Hewlett Packard
Enterprise

HPE Security ArcSight Investigate

Software Version: 1.01

Query Quick Reference Guide

June 15, 2017

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

The network information used in the examples in this document (including IP addresses and hostnames) is for illustration purposes only.

HPE Security ArcSight products are highly flexible and function as you configure them. The accessibility, integrity, and confidentiality of your data is your responsibility. Implement a comprehensive security strategy and follow good security practices.

This document is confidential.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Follow this link to see a complete statement of copyrights and acknowledgements:
https://community.saas.hpe.com/t5/Discussions/Third-Party-Copyright-Notices-and-License-Terms/td-p/1589228

Support

Contact Information

Phone	A list of phone numbers is available on the HPE Security ArcSight Technical Support Page: https://softwaresupport.hpe.com/documents/10180/14684/esp-support-contact-list
Support Web Site	https://softwaresupport.hpe.com
Protect 724 Community	https://community.saas.hpe.com/t5/ArcSight/ct-p/arcsight

Contents

Informal search input query language specifications

Search input is a control, where the main query condition is built in ArcSight Investigate. This guide describes constructs provided in that control.

Search input query language is a simplified SQL `WHERE` clause with some "syntactic enhancements" designed to help you express complicated search constructs.

The goal of the search input query is to restrict a search to events having particular values and located in specific columns.

Tuples

A *tuple* is a building block of the query. A tuple restricts one or several columns as follows:

```
<tuple> ::= <column-list>? <operator>? <value-list>
```

```
<column-list>? <range-operator> <value> and <value>
```

Example query:

```
Request URL = www.test.com
```

This query consists of one tuple, which has a column list with one column and a value list with one value.

Lists

Column lists and value lists use commas as separators:

- `<value-list> ::= <value> (, <value>)*`
- `<column-list> ::= <column-expr> (, <column-expr>)*`

Note:

- For multiple fields left of the operator, `OR` is implied between the fields. For example, `<field-1>, <field-2> = <value>` is equivalent to `<field-1> OR <field-2> = <value>`
- To have an `AND` condition for multiple fields, set each field equal to a specific value, with the `AND` operator separating the fields. For example, `<field-1>=<value-1> AND <field-2>=<value-2>`

Examples

- `Request URL = www.test.com, www.test2.com, www.test3.com`

This query searches for events having a Request URL equal to one of the three URLs.

- `File Name, Destination Process Name = malware.exe`

  This query finds events having `File name` or `Destination Process Name` equal to `malware.exe`.

- `Source Address, Destination Address = 127.0.0.1, 192.168.1.1`

  This query searches for events with `Source Address` or `Destination Address` equal to 127.0.0.1 or 192.168.1.1. Any of the four cases satisfies this query: a) Source Address = 127.0.0.1, b) Destination Address = 127.0.0.1, c) Source Address = 192.168.1.1, and d) Destination Address = 192.168.1.1).

  The use of a comma in the list depends on the operator and is natural. For operators that do not have `not` in their name it is `OR`, for operators with `not` it is `AND`.

- `Source Address, Destination Address != 127.0.0.1, 192.168.1.1`

  This query searches for events that have both `Source Address` not equal to 127.0.0.1 or 192.169.1.1 and `Destination Address` not equal to 127.0.0.1 or 192.169.1.1. In any of the four following cases the event will not satisfy the query: a) Source Address = 127.0.0.1, b) Destination Address = 127.0.0.1, c) Source Address = 192.168.1.1, and d) Destination Address = 192.168.1.1).

