Sort Events Descending order: resourcegroupname = Google_Login   STATS count by</code>   | Sort events by ascending or descending or field. Default asc or desc will sort events by count   | <code>ORDERBY &lt;asc or desc or &lt;field asc or desc&gt;&gt;</code> |

| Command   | Description   | Syntax |
|---|---|--------|
| <pre>ipaddress firstname   ORDERBY desc; Sort Field By As- cending order: re- sourcegroupname = Google_Login   STATS count by ipaddress firstname   ORDERBY firstname asc; Sort Field By Descending order: resourcegroupname = Google_Login   STATS count by ipaddress firstname   ORDERBY ipaddress desc</pre> | <p><b>Note</b></p> <p>ORDERBY command should be used with the following commands: TOP, RARE and STATS</p> |        |

## Additional Search Examples

See the following examples to find top risk users, malicious IP addresses, and other common queries.

### Examples

| Description   | Syntax  |
|---|---|
| <p>Get top risk users, activity accounts, activity IP addresses, and resources</p> <p><b>Examples:</b> <code>index=riskscore   top violatorid; Top Risk Users : index=riskscore and violator=Users   top violatorid; Top Activityaccount : index=riskscore and violator=Activityaccount   top violatorid; Top Activityip : index=riskscore and violator=Activityip   top violatorid; Top Resources :</code></p> | <pre>Index=riskscore &lt;and&gt; &lt;viola- tor&gt; = &lt;Users   Activityip   Activityaccount   Resources&gt;  &lt;top&gt; &lt;violator ID&gt;</pre> |

| Description  | Syntax   |
|--|--|
| <p><code>index=riskscore and violator=Activityip   top violatorid</code></p>   |  |
| <p>Get flight risk users</p> <p><b>Example:</b> <code>index=riskscore and violator = Users   Filter index = watchlist and entityname = violatorid</code></p>   | <p><code>Index=riskscore &lt;and&gt; &lt;viola-<br/>tor&gt; = &lt;Users&gt; ↓Filter&gt;<br/>&lt;index&gt; &lt;=&gt; &lt;watchlist&gt; &lt;and&gt;<br/>&lt;field1&gt; &lt;=&gt; &lt;field2&gt;</code></p>   |
| <p>Check if IP address is malicious</p> <p><b>Examples:</b> <code>Index=riskscore and violator = Activityip   Filter; index = tpi and addr = entityid and criticality = high</code></p>  | <p><code>Index=riskscore &lt;and&gt; &lt;viola-<br/>tor&gt; = &lt;Activityip&gt;   &lt;Filter&gt;<br/>&lt;index&gt; &lt;=&gt; &lt;tpi&gt; &lt;and&gt;<br/>&lt;field1&gt; &lt;=&gt; &lt;field2&gt; and<br/>&lt;field3&gt; &lt;=&gt; &lt;field4&gt;</code></p> |
| <p>Get information about assets on the network</p> <p><b>Example:</b> <code>Resourcegroupname = BCP1   index = asset and entityname = accountname</code></p>   | <p><code>Index = asset &lt;and&gt; &lt;field1&gt; = &lt;field2&gt;</code></p>  |
| <p>Check if user has sent email to personal email address</p> <p><b>Examples:</b> <code>resourcegroupname = "ADEvents"   EVAL matchPerc = emailtoSelf(firstname,workemail,0.4); resourcegroupname = "ADEvents"   eval x = SUBSTRBYINDEX (workemail , "@", "1" )</code></p> | <p><code>resourcegroupname = &lt;value&gt;  <br/>EVAL X = emailtoSelf<br/>(firstname,workemail,0.4)</code></p>   |



## Best Practice

This section describes the recommended best practices and provides support for frequently asked questions for searching in Spotter.

### Optimizing Queries

Follow these best practices to optimize queries.

- Use Operators that perform equals / exact matches.
- Limit operators to 3-5 for a search if the operators convert to a Regex match in Solr, as they will be CPU intensive.

### Refining Queries

Follow these best practices if a query times out:

1. Check if query is performing a transformation such as STATS.

If No, check the following:

- Time frame
- Search terms for excessive contains and wildcard usage
- Restrict to a single datasource

If Yes check the following:

- Number of unique values by replacing STATS with: distcount
- If value is excessively high use TOP/RARE Limit=#
- If value is low reduce timeframe and run again

2. Add additional search terms to reduce dataset which has to be counted

### Querying Threat Models

Follow these best practices to query threat models:

- Query the name of the threat model in the violation index when name is know.

