Advanced Search Queries

The Carbon Black console provides a checkbox interface to choose criteria for searches of processes, binaries, alerts, and threat reports. This chapter describes how to construct more complex queries. The fields, datatypes, and examples here focus on queries to search for processes and binaries, but most of the syntax descriptions here also apply to alerts and threat reports.

Sections

Query Syntax Details

Carbon Black supports multiple types of operators and syntax that can be used to form more complex queries in the Search boxes on the Process Search, Binary Search, Threat Report Search, and Triage Alerts pages.

Searches are generally case-insensitive.

Terms, Phrases and Operators

A term is a single keyword (without whitespace) that is searched in the Carbon Black process or binary data store, or in the alerts or threat reports on your server. For example, a keyword could be: `foo`

Terms can be combined by logical operators and nested to form more complex queries, for example:

- and, AND, or whitespace: Boolean AND operator: `foo bar, foo and bar`
- or, OR: Boolean OR operator: `foo or bar`
- - : Boolean NOT operator: `-foo`
- nesting using parenthesis: `(foo or bar) baz`
- Wildcard searches with `*`, for example, `process_name:win*.exe`

Terms can be limited to a single field with <*field*>:<*term*> syntax, for example:

```
process_name:svchost.exe
```

Multiple terms are connected with `AND` if not otherwise specified.

Terms not preceded by fields are expanded to search *all* default fields.

Because terms are whitespace delimited, use double-quotes, or escape whitespaces with a single backslash, when required, for example:

```
path:"microsoft office\office15\powerpnt.exe"
```

or

```
path:microsoft\ office\office15\powerpnt.exe
```

Terms can be combined to form phrases. A phrase is a set of terms separated by whitespace and enclosed in quotes. Whitespace between the terms of a quoted phrase is not treated as a logical AND operator; rather a phrase is searched as a single term, for example: `"foo bar"`

Phrases can be combined and nested with other phrases and terms using logical operators, for example: `"foo bar" or baz`

Restrictions on Terms

Whitespace

Whitespace is the default delimiter. A query with whitespace would be “tokenized” and so parsed as multiple terms, for example:

Input: `microsoft office\office15\powerpnt.exe`

Is intepreted as two terms: `microsoft AND office\office15\powerpnt.exe`

You can use quotation marks to avoid automatic parsing into individual terms, for example:

Input: `"microsoft office\office15\powerpnt.exe"`

Is interpreted as: `microsoft office\office15\powerpnt.exe`

Alternatively, whitespaces can be escaped using the backslash (\):

Input: `microsoft\ office\office15\powerpnt.exe`

Is interpreted as: `microsoft office\office15\powerpnt.exe`

See “path” on page 13 for more information about how whitespaces and slashes affect path tokenization.

Parenthesis

Parentheses are used as a delimiter for nested queries. A query with parentheses is parsed as a nested query, and if a proper nesting cannot be found, a syntax error is returned, for example:

Input: `c:\program files (x86)\windows`

Is intepreted as: `c:\program AND files AND x86 AND \windows`

You can use quotation marks around the whole phrase to avoid automatic nesting, or you can escape the parentheses (and whitespaces) using the backslash (\) as the escape character. For example:

Input: `c:\program\ files\ \(x86\)\windows`

Is interpreted as: `c:\program files (x86)\windows`

Negative Sign

The negative sign is used as logical NOT operator. Queries that begin with a negative sign are negated in the submitted query, for example:

Input: `-system.exe`

Is interpreted as: `not system.exe`

Input: `-alliance_score_srstrust:*`

Is intrepreted as: *Return all results that are not trusted by the alliance.*

You can use a phrase query to avoid automatic negation, for example:

Input: `"-system.exe"`

Becomes: `"-system.exe"`

Double Quotes

Double quotes are used as a delimiter for phrase queries. A query in which double quotes should be taken literally must be escaped using backslash (\). For example, the following query input:

```
cmdline:"\"c:\program files \(x86\)\google\update\googleupdate.exe\" /svc"
```

would be interpreted to match the following command line (with the command line including the quotes as shown):

```
"c:\program files \(x86\)\google\update\googleupdate.exe\" /svc
```

Leading Wildcards

While not literally restricted, the use of leading wildcards in a query is not recommended unless absolutely necessary. Leading wildcards carry a significant performance penalty for the search. For example, the following query is *not* recommended:

```
filemod:*/system32/ntdll.dll
```

The same results would be returned by the following query, and the search would be much more efficient:

```
filemod:system32/ntdll.dll
```

Fields in Process and Binary Searches

This section contains a complete list of fields that are searchable in Carbon Black process and/or binary searches. Some fields are valid in only one of the two, some in both. Any binary-related field used in the process search searches the executable file backing the process.