Values

Investigate automatically detects the type of values used in the search input. You do not need to use quotes to specify string values unless a value contains query language syntax.

```
<value> ::= <non-escaped-value> | <escaped-value>
```

```
<escaped-value> ::= '<text-without-single-quote>' | "<text-without-double-qoute>"
```

```
<non-escaped-value> ::= <text-without-language-constructs>
```

For example, the following query does not need any quotes for values:

```
Device Vendor = blue coat
```

Querying some fields may require the use of quotes to disambiguate the value from the query language:

```
Raw Event contains 'Name = "test" and sourcePort = 8080'
```

Some other examples include the use of special values such as `null` (see "Value formats" on page 10). This query searches for events that have no `Source Username` specified:

```
Source Username = null
```

This query will search for events that have `Source Username` equal to string `Null`:

```
Source Username = "null"
```

There are two types of quotes, double and single. They are interchangeable except when you have quotes inside your value. The following two queries are interchangeable:

- `Raw Event contains 'sourcePort = 8080'`
- `Raw Event contains "sourcePort = 8080"`

Use single quotes if your value contains double quotes or double quotes if your value contains single quotes:

- `Raw Event contains 'Name = "test" and sourcePort = 8080'`
- `Raw Event contains "Name = 'test' and sourcePort = 8080"`

Quotes are taken into account only if found at the beginning of the value. Otherwise, they are treated as a regular value characters. The following query will find events with `Source Username` equal to `O'Brien`:

```
Source Username = O'Brien
```

Column expressions

Column expressions enable you to specify a particular column name or use a special alias to denote a group of columns.

- `<column-expr> ::= <column name> | <group-expression>`
- `<group-expression> ::= (any | event) <group-name>? | <group-name>`

Column (field) names can be any of the column names available in ArcSight Investigate. They are not restricted to the fieldset of the query. In the search field of the Search page, Investigate underlines an unknown column and then suggests a fix if one is available.

Column names are case-insensitive. Many columns have shortcut synonyms. Some columns have more than one synonym. For example, `Source Address` can be referred to as `sourceAddress`, `src`, `source ip`, `src ip`, and `source ip`.

Group names are shortcuts to common lists of columns. The following group names are supported:

category – list of all category fields

domain – list of all domain fields

custom float – list of all custom float fields

hostname – list of two columns, Source Hostname and Destination Hostname

id – list of all id columns

ip – list of all IP address columns (`agentAddress`, `destinationAddress`, etc.)

ip6 – list of all IP6 address columns

label – list of all label columns

mac – list of all mac address columns

path – list of all path columns

port – list of all port columns

timestamp or **time** – list of all time columns (`End Time`, `Start Time`)

uri – list of all uri columns

url – list of all url columns

username or **user** – list of two columns, `Source Username` and `Destination Username`

Example

The following interchangeable queries search for events with `Source Port` or `Destination Port` equal to 8080:

- `any port = 8080`
- `port = 8080`

If you do not specify a group name, the query will infer the CEF type of the value and run a query over all columns that satisfies the value type. For example, the following query searches for events with any of the integer fields equal to 10:

`any = 10`

The use of `any` with string values triggers a free text search.

Operators

The query language supports the following operators:

- `equals`
  alternatives: `=`, `==`, `is equal to`, `equal`
- `not equal`
  alternatives: `<>`, `!=`, `not equal`
- `less than`
  alternatives: `<`, `is less`, `is lower`, `less`
- `greater than`
  alternatives: `>`, `is greater`
- `less equal than`
  alternatives: `<=`, `lte`, `less equal`
- `greater equal than`
  alternatives: `>=`, `gte`, `greater equal`
- `contains`
  alternatives: `contain`, `like`, `has substring`
- `does not contain`
  alternative: `does not have`
- `starts with`

alternative: `startswith`

- `ends with`
  alternative: `endswith`
- `does not start with`
- `does not end with`

The `equals` and `not equal` operators work with columns of any type. The `less`, `greater`, `less equal`, `greater equal` work only with numbers. The `contains`, `starts with`, and `ends with` operators work only with strings. A warning message appears if you try to use an operator with a column of a type that does not accept that operator.

If you omit an operator, the query will infer one. If both `column list` and `value list` are present, `equals` will be used. If only `value list` is present, the operator will be deduced based on the value type. It will be `contains` for strings and `equals` for other types.