- Use the following query to see all threat model violations when name is unknown: index= violation | FILTER index=riskscore and employeeid=employeeid and doctype = entity\_threatmodel

You will be unable to see the violated policies that resulted in the threat model violation.

## Support Queries

| Issue   | Validation   | Additional Notes  | Fix  |
|---|--|---|--|
| Geomap / Geolink not displaying points on the map | eventlatitude NOT NULL and eventlongitude NOT NULL | Substitute / add other fields from GEOMAP table to validate if they are populated | <ul style="list-style-type: none"> <li>• If enrichment is functioning, Open ticket for further support.</li> <li>• If no, validate enrichment in Activity import screens and further validate enrichment job.</li> </ul> |
| Geolookup is not populate Geolocation attributes  | index = geolocation                                | ipto and ipfrom must be populated   | <ul style="list-style-type: none"> <li>• If no data in index: import geolocation data (Maxmind)</li> <li>• If iptto and ipfrom are missing or has 0.0.0.0 the index is not valid. Re-import geolocation data</li> </ul>  |

| Issue                                   | Validation  | Additional Notes   | Fix   |
|---|---|--|---|
| Validate Unix Time Conversions          | EVAL x = to_unixtime(time_attribute)   eval y =from_unixtime(x,"MM/dd/yyyy HH:mm:ss")   FIELDS + x, time_attribute, y | <ul style="list-style-type: none"> <li>• Use external unix to human time convertor to validate.</li> <li>• All Times should match in human readable format</li> <li>• If field is missing from fields command switch to table</li> </ul> | Open ticket if they are not functioning   |
| Missing details on Violation Events Tab | index=violation and accountname = AC-COUNT_N_Question   Stats policy-name   | Validate policy name matches from SCC, trailing spaces are sometimes cut off.  | <ul style="list-style-type: none"> <li>• If match observed check continue does the name match when added to spotter search including all white-space characters?</li> <li>• If no match, check if any other violation details exist in risk-score and/or riskcorehistory</li> </ul> |

| Issue  | Validation   | Additional Notes  | Fix  |
|--|--|---|--|
| undefined field  | N/A  | <ul style="list-style-type: none"> <li>• Need to know data source name</li> <li>• Validate field is being parsed in activity import configuration screens</li> <li>• Validate parsed field is stored in collection as it may not exist in older collections.</li> </ul> | <ul style="list-style-type: none"> <li>• If field is being parsed send ticket to PM for further Support</li> <li>• If field is not being parsed work with field &amp; Content to parse and revalidate going forward</li> </ul> |
| SOLR Error: Unknown RefineMethod                       | Spotter Setting / Spotterconfig<br><br>Faceting Level > Refine   | SOLR 7 or newer<br><br>Default Value: true<br><br>Acceptable values: <ul style="list-style-type: none"> <li>• simple</li> <li>• TRUE</li> <li>• none</li> </ul>   | <ul style="list-style-type: none"> <li>• If Solr version is old, remove keys <b>refine</b> and <b>overrefine</b> from spotter-config in DB</li> <li>• Change value to true</li> </ul>  |
| Policy Do not show in spotter, but show current on SCC | index = violation   Stats policyname<br><br>For time frame select: <ul style="list-style-type: none"> <li>• Last 24 hours</li> </ul> | <ul style="list-style-type: none"> <li>• SCC is based on generation time</li> <li>• Spotter is based on event time</li> </ul>   | If the policy appears in a wider time range there is a delay in one of the jobs.<br><br>Example: policy A eventtime is 11/1  |

| Issue | Validation   | Additional Notes | Fix  |
|-------|--|------------------|--|
|       | <ul style="list-style-type: none"> <li>• Last 48 hours</li> <li>• Last 72 hours</li> </ul> |                  | <p>but is processed as a violation with generation time of 11/3</p> <p>This will not be found in Spotter unless generation time specified in query or time frame selected includes 11/1.</p> |

## Limitations

All of the following limitations are configurable in Configxml > spotter\_config. If you want to change beyond the default value, you must consider the Application Resources & the Solr Cell configuration / resources to maintain application stability. If changed while application is running, use Refresh Config button in Spotter expanded spotter search bar to apply.

### Warning

Capital letters, dashes (-), and spaces in the generated field name breaks the query when piped into other operators.

### Note

Only converts to 10 digit epoch (unix) time.

### Note

Each field refers to faceting of the previous field. Field1 is L1, Field2 is L2, etc.

### Note

You will be unable to see the violated policies that resulted in the threat model violation.

## Note

All of the following limitations are configurable in Configxml > spotter\_config. If you want to change beyond the default value, you must consider the Application Resources & the Solr Cell configuration / resources to maintain application stability. If changed while application is running, use Refresh Config button in Spotter expanded spotter search bar to apply.