If a query specifies a term without specifying a field, the search is executed on all default fields. Default fields are indicated by `(def)`.

Table 1: Fields in Carbon Black Process and Binary Searches

Field	Process Search	Binary Search	Field Type	Description
blocked_md5	x (def)	-	md5	MD5 of a process blocked due to a banning rule.
blocked_status	x	-	status	Status of a block attempt on a running process due to a banning rule, one of the following: a-ProcessTerminated b-NotTerminatedCBProcess c-NotTerminatedSystemProcess d-NotTerminatedCriticialSystemProcess e-NotTerminatedWhiltestedPath f-NotTerminatedOpenProcessError g-NotTerminatedTerminateError
childproc_count	x	-	count	Total count of child processes created by this process.

Table 1: Fields in Carbon Black Process and Binary Searches (continued)

Field	Process Search	Binary Search	Field Type	Description
childproc_md5	x (def)	-	md5	MD5 of the executable backing the created child processes.
childproc_name	x (def)	-	keyword	Filename of the child process executables.
cmdline	x (def)	-	cmdline	Full command line for this process.
comments	-	x (def)	text	Comment string from the class FileVersionInfo.
company_name	x	x (def)	text	Company name string from the class FileVersionInfo.
copied_mod_len	x	x	count	Number of bytes collected.
crossproc_count	x		count	Total count of cross process actions by an actor process.
crossproc_md5	x		md5	MD5 of an actor process that performed a cross process action on a target process.
crossproc_name	x		keyword	Name of an actor process that performed a cross process action on a target process.
crossproc_type	x (def)		remote_thread process_open	A remote thread or a process handle is open.
digsig_issuer	x	x (def)	text	If digitally signed, the issuer.
digsig_prog_name	x	x (def)	text	If digitally signed, the program name.
digsig_publisher	x	x (def)	text	If digitally signed, the publisher.
digsig_result	x	x (def)	sign	If digitally signed, the result. Values are: • “Bad Signature” • “Invalid Signature” • “Expired” • “Invalid Chain” • “Untrusted Root” • “Signed” • “Unsigned” • “Explicit Distrust”
digsig_sign_time	x	x	datetime	If digitally signed, the time of signing.
digsig_subject	x	x (def)	text	If digitally signed, the subject.

Table 1: Fields in Carbon Black Process and Binary Searches (continued)

Field	Process Search	Binary Search	Field Type	Description
domain	x (def)	-	domain	Network connection to this domain.
file_desc	x	x (def)	text	File description string from the class FileVersionInfo.
file_version	x	x (def)	text	File version string from the class FileVersionInfo.
filemod	x (def)	-	path	Path of a file modified by this process.
filemod_count	x	-	count	Total count of file modifications by this process.
filewrite_md5	x (def)	-	md5	MD5 of file written by this process.
group	x (def)	x (def)	keyword	Sensor group this sensor was assigned to at the time of process execution.
has_emet_config	x	-	bool	Values are True or False - Indicates whether process has EMET mitigations configured/ enabled.
has_emet_event	x	-	bool	Values are True or False - Indicates whether process has EMET mitigation events.
host_count	-	x	integer	Count of hosts that have seen a binary.
host_type	x (def)	-	keyword	Type of the computer: workstation, server, or domain controller.
hostname	x (def)	x (def)	keyword	Hostname of the computer on which the process was executed.
internal_name	x	x (def)	text	Internal name string from the class FileVersionInfo.
ipaddr	x	-	ipaddr	Network connection to or from this IP address.
ipport	x	-	integer	Network connection to this destination port.
is_64bit	x	x	bool	True if architecture is x64.
is_executable_image	x	x	bool	True if the binary is an EXE (versus DLL or SYS)
last_server_update	x	-	datetime	Last activity in this process in the server's local time.