For example, the following queries assumes the `equals` operator:

- `username Mike`
- `8080`

The following query will assumes the `contains` operator:

```
Microsoft
```

In addition to binary operators, Investigate supports range operators:

- ibt, between – in between
- `nibt, not between` – not in between

Range operators take a column list and two values.

Example

```
Source Port between 0 and 32565
```

Omitting operators and column expressions

You can omit column names and operators. Omitting column names leads to value type being used to determine the list of columns. Omitting operator leads to value types or column types used to determine the omitted operator.

There is one invalid omitting combination: omitting `groupname` and operator between any and a value. The following query is invalid and will have red underline in the Search page.

```
any John Doe
```

If you want to construct an equivalent valid query use:

```
equals John Doe
```

Note that the query `John Doe` is equivalent to `contains John Doe`

Note that you can omit the operator after `any` if you have a group name like in the following query:

```
any user John Doe
```

Or you can simply use:

```
user John Doe
```

Combining tuples

Combining with logic

If you want to combine several tuples to further narrow your query, you can use and and or logic in between. As usual, and has a stronger binding than or. To form complicated conditions, you can use parenthesis.

Examples

```
<sub-query> ::= <or-element> (or <or-element>)*
```

```
<or-element> ::= <and-element> (and <and-element>)*
```

```
<and-element> ::= <tuple> | <lpar> <sub-query> <rpar>
```

```
<lpar> ::= (
```

```
<rpar> ::= )
```

The following query demonstrates basic use of parenthesis:

```
Category Device Group = /Firewall and Category Object starts with
/Host/Application/Service and (Category Behavior starts with /Access or
Category Behavior = /Communicate/Query) and Category Outcome = /Failure
```

Parenthesis can be arbitrary nested.

The and/or logic has some alternatives that can be used in the query formulation. You can replace and with &, &&, connecting to, with, and for. You can replace or with | and ||. Special forms of and (for, connecting to, with) allow you to better express intent of the query and enable Investigate to display more precise warnings and errors, as in the following example:

```
#Failed logins for user Mike
```

Cross-referencing

Investigate enables you to reference a query in another query. Whenever you use a query name in place of a tuple, the query is referenced. To indicate the use of a cross-referenced query, attach a pound sign (#) to the beginning of the referenced query name, as in the following example:

```
#Failed logins for user Mike
```

Queries are referenced by name. If multiple queries have the same name, the first one will be used. There are also several preset queries, which you can reference as any other query. You can also hide them by creating a query with the same name as a preset one.

Preset queries list:

```
Malicious code activity

Firewall drop

Windows account creation

Failed logins

Failed logins for $username

SSH authentication
```

A cross-referenced query is a textual substitution. Suppose you have a query named **Failed logins**:

```
categoryBehavior = /Authentication/Verify and categoryOutcome = /Failure
```

The following query,

```
#Failed logins for user Mike
```

would be equivalent to:

```
(categoryBehavior = /Authentication/Verify and categoryOutcome = /Failure)
for user Mike
```

Value formats

Special values

Some values have special meaning. All values with special meaning are written without single or double quotes. If you need to use a special value as a regular string, just put quotes around as was mentioned before. For example, the following query searches events with `Source Username` being `NULL` in the database:

```
Source Username = Null
```

If you want to search for users with the name `'Null'`, add single or double quotes:

```
Source Username = 'Null'
```

The following values are treated in a special way:

- `Null`, `NULL`, `null`, when used after `equals` or `not equals` or their synonyms, works as `IS/IS NOT NULL`
- A timestamp in ISO 8601 format is converted to the integer format when running a query.

  Example

  The following timestamp will be converted to integer time:

  ```
  End Time = 2017-01-01 00:00:00
  ```

When translated to SQL, this expression would be transformed to Epoch time using your local time zone. For example, `endTime = 1483257600000`

- Values starting with a dollar sign ($) indicate a template query.