**Max Bucket Size:** When you execute a Spotter Search that threatens to take the application out of memory by returning too many tuples (unique Values). The application will cancel the Query to maintain application stability.

- **Tag & default value:** <maxBucketSize>100000</maxBucketSize>
  - **UI message:** Query matched too many unique values. Query has been stopped to maintain application stability.
- **Tag & default value:** <maxBucketSize>100000</maxBucketSize>
  - **UI message:** Query matched too many unique values. Query has been stopped to maintain application stability.
- **Query timeout:** All spotter queries will be canceled at the 3 minute mark regardless of results returned, to prevent the underlying system, SOLR, from crashing which impacts the application itself
  - **Tag & default value:** <timeAllowed>3000</timeAllowed>
  - **UI Message:**
    - **0 Results:** Query Time out passed , current query is too resource intensive or search servers are busy. Please narrow your search.
    - **Partial Results:** Query time allowed exceeded. Partial results returned. Please refine query.
- **Allowed Faceting Fields for Commands:** Any command that using faceting / aggregation has a set limit of fields allowed to be utilized to maintain application stability.
  - **Tag & default value:** <allowedFacetFieldsForCommands>3</allowedFacetFieldsForCommands>
- **Leftside Maximum collections:** This is the number of SOLR collections that the system will search through for the attributes added to the Selected Fields panel on the left side for simple searches.

- **Tag & default value:** <leftSideMaxCollections>5</leftSideMaxCollections>
- **Facet Levels:** All aggregation commands and the time line that appears on spotter use faceting to obtain the counts of events as designed. These are used to maintain application stability. No message is displayed for this.
  - **Tags & default value:**

```
1  <facetLevelPageInfo>
2    <metaList>
3      <entry>
4        <key>L1</key>
5        <value>500</value>
6      </entry>
7      <entry>
8        <key>L2</key>
9        <value>100</value>
10     </entry>
11     <entry>
12       <key>L3</key>
13       <value>100</value>
14     </entry>
15     <entry>
16       <key>L4</key>
17       <value>100</value>
18     </entry>
19   </metaList>
20 </facetLevelPageInfo>
```



- **Others:**

- Transforming and Aggregation Operators do not cross Archive. The operators are performed in different systems and therefore can not be combined. If you need to perform these action for data that includes archive prefix the query with index=archive

## Facet Levels

Facet levels control the number of unique values returned for any operator that uses aggregation. The following application limits apply:

- A maximum of 10,000 values are returned on the interface for aggregation operators.
- Reports return a maximum of 100,000 values.

## Next-Gen SIEM

All indexes, except Archive, apply the facet level on a per-field basis. Each level returns a number of unique values up to the configured limit, with the next level applying to each value of the previous level.

- **Example 1:** `index = activity | Stats Accountname IPAddress`

- **Accountname:** L1 facet
- **IPAddress:** L2 facet

The query will return the following:

- Up to 500 unique accountnames
- Up to 100 unique ipaddress per accountname

## Archive Index

The Archive index uses a different storage technology that applies the limit as a total limit on the query.

- **Example 2:** `index = archive | Stats Accountname IPaddress`. The query returns up to 50,000 combined unique values [500 \* 100 = 50,000; L1 \* L2 = total to return].

### Unified Defense SIEM

All indexes, except Activity and Archive, apply the facet lever on a per-field basis. Each level returns a number of unique values up to the configured limit, with the next level applying to each value of the previous level.

**Example 1:** `index = violation | Stats Accountname IPaddress`

- **Accountname:** L1 facet
- **IPaddress:** L2 facet

The query will return the following:

- Up to 500 unique accountnames
- Up to 100 unique ipaddress per accountname

### Activity and Archive Index

The Archive index uses a different storage technology that applies the limit as a total limit on the query instead of by each group level.

**Example 2:** `index = activity | Stats Accountname IPaddress`. The query returns up to 50,000 combined unique values [500 \* 100 = 50,000; L1 \* L2 = total to return].

## Indicator Service (Beta)

The Indicator Service compiles all possible values for common Indicator of Compromise (IoC) data types and enables sub-second searches to help you quickly identify if an IoC has touched your environment. An IoC is evidence that a bad actor attempted to breach your network.

The Indicator Service allows you to scan IoCs received from Securonix and third-party intelligence by using multiple metadata indexes. These indexes ensure only a small subset of data is stored instead of scanning large segments of the Activity index.

### Note

Users will have indicator metadata available for the same search period as stated in their contract.