Table 1: Fields in Carbon Black Process and Binary Searches (continued)

Field	Process Search	Binary Search	Field Type	Description
last_update	x	-	datetime	Last activity in this process in the computer's local time.
legal_copyright	x	x (def)	text	Legal copyright string from the class FileVersionInfo.
legal_trademark	x	x (def)	text	Legal trademark string from the class FileVersionInfo.
md5	x (def)	x (def)	md5	MD5 of the process, parent, child process, loaded module, or a written file.
modload	x (def)	-	path	Path of module loaded into this process.
modload_count	x	-	count	Total count of module loads by this process.
netconn_count	x	-	count	Total count of network connections by this process.
observed_filename	x	x (def)	path	Full path of the binary at the time of collection.
orig_mod_len	x	x	count	Size in bytes of the binary at time of collection.
original_filename	x	x (def)	text	Original name string from the class FileVersionInfo.
os_type	x	x	keyword	Type of the operating system: Windows, OSX or Linux.
parent_id	x	-	long	The internal Carbon Black process guid for the parent process.
parent_md5	x (def)	-	md5	MD5 of the executable backing the parent process.
parent_name	x (def)	-	keyword	Filename of the parent process executable.
path	x (def)	-	path	Full path to the executable backing this process.
private_build	x	x (def)	text	Private build string from the class FileVersionInfo.
process_id	x	-	long	The internal Carbon Black process guid for the process.
process_md5	x (def)	-	md5	MD5 of the executable backing this process.

Table 1: Fields in Carbon Black Process and Binary Searches (continued)

Field	Process Search	Binary Search	Field Type	Description
process_name	x (def)	-	keyword	Filename of the executable backing this process.
product_desc	x	x (def)	text	Product description string from the class FileVersionInfo.
product_name	x	x (def)	text	Product name string from the class FileVersionInfo.
product_version	x	x (def)	text	Product version string from the class FileVersionInfo.
regmod	x (def)	-	path	Path of a registry key modified by this process.
regmod_count	x	-	count	Total count of registry modifications by this process.
sensor_id	x	-	long	The internal Carbon Black sensor guid of the computer on which this process was executed.
server_added_ timestamp	-	x	datetime	The time this binary was first seen by the server.
special_build	x	x (def)	text	Special build string from the class FileVersionInfo.
start	x	-	datetime	Start time of this process in the computer's local time.
tampered	x	x	bool	Values are True or False - indicates if the event has been tampered with.
username	x (def)	-	keyword	User context with which the process was executed.
watchlist_<id>	x	x	datetime	The time that this process or binary matched the watchlist query with <id>

Fields in Alert and Threat Report Searches

Different sets of fields are searchable on the Triage Alerts and Threat Report Search pages. As with process and binary searches, if no field is specified for a term, the search is executed on all default fields. In the tables below, default fields are indicated by (def).

Table 2: Fields in Carbon Black Alert Page Searches

Field	Field Type	Description
alert_severity	float	Overall score of the alert (combines report score, feed rating, sensor criticality).
alert_type	keyword	Type of the alert: one of "watchlist.hit.ingress.binary", "wathclist.hit.ingress.process", "watchlist.hit.query.process", "watchlist.hit.query.binary", "watchlist.hit.ingress.host"
assigned_to	keyword (def)	Name of the Carbon Black administrator who triaged (changed the status) of the alert.
created_time	datetime	Creation time of the alert.
feed_id	int	Numeric value of the feed id (-1 for watchlists).
feed_name	keyword (def)	Name of the feed that triggered the alert. All user-created watchlists have the feed name "My Watchlists" as a special case.
group	keyword	Sensor group name of the endpoint on which the process/binary that triggered the alert was observed.
hostname	keyword (def)	Hostname of endpoint that the process/ binary that triggered the alert was observed on.
ioc_value	keyword (def)	Value (IP address or MD5) of the IOC that caused the alert to be triggered.
md5	md5 (def)	MD5 of the process that triggered the alert.
observed_filename	keyword (def)	Full path name of the process triggered the alert (not tokenized).
process_name	keyword (def)	Filename of the process that triggered the alert.
process_path	path (def)	Full path to the executable backing this process.
report_score	float	Report score of the feed that triggered the alert.
resolved_time	datetime	Time this alert was triaged by a resolution action.

Table 2: Fields in Carbon Black Alert Page Searches (continued)

Field	Field Type	Description
status	keyword	Status of the alert: one of "resolved", "unresolved", "in progress", "false positive".
username	keyword (def)	Username in whose context the process that triggered the alert event was executed.
watchlist_id	int (def)	Numeric value of the watchlist id (not applicable to feeds).
watchlist_name	keyword (def)	Name of the watchlist or the report name (for feeds).