  Before running the template query, replace the $-value with information for which you would like to search.

  Example

  The following query requires you to modify `$username`:

  ```
  #Failed logins for user $username
  ```

- Values starting with # are treated as references if they point to an existing query. The search field shows an underline with yellow if a reference cannot be resolved. In this case, the query uses a reference as a value. In the example above, `#Failed logins` is a reference.

Value types

A value type is associated with each value. The value type depends on the value format. The value type determines suggestions and column lists in case no operators and/or columns are provided.

Investigate recognizes the following value types:

- Null literal value
- IPv4 address
- IPv6 address
- DNS domain
- Unsigned integer
- Floating point number
- Time stamp in ISO 8601 format
- MAC address

Adding and removing quotes does not change the value type except for `null` and time stamp types.

Example

The following queries are valid and interchangeable:

- `Source Address = 127.0.0.1`
- `Source Address = '127.0.0.1'`

Search input use cases

In the examples below | indicates the caret position. All suggestions are listed according to the present SuperSchema. Suggestions might change if another superschema is used.

– query behavior is fine and unlikely to change.

– query behavior has minor issues and may change.

– query behavior has major issues and will eventually change.

When testing suggestions, note that spaces before and/or after caret affect the behavior of suggestions in a meaningful way. Some of the examples below show how.

Empty query suggestions

Query: |

Popup aligned to the beginning of search input field.

Expected suggestions:

```
Source Address
Destination Address
Source Username
Destination Username
Source Hostname
Destination Hostname
Destination DNS Domain
Source DNS Domain
Device Address
Request URL
any category
any custom float
any domain
any hostname
any id
```

```
any ip
any ip6
any label
any mac
...
```

Column names suggestions

Query: `1.1.1.1 = A`

Popup aligned to the beginning of A.

Expected suggestions:

```
Source Address
Destination Address
Device Address
any ip
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

Query: `Dest = 1.1.1.1`

Popup aligned to the beginning of `Dest`.

Expected suggestions:

```
Destination Address
Destination Username
Destination Hostname
Destination DNS Domain
Destination Geolocation Info
Destination Mac Address
Destination NT Domain
Destination Port
```

```
Destination Process ID
Destination Process Name
Destination Service Name
Destination Translated Address
Destination Translated Port
Destination Translated Zone URI
Destination User ID
Destination User Privileges
Destination Zone URI
Source Address
Destination Address
...
```

Query: `Source = 1.1.1.1`

Popup aligned to the beginning of `Source`.

Expected suggestions:

```
Source Address
Destination Address
Device Address
any ip
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
Source Address
Source Username
Source Hostname
Source DNS Domain
Category Significance
Severity
Source Geolocation Info
```

```
Source Mac Address
Source NT Domain
Source Port
...
```

Query: `ByteSource Address = 1.1.1.1`

Popup aligned to the beginning of `ByteSource`.

Expected suggestions:

```
Bytes In =
Bytes Out =
Source Address
Destination Address
Device Address
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

Query: `Source Address| 1.1.1.1`

Popup aligned to the beginning of `Source Address`.

Expected suggestions:

```
=
Destination Address
Device Address
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
Source Address =
```

Query: `1.1.1.1 equals Source Address`

Popup aligned to the beginning of `Source Address`.