## Availability for the Indicator Service

The Indicator Service is available through an API using the existing search endpoints:

```
https://{BASE_URL}/ws/spotter/index/search?query=index=indicatorip AND ipaddress=50.167.20.139&eventtime_from= "11/15/2020 00:00:00"&eventtime_to= "11/15/2023 00:00:00"
```

## Indexes and Data Types

The following indexes and data types are available for the Indicator Service:

- **URLs:** Search `index = indicatorurl`.
- **IPv4 & IPv6 addresses:** Search `index = indicatorip`.
- **Hash values:** Search `index = indicatorhash`.
- **Domains:** Search `index = indicatordomain`.

## Metadata for Each Entry

- **lastseen:** The event time in which the value is observed.
- **resourcegroupname:** The resource group name of the log the value was ingested as a part of.
- **resourcegroupid:** The ID of the resource group name.
- **tenantid:** The ID of the tenant the value is associated with.
- **value:** The attribute value is based on the type of indicator and will vary per index. The value of the data point from the ingested log:
  - **Ipaddress:** `index = indicatorip`.
  - **url:** `index = indicatorurl`.

- **hash:** `index = indicatorhash`.
- **domain:** `index = indicatordomain`.

## Limitations

- **Masking:** The service is not aware of fields with masking. It only knows if masking is turned on or off in an environment. If masking is enabled, a privacy master will be required to search these indexes.
- **Granular RBAC:** The service is unaware of any granular permissions defined for users.

## Working with the Indicator Service (Beta)

### Note

To enable the Indicator Service, you must submit a service request. By default, the Indicator Service is disabled.

## To Observe IoCs in your Environment

To observe an IoC in your environment, complete the following steps.

- It is recommended to perform your search through an API.
- The indicator index cannot determine the field in which the data point is stored. For example, an IP address can have multiple fields.
- The Indicator Service does not include historical data.

1. In Unified Defense SIEM, go to **Menu > Security Center > Spotter**.
2. In the search bar, enter the indicator index that you want to search for. The following indicator indexes are available:
  - `index = indicatorurl`
  - `index = indicatorip`

- index = indicatorhash
- index = indicatordomain

**Example**

index=indicatorip and ipaddress = 99.88.252.90

3. When the search results return, use metadata to search events in activity index.

For example:

- ipaddress = 99.88.252.90
- resourcegroupid = 3476
- resourcegroupname = ResponseKaratePrerequisiteDS1
- lastseen = 1702556235727

**Note**

In the search, change **lastseen** to **eventtime**.

**Example**

index = activity and ipaddress = 99.88.252.90 and resourcegroupid = 3476 and eventtime <= 1702556235727

4. Export the results or create an incident.

## To Use the Indicator Service

1. In Unified Defense SIEM, go to **Menu > Security Center > Spotter**.
2. In the search bar, enter the indicator index that you want to search for.

**Example**

index=indicatorip and ipaddress = 99.88.252.90

3. Use the metadata to search events in the activity index.

**Example:**

- **ipaddress:** 99.88.252.90.
- **resourcegroupid:** 3476.
- **resourcegroupname:** ResponseKaratePrerequisiteDS1.
- **lastseen:** 1702556235727.

4. In the search bar, change **lastseen** to **eventtime**.

**Example**

index = activity and ipaddress = 99.88.252.90 and resourcegroupid = 3476 and eventtime <= 1702556235727

You may need to use **rawevent contains** to locate the indicator value, as the field is unknown.

5. Export the results or create an incident.

## Data Masking in Spotter

This section explains data masking in Spotter search. All tasks and validations in this section apply to both MSSP and non-MSSP deployments.

### Tasks Performed by Unified Defense SIEM

To reduce cross-tenant and cross resource group conflicts in data masking configurations, Unified Defense SIEM will perform the following tasks before querying data.

#### Extracting Tenants

Tenant extraction will filter out the tenant in the query based on the following:

- When the **tenantname** or **tenantid** parameter is used in a query, Unified Defense SIEM will only query data for that particular tenant.

- When `resourcegroupname`, `policyname`, or `resourcegroupfunctionality` is used in query, Unified Defense SIEM extracts the respective tenant ID for the query.
- Unified Defense SIEM extracts tenants for all or multiple resource groups.

### Filtering Tenants with Unmasked Fields

- When a query has fields with unmasked values, additional filtering is applied on tenants.
- Unified Defense SIEM will exclude all the tenants from the query in which a particular field has masking enabled and a user is searching with unmasked values.

#### Example

`index = activity and accountname="xyz"`

The previous example will query data where `accountname` is unmasked for the specified tenants.