Table 3: Fields in Carbon Black Threat Report Searches

Field	Field Type	Description
create_time	datetime	Date and time this feed report was created.
description	text (def)	Description of the feed report, whitespace tokenized so each term is individually searchable.
domain	domain (def)	A domain IOC value in the feed report.
feed_category	text (def)	Category of this report/feed, whitespace tokenized
feed_id	int	The numeric value of a feed.
feed_name	keyword (def)	The name a feed.
ipaddr	ipaddr	An ip address IOC value in the feed report.
is_ignored	bool	Indicates whether the report has been marked to be ignored on this server.
md5	md5 (def)	An MD5 IOC value in the feed report.
report_id	keyword	Name or unique identifier of the threat report that is part of the field.
tags	text (def)	Tags related to this report/feed, whitespace tokenized
title	text	Text title of the feed report, whitespace tokenized.
update_time	datetime	Date and time this feed report was last updated.

Datatypes

domain

Domains are split into labels for query purposes -- for example, “foo.com” is split into “foo” and “com”. If provided in a query, “dot” separator characters (.) between labels are maintained to enable position-dependent domain searches. This has the following results:

- *Leading dot after the label, no trailing dot* -- Returns results for matching labels that are at the *end* of the domain name.
- *Trailing dot after the label, no leading dot* -- Returns results for matching labels that are at the *beginning* of the domain name.
- *Leading and trailing dots surrounding the label* -- Returns results for matching labels that are in the middle of the domain name (i.e., not the first or last label).
- *Two labels with a dot between them* -- Treated as a search for the entire phrase, and so returns results for domains that include the entire string.
- *No dot separators* -- Returns results for any domain that includes the query string anywhere in the domain name.

The following table provides examples of these different domain searches:

Table 4: Use of Separators (Dots) in Domain Searches

Search	If domain is foo.com	If domain is foo.com.au
domain:com	**match**	**match**
domain:.com	**match**	no match
domain:.com.	no match	**match**
domain:com.	no match	no match
domain:foo.	**match**	**match**
domain:foo.com	**match**	no match

ipaddr

IP addresses are searched with CIDR notation:

`(ip)/(netmask)`

If the netmask is omitted, it is presumed to be 32, for example:

`ipaddr:192.168.0.0/16` or `ipaddr:10.0.1.1`

text

Text fields are tokenized on whitespace and punctuation. Searches are case-insensitive.

The string from the product_name field, for example:

`Microsoft Visual Studio 2010`

will be intpreted as `microsoft AND visual AND studio AND 2010`.

Searches for any one of these strings will match on the binary. Phrase queries for any two consecutive terms will also match on the binary, for example:

```
product_name: "visual studio"
```

count

An integer value. If it exists, values from 0 to `MAXINT`. Supports two types of search syntaxes:

- `X:` Matches all fields with precisely X, for example, `modload_count:34` for processes with exactly 34 modloads.
- `[X TO Y]:` Matches all fields with counts >= X and <= Y, for example, `modload_count:[1 TO 10]` for processes with 1 to 10 modloads.

In both cases, either X or Y can be replaced the wildcard `*`. for example,

`netconn_count:*` for any process where the `netconn_count` field exists.
`netconn_count:[10 TO *]` for any process with more than 10 network connections.

datetime

Datetime fields have five types of search syntaxes:

- `YYYY-MM-DD` matches all entries on this day, for example, `start:2013-12-01` for all processes started on Dec 1, 2013.
- `YYYY-MM-DDThh:mm:dd` matches all entries within the next 24 hours from this date and time, for example, `start:2013-12-01T22:15:00` for all processes started between Dec 1, 2013 at 22:15:00 to Dec 2, 2013 at 22:14:59.
- `[YYYY-MM-DD TO YYYY-MM-DD]` matches all entries between, for example, `start:[2013-12-01 TO 2013-12-31]` for all processes started in Dec 2013.
- `[YYYY-MM-DDThh:mm:ss TO YYYY-MM-DDThh:mm:ss]` matches all entries between, for example, `start:[2013-12-01T22:15:00 TO 2013-12-01:23:14:59]` for all processes started in Dec 1, 2013 within the given time frame.
- `-Xh` relative time calculations matches all entries with a time between `NOW-10h` and `NOW`. Support units supported are h: hours, m: minutes, s: seconds as observed on the host, for example, `start:-24h` for all processes started in the last 24 hours.

As with counts, YYYYMMDD can be replaced the wildcard `*`. for example, `start:[2013-01-01 TO *]` for any process started after 1 Jan 2013.

keyword

Keywords are `text` fields with no tokenization. The term that is searched for must exactly match the value in the field, for example, `process_name:svchost.exe`. Queries containing wildcards can be submitted with keyword queries, for example, `process_name:ms*.exe`

md5

MD5 fields are keyword fields with an md5 hash value. The term searched for must exactly match the value in the field, for example, `process_md5:6d7c8a951af6ad6835c029b3cb88d333`

path

Path fields are special text fields. They are tokenized according to path hierarchy, for example, `path:c:\windows`.