Expected suggestions:

```
Destination Address
Device Address
any ip
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

⚠ Query: `1.1.1.1 equals Source Address`

Popup aligned to the beginning of `Source Address`.

Expected suggestions:

```
and
or
```

✔ Query: `Device Action = Amazon`

Popup aligned to the beginning of `Device Action`.

Expected suggestion:

```
Device Vendor
```

Text value suggestions

✔ Query: `Source P`

Popup aligned to the beginning of `Source`.

Expected suggestions:

```
Source Port =
Source Process ID =
Source Process Name =
Source Username = Source P
Destination Username = Source P
Source Hostname = Source P
```

```
Destination Hostname = Source P
Destination DNS Domain = Source P
Source DNS Domain = Source P
Agent DNS Domain = Source P
Agent Hostname = Source P
Destination NT Domain = Source P
Device DNS Domain = Source P
Device Domain = Source P
Device Hostname = Source P
Device NT Domain = Source P
Source NT Domain = Source P
```

⚠ Query: `Source Port = 2`

Popup aligned to the beginning of `Source`.

Expected suggestions:

```
and
or
```

✅ Query: `Device Vendor = Amazon`

Popup aligned to the beginning of `Amazon`.

Expected suggestions:

```
AirMagnet
Apache
Application Security
Arbor Networks
Barracuda Networks
Blue Coat
Box
CA Technologies
Check Point Software Technologies
Cisco Systems Inc.
Citrix
Dell
```

```
eEye
EMC
Extreme Networks
F5 Networks
F-Secure Corporation
Gemalto
HP
...
```

⚠ Query: `Source Address = 1.1.1.1 a`

Popup aligned to the beginning of `1.1.1.1`.

Expected suggestion:

```
and
```

⚠ Query: Source Username e

Popup aligned to the beginning of `Source`.

Expected suggestions:

```
equals
ends with
```

✔ Query: Device Vendor Ap

Popup aligned to the beginning of Ap.

Expected suggestions:

```
Apache
Application Security
NetApp Inc.
=
```

✔ Query: `Source Username = an`

Suggestions popup should not be visible.

✔ Query: `any ip e`

Popup aligned to the beginning of e.

Expected suggestions:

```
equals
```

`ends with`

⚠ Query: `source port in between 2 a`

Popup aligned to the beginning of 2.

Expected suggestion:

`and`

✅ Query: `www.google.com`

Popup aligned to the beginning of `www.google.com`.

Expected suggestions:

`Destination DNS Domain = www.google.com`

`Source DNS Domain = www.google.com`

`Request URL = www.google.com`

`any domain = www.google.com`

`any url = www.google.com`

`Agent DNS Domain = www.google.com`

`Destination NT Domain = www.google.com`

`Device DNS Domain = www.google.com`

`Device Domain = www.google.com`

`Device NT Domain = www.google.com`

`Source NT Domain = www.google.com`

✅ Query: `andrey`

Popup aligned to the beginning of `andrey`.

Expected suggestions:

`Source Username = andrey`

`Destination Username = andrey`

`any username = andrey`

`Source Hostname = andrey`

`Destination Hostname = andrey`

`any hostname = andrey`

`Agent Hostname = andrey`

`Device Hostname = andrey`

```
Destination DNS Domain = andrey
Source DNS Domain = andrey
any domain = andrey
Agent DNS Domain = andrey
Destination NT Domain = andrey
Device DNS Domain = andrey
Device Domain = andrey
Device NT Domain = andrey
Source NT Domain = andrey
```

Query: #

Popup aligned to the beginning of #.

Expected suggestions:

```
#Configuration changes
#Failed logins
#Failed logins for $username
#Firewall drop
#Firewall drop for $ip
#Malicious code activity
#SSH authentication
#VPN connections
#Windows account creation
```

Comma suggestions

Query: `Source Address , = 1.1.1.1`

Popup aligned to the beginning of `,`.

Expected suggestions:

```
Destination Address
Device Address
any ip
```

```
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

Query: `Device Vendor = Apache ,`

Popup aligned to the beginning of `,`.

Expected suggestions:

```
AirMagnet
Amazon
Apache
Application Security
Arbor Networks
Barracuda Networks
Blue Coat
Box
CA Technologies
Check Point Software Technologies
Cisco Systems Inc.
Citrix
Dell
eEye
EMC
Extreme Networks
F5 Networks
F-Secure Corporation
Gemalto
...
```

Logic suggestions

✅ Query: `Device Vendor = Apache and`

Popup aligned to the beginning of and.