### Consolidating Masked Fields

When data is queried from more than one tenant:

- Masking is applied based on the particular tenant configuration when `tenantid` is shown in the results.

In the following example, the records have a variation of masked and unmasked data:

```
index=activity | table accountname ipaddress tenantid
index=activity | stats accountname ipaddress tenantid
index=activity
```

In the previous example, the `tenantid` is in response.

The data in the following image is based on the tenant masking configuration:

The screenshot displays the Spotter Security Center interface. The top navigation bar includes a menu, search bar, and user profile. The main content area shows search results for the query: `INDEX = ACTIVITY AND ( RESOURCEGROUPNAME = "SK-01" OR RESOURCEGROUPNAME = "TSDS1" ) | STATS ACCOUNTNAME ipaddress tenantid`. The results table has columns: ACCOUNT, IPADDRESS, TENANTID, and COUNT.

| ACCOUNT                          | IPADDRESS                       | TENANTID | COUNT |
|----------------------------------|---------------------------------|----------|-------|
| 03EF175C4381103071BC9983F2AFA230 | A059781427E0A9E498B930CEBEB5142 | 2        | 13    |
| EA02B8B154DD3F2F4A0E2ABC1ABF742D | BD6F9495EAB07E7F3198481C2764AF  | 2        | 11    |
| TSDS1                            | 108.35.55.230                   | 3        | 4     |
| TSDS1                            | 10.21.1.195                     | 3        | 1     |
| TSDS1                            | 10.21.2.214                     | 3        | 1     |
| TSDS1                            | 106.51.16.222                   | 3        | 1     |
| TSDS1                            | 111.93.188.90                   | 3        | 1     |

Below the table, detailed logs are shown for specific events. The first log entry is dated TUE, 2 FEB 2021 @ 08:15:48 AM, resourcegroupname: TSDS1. It shows a successful login for accountname = TSDS1, transaction = An account was successfully logged on, with various metadata fields like ipaddress, resourcegroupid, resourcegroupname, rg\_functionality, rg\_vendor, alertid, email, and categoryseverity.

The second log entry is dated TUE, 2 FEB 2021 @ 08:15:48 AM, resourcegroupname: TSDS1. It shows special privileges assigned to a new login for accountname = TSDS1, transaction = Special Privileges Assigned to new login, with metadata fields like ipaddress, resourcegroupid, resourcegroupname, rg\_functionality, rg\_vendor, department, employeeid, firstname, lastname, status, workemail, usercriticality, enabledate, lastsyncdate, userscore, and usertimezoneoffset.

The third log entry is dated FRI, 15 JAN 2021 @ 04:34:51 AM, resourcegroupname: sk-01. It shows a network share object accessed for account = 24FB16682C34529FEF3BD09E077DF1E, transaction = A network share object was accessed, with extensive metadata including ip, resourcegroupid, resourcegroupname, rg\_functionality, rg\_vendor, companycode, costcentername, country, department, division, employeeid, employeetype, employeetypedescription, fulltime, hiredate, jobcode, land, lastname, location, manageremployeeid, status, staturesdescription, title, workemail, workphone, mobile, usercriticality, companynumber, street, street2, realmtype, hierarchy, costcentercode, enabledate, managerlastname, customerfield1, customerfield2, usertimezoneoffset, province, networkid, zipcode, orgnumber, city, savannah, lastsyncdate, managerfirstname, fulltimeparttime, and userscore.

- If the results do not have `tenantid`, then masking is performed based on consolidated masking fields across tenants that are used in the query.

For example:

- Tenant one masked the `accountname` field.
- Tenant two masked the `ipaddress` field.

Both `accountname` and `ipaddress` are masked in the results.

```
INDEX = ACTIVITY AND ( RESOURCEGROUPNAME = "SK-01" OR
RESOURCEGROUPNAME = "TSDS1" ) | STATS ACCOUNTNAME ip
address
```



In the following image, TSDS1 masking is disabled, but for SK-0, masking is enabled. Both resource groups belong to different tenants and data masking is applied based on common masked fields.

| ACCOUNT                          | IPADDRESS                        | COUNT |
|----------------------------------|----------------------------------|-------|
| 03EF175C4381103071BC9983F2AFA230 | A0597B1427EDAE498B930CEB8B5142   | 13    |
| EA02B8154DD3F2F4A0E2ABC1ABF742D  | BDFAF9495EAB07E1F73198481C2764AF | 11    |
| 51F4903F91499994DDEDB1F339C6FE79 | 4AC3ABE8691BCAF020E744A8A28C69F  | 4     |
| 51F4903F91499994DDEDB1F339C6FE79 | F4E0AE1D3B07088208789FC34473E94  | 1     |
| 51F4903F91499994DDEDB1F339C6FE79 | CE4397139BAAF6C78E222DC39F6F7482 | 1     |
| 51F4903F91499994DDEDB1F339C6FE79 | 252AF38479188EC447E94397A18C16   | 1     |
| 51F4903F91499994DDEDB1F339C6FE79 | 053691473CA2088C7850400B354C0C6E | 1     |
| 51F4903F91499994DDEDB1F339C6FE79 | E2AB216A087CEB54CCA404D5002A6    | 1     |
| 4FBCD29D895D3B4C0FF2249A0F7E3492 | 01309D112BA6FE2301D9524E52BAB0E5 | 8     |
| 20CF1C3C28BF877C91630B1BE1F9A9FC | 7BA85C75D68E5F13A02A98AF09EBB49  | 8     |
| 93E1F5B31A6289505218EBD9FBB06126 | AA35063EA44742B62B739CC4026CC7   | 8     |

- Data masking is performed based on consolidated masked fields, as seen in the following image:

| Attribute                        | Count |
|----------------------------------|-------|
| 03EF175C4381103071BC9983F2AFA230 | 13    |
| EA02B8154DD3F2F4A0E2ABC1ABF742D  | 11    |
| 51F4903F91499994DDEDB1F339C6FE79 | 9     |
| 20CF1C3C28BF877C91630B1BE1F9A9FC | 8     |
| 4FBCD29D895D3B4C0FF2249A0F7E3492 | 8     |
| 93E1F5B31A6289505218EBD9FBB06126 | 8     |
| A74B98A7766F0365673F770837DEC1DB | 7     |

- There are scenarios where Unified Defense SIEM cannot find a relevant tenant for a query, as seen in the following example:

```
index=activity
```

In the previous example, Unified Defense SIEM will query data from all tenants assigned to a user.

### Corner Cases

- Results will not return if a user without privacy master permissions searches for a masked value and masking is disabled for that field across all of the users accessible tenants.
- User masking is dependent on the `u_masked` value stored in the user index. Verify that the user attribute is masked. The masking job will run properly and `u_masked` is true in user index.
- When multiple tenants are used in a query and searches have masked values, results may display unmasked values. This is dependent on the masking configuration for tenants.

## Using Special Characters in Spotter

Special characters are handled in the Spotter UI and web service API. It is important to consider the following points before writing a query:

- **Special Characters:** All special characters can be searched in Spotter. However, only (? , \*, " and \) have special meanings throughout the application, so an escape sequence is required just before those special characters.
- **Escape Sequence:** Use an escape sequence before a special character to ensure it is treated as a literal character in the query.
- **Backslash (\):** In the response of the Spotter API, it will show as double the number of backslashes since JSON itself considers backslash as an escape sequence.

## What's Changed

| Character | Before<br>April_2024_R1     | On/After<br>April_2024_R1   | What is change                    |
|-----------|-----------------------------|-----------------------------|-----------------------------------|
| "         | Was not Working             | Data: \"Escape Sequence: \" | Need to escape \" with backslash. |
| \         | Data: \"Escape Sequence: No | Data: \\Escape Sequence: \" | Need to escape \" with backslash. |

### Example

If you want to search for a single backslash attribute value, use `customstring7="t\\cp"`.

If the query formation includes a double backslash, (`customstring7="t\\\\cp"`), the response will show double as well. See below:

GET https://a1t2qate.securonix.net/Snypr/ws/spotter/index/search?eventtime\_from=10/01/2023 00:00:00&...

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

eventtime\_to 12/30/2023 23:59:59

query index = archive and resourcegroupname = "6thOctHunt" and @customstring7 = "t%5C%5Ccp"

data\_labels

200 OK 4.94 s 5.65 KB Save Response

Pretty Raw Preview Visualize Text

```

58     "rg_ipaddress": "",
59     "transactionstring4": "<never>",
60     "minute": "4",
61     "customstring5": "?format=embeddedhtml\\",
62     "destinationuserid": "S-1-5-21-1214440339-1454471165-6839522115-15620731",
63     "categoryseverity": "0",
64     "customstring61": "Security",
65     "customstring8": "\\bs1",
66     "customstring7": "t\\cp",
67     "sourceusername": "iSACA0DMGR",
68     "sourceaddress": "10.0.100.69",
69     "dayofweek": "1",
70     "deviceeventcategory": "User Account Management",
71     "ingestionnodeid": "CONSOLE"
72   }
73 ],
74   "error": false,

```

customstring7 Aa Abi \* 1 of 2 ↑ ↓ ≡ ×

## Escape Sequences

The following table provides examples of special characters and their corresponding escape sequences:

| Special Character | Escape Sequence |
|-------------------|-----------------|
| ?                 | \?              |
| *                 | \*              |
| \                 | \\              |
| "                 | \"              |

## Specific Meanings

In addition, the following special characters have specific meanings:

**\* (Asterisk):** Used to represent 0 or more wildcard characters in a search.

**? (Question mark):** Used to represent a single wildcard character in a search.

**\ (Backslash):** Because \ is used as the escape sequence, an additional backslash is required to use the backslash as a string character.