Expected suggestion:

or

⚠️ Query: `Device Vendor = Apache and`

Popup aligned to the beginning of and.

Expected suggestions:

```
Source Address
Destination Address
Source Username
Destination Username
Source Hostname
Destination Hostname
Destination DNS Domain
Source DNS Domain
Device Address
Request URL
any category
any custom float
any domain
any hostname
any id
any ip
any ip6
any label
any mac
...
```

Operators suggestions

✅ Query: `Device Vendor =`

Popup aligned to the beginning of =.

Expected suggestions:

```
not equal
less than
greater than
less equal than
greater equal than
contains
does not contain
starts with
ends with
does not start with
does not end with
```

⚠️ Query: Device Vendor =

Popup aligned to the beginning of =.

Expected suggestions:

```
AirMagnet
Amazon
Apache
Application Security
Arbor Networks
Barracuda Networks
Blue Coat
Box
CA Technologies
Check Point Software Technologies
Cisco Systems Inc.
```

```
Citrix
Dell
eEye
EMC
Extreme Networks
F5 Networks
F-Secure Corporation
Gemalto
...
```

⚠️ Query: `www.google.com =`

Popup aligned to the beginning of =.

Expected suggestions:

```
Source Address
Destination Address
Source Username
Destination Username
Source Hostname
Destination Hostname
Destination DNS Domain
Source DNS Domain
Device Address
Request URL
any category
any custom float
any domain
any hostname
any id
any ip
any ip6
any label
any mac
```

...

Group suggestions

⚠ Query: any

Popup aligned to the beginning of any.

Expected suggestions:

```
category
domain
custom float
hostname
id
ip
ip6
label
mac
path
port
timestamp
uri
url
username
```

✅ Query: `any ip 1.1.1.1`

Popup aligned to the beginning of any.

Expected suggestions:

```
Source Address
Destination Address
Device Address
Agent Address
Agent Translated Address
Destination Translated Address
```

```
Device Translated Address
Source Translated Address
```

Query: `any = 1.1.1.1`

Popup aligned to the beginning of any.

Expected suggestions:

```
Source Address
Destination Address
Device Address
any ip
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

Groupname suggestions

Query: `any ip`

Popup aligned to the beginning of any.

Expected suggestions:

```
equals
not equal
less than
greater than
less equal than
greater equal than
contains
does not contain
starts with
ends with
does not start with
```

`does not end with`

`in between`

`is not between`

✅ Query: `any i`

Popup aligned to the beginning of `i`.

Expected suggestions:

`id`

`ip`

`ip6`

✅ Query: `any i = 1.1.1.1`

Popup aligned to the beginning of `i`.

Expected suggestions:

`id`

`ip`

`ip6`

❌ Query `any i 1.1.1.1`

Popup aligned to the beginning of `i`.

Expected suggestions:

`category`

`domain`

`custom float`

`hostname`

`id`

`ip`

`ip6`

`label`

`mac`

`path`

`port`

`timestamp`

```
uri
url
username
```

Query: `any ip google.com`

Popup aligned to the beginning of `ip`.

Expected suggestions:

```
=
Destination DNS Domain
Source DNS Domain
Request URL
any domain
any url
Agent DNS Domain
Destination NT Domain
Device DNS Domain
Device Domain
Device NT Domain
Source NT Domain
```

Query: `any ip = 1.1.1.1`

Popup aligned to the beginning of `ip`.

Expected suggestions:

```
Source Address
Destination Address
Device Address
Agent Address
Agent Translated Address
Destination Translated Address
Device Translated Address
Source Translated Address
```

Query: `any domaIn = 100`

Popup aligned to the beginning of `domaIn`.

Expected suggestion:

```
domain
```

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Query Quick Reference Guide (Investigate 1.01)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to arc-doc@hpe.com.

We appreciate your feedback!