**" (Quotes):** Use around phrases that contain white spaces so they are not treated as multiple attributes.

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query   |
|-------------------|------------------------|------------------------|--|--|
| \n                | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\n"cp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5Cncp%22 |
| #                 | No                     | No                     | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t#cp"   | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3                     |

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query   |
|-------------------|------------------------|------------------------|--|--|
|                   |                        |                        |  | D%20%22t%23cp%22   |
| ;                 | No                     | No                     | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t;cp"   | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%3Bcp%22     |
| \t                | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\ttcp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5Ctcp%22 |
| \$                | No                     | No                     | index = activity and resourcegroupname = "6thOctHunt" and @custom-                   | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOc-   |

| Special Character | index=activity | index=archive | Query Plain   | Example of Encoded Query  |
|-------------------|----------------|---------------|---|---|
|                   |                |               | string7 = "t\$cp"   | tHunt%22%20and%20%40custom-string7%20%3D%20%22t%24cp%22   |
| '                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and @custom-string7 = "t'cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%27cp%22 |
| _                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and @custom-string7 = "t_cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t_cp%22   |

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query   |
|-------------------|------------------------|------------------------|--|--|
| \                 | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\ \cp"   | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5Ccp%22    |
| \\                | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\\ \cp"  | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5C%5Ccp%22 |
| \\\               | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\\\ \cp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40cu  |



| Special Character | index=activity         | index=archive          | Query Plain   | Example of Encoded Query   |
|-------------------|------------------------|------------------------|---|--|
|                   |                        |                        |   | stom-string7%20%3D%20%22t%5C%5C%5C%5C%5Ccp%22  |
| \\                | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\\cp"   | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5C%5C%5C%5Ccp%22 |
| \\\\              | Escape sequence Needed | Escape sequence Needed | index = activity and resourcegroupname = "6thOctHunt" and @custom-string7 = "t\\\\cp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5C%5C%5C%5Ccp%22 |

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query  |
|-------------------|------------------------|------------------------|--|---|
|                   |                        |                        |  | C%5C%5Ccp%22  |
| -                 | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t-cp"   | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t-cp%22       |
| \-                | Escape sequence Needed | Escape sequence Needed | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t\\-cp" | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5C-cp%22 |
| ~                 | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and @custom-                   | index%20%3D%20archive%20and%20resource-   |

| Special Character | index=activity         | index=archive          | Query Plain   | Example of Encoded Query  |
|-------------------|------------------------|------------------------|---|---|
|                   |                        |                        | string7 = "t~cp"  | group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t~cp%22  |
| \~                | Escape sequence Needed | Escape sequence Needed | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t\~cp" | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%5C~cp%22 |
| +                 | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t+cp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3                     |

| Special Character | index=activity | index=archive | Query Plain   | Example of Encoded Query   |
|-------------------|----------------|---------------|---|--|
|                   |                |               |   | D%20%22t%2Bcp%22   |
| \+                | No             | No            | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t\+cp" | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C%2Bcp%22 |
| :                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t:cp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%3Acp%22    |
| %                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and @custom-                  | index%20%3D%20archive%20and%20resource-  |

| Special Character | index=activity | index=archive | Query Plain  | Example of Encoded Query   |
|-------------------|----------------|---------------|--|--|
|                   |                |               | string7 = "t%cp"   | group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%25cp%22                                     |
| `                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t`cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%60cp%22 |
| !                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t!cp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3                 |

| Special Character | index=activity         | index=archive          | Query Plain   | Example of Encoded Query   |
|-------------------|------------------------|------------------------|---|--|
|                   |                        |                        |   | D%20%22t!<br>cp%22   |
| \b                | Escape sequence Needed | Escape sequence Needed | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t\bcp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%5C%5Cbc%22  |
| \t                | Escape sequence Needed | Escape sequence Needed | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t\tcp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%5C%5Ctcp%22 |
| \r                | Escape sequence Needed | Escape sequence Needed | index = activity and resource-groupname = "6thOctHunt" and custom-                  | index%20%3D%20activity%20and%20resourcegroup-name%20%3D  |

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query  |
|-------------------|------------------------|------------------------|--|---|
|                   |                        |                        | string7 = "t\lrp"  | %20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%5C%5Crcp%22   |
| "                 | Escape sequence Needed | Escape sequence Needed | index= archive and resource-groupname = "HuntingAutomationTestRg" AND tenant-name = "TestOctT64" and olduacvalue starts with "double-quote"" | index%3D%20archive%20and%20resource-group-name%20%3D%20%22HuntingAutomationTestRg%22%20AND%20tenant-name%20%3D%20%22TestOctT64%22%20and%20%20olduacvalue%20starts%20with%20%22double-quote%5C%22%22 |
| ^                 | No                     | No                     | index = activity and resource-groupname = "6thOctHunt" and custom-   | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOct-  |

| Special Character | index=activity | index=archive | Query Plain  | Example of Encoded Query   |
|-------------------|----------------|---------------|--|--|
|                   |                |               | string7 = "t^cp"   | tHunt%22%20and%20custom-string7%20%3D%20%22t%5Ecp%22   |
| @                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t@cp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%40cp%22 |
| &                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t&cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%26cp%22 |
| (                 | No             | No            | index = archive and resource-  | index%20%3D%   |



| Special Character | index=activity | index=archive | Query Plain  | Example of Encoded Query   |
|-------------------|----------------|---------------|--|--|
|                   |                |               | groupname = "6thOctHunt" and custom-string7 = "t(cp"                               | 20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t(cp%22             |
| )                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t)cp"  | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t)cp%22 |
| <                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t<cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20custom-                           |

| Special Character | index=activity | index=archive | Query Plain  | Example of Encoded Query  |
|-------------------|----------------|---------------|--|---|
|                   |                |               |  | string7%20%3D%20%22t%3Ccp%22  |
| >                 | No             | No            | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t>cp"  | in-dex%20%3D%20ar-chive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20cus-tom-string7%20%3D%20%22t%3Ecp%22 |
| [                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t[cp" | in-dex%20%3D%20activi-ty%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20cus-tom-string7%20%3D%20%22t%5Bcp%22 |
| ]                 | No             | No            | index = activity and resource-groupname = "6thOctHunt"                             | in-dex%20%3D%20activi-ty%20and%20resourcegroup-   |

| Special Character | index=activity | index=archive | Query Plain  | Example of Encoded Query  |
|-------------------|----------------|---------------|--|---|
|                   |                |               | and custom-string7 = "t]cp"  | name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%5Dcp%22   |
| {                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t{cp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%7Bcp%22   |
| }                 | No             | No            | index = activity and resource-groupname = "6thOctHunt" and custom-string7 = "t}cp" | index%20%3D%20activity%20and%20resourcegroupname%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%257Dcp%22 |

| Special Character | index=activity         | index=archive          | Query Plain  | Example of Encoded Query   |
|-------------------|------------------------|------------------------|--|--|
| .                 | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t.cp"    | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t.cp%22   |
|                   | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and custom-string7 = "t cp"    | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20custom-string7%20%3D%20%22t%7Ccp%22 |
| ?                 | Escape sequence Needed | Escape sequence Needed | index = activity and resource-groupname = "6thOctHunt" and @custom-string7 = "t\?cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20  |

| Special Character | index=activity         | index=archive          | Query Plain   | Example of Encoded Query   |
|-------------------|------------------------|------------------------|---|--|
|                   |                        |                        |   | and%20%40custom-string7%20%3D%20%22t%5C%3Fcp%22  |
| *                 | Escape sequence Needed | Escape sequence Needed | index = activity and resource-groupname = "6thOctHunt" and @custom-string7 = "t\\*cp" | index%20%3D%20activity%20and%20resourcegroup-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%5C*cp%22 |
| /                 | No                     | No                     | index = archive and resource-groupname = "6thOctHunt" and @custom-string7 = "t/cp"    | index%20%3D%20archive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%2Fcp%22  |
| =                 | No                     | No                     | index = archive and resource-groupname =  | index%20%3D%20ar-  |

| Special Character | index=activity | index=archive | Query Plain                                     | Example of Encoded Query   |
|-------------------|----------------|---------------|---|--|
|                   |                |               | "6thOctHunt"<br>and @custom-string7 =<br>"t=cp" | chive%20and%20resource-group-name%20%3D%20%226thOctHunt%22%20and%20%40custom-string7%20%3D%20%22t%3Dcp%2 |