For a given path, all subpaths are tokenized. For example:

```
c:\windows\system32\boot\winload.exe
```

is tokenized as:

```
c:\windows\system32\boot\winload.exe
windows\system32\boot\winload.exe
system32\boot\winload.exe
boot\winload.exe
winload.exe
```

Note

As the example above shows, leading slashes (both forward and back) are *not* tokenized. The leading slash can and should be skipped when doing path-based queries. For example, you would search for `boot\winload.exe`, not `\boot\winload.exe`.

For queries involving path segments that are not tokenized, wildcard queries can be submitted, for example, `path:system*,` for any path that has `system` as sub-path in it.

bool

Boolean fields have only two possible values, the string `true` or `false`. Searches are case-insensitive.

sign

Signature fields can be one of the eight possible values: `Signed, Unsigned, Bad Signature, Invalid Signature, Expired,Invalid Chain, Untrusted Root, Explicit Distrust`. Values with whitespace must be enclosed in quotes, for example, `digsig_result:Signed` or `digsig_result:"Invalid Chain"`

cmdline

Command line strings that contain parenthesis or double quotes must be escaped using a backslash. If the string also contains whitespaces, enclose it in double quotes or use escape for the whitespaces, for example: `cmdline:"\"c:\program files\(x86\)\google\update\googleupdate.exe\" /svc"` or

```
cmdline:\"c:\program\ files\
\(x86\)\google\update\googleupdate.exe\"\ /svc
```

Searching with Multiple (Bulk) Criteria

You can search for multiple Incidents of Compromise (IOCs) by using bulk search criteria in both the process search and binary search pages. Carbon Black determines if the search string contains MD5 hash values or IP addresses. If the criteria consists of MD5 hash values, Carbon Black automatically sets these search parameters to `md5:<value>`. If the criteria is an IP address, Carbon Black sets the search parameter to ip_address:<*value*>. If the criteria do not match either of these types, the search is treated as a generic text search and uses the criteria to search on any field, such as a filename, a host name, a registry file, or a network connection (DNS name).

Note

If you are using multiple MD5 hash values for search criteria to create a watchlist, you must enclose the values in parentheses ( ). For example:

```
(md5:45cc061d9581e52f008e90e81da2cfd9
md5:829e4805b0e12b383ee09abdc9e2dc3c
md5:ac9fa2ba34225342a8897930503ae12f
md5:5f7eaaf5d10e2a715d5e305ac992b2a7)
```

If you do not enclose the list in brackets, the only value that is tagged for the watchlist is the last value in the list.

Searching with Binary Joins

Some binary search fields can be used as part of a process search query. (See Table 1, “Fields in Carbon Black Process and Binary Searches”, on page 4, for more information.) In this case, the results returned are process instances backed by binaries that match the binary search criteria. This is called a joined searched. For example, consider submitting the following query on the process search page:

```
digsig_result:Unsigned
```

This query returns all process instances backed by an MD5 that is unsigned.

By default, join searches are performed against the MD5 of the standalone process executable (`process_md5`). However, joined searches can also be performed against the MD5 of the following related events:

- filewrites
- parent processes
- child processes
- modloads

Specify the search by appending the following suffixes to the end of the binary search field: filewrite, parent, child and modload. For example:

```
digsig_result_modload:Unsigned
```

This query returns all process instances that have loaded an unsigned module.

Example Searches

Process Search Examples

Table 5: Process Search Query String Examples

Example Query Strings	Result
domain:www.carbonblack.com	Returns all processes with network connections to or from domains matching the given FQDN.
domain:.com	Returns all processes with network connections to or from domains matching `*.com`
domain:.com.	Returns all processes with network connections to or from domains matching the form `*.com.*`
domain:www.	Returns all processes with network connections to or from domains matching the form `www.*`
domain:microsoft	Returns all processes with network connections to or from domains matching `*.microsoft OR *.microsoft.* OR microsoft.*`
ipaddr:127.0.0.1	Returns all processes with network connections to or from IP address `127.0.0.1`
ipaddr:192.168.1.0/24	Returns all processes with network connections to or from IP addresses in the network subnet `192.168.1.0/24`
modload:kernel32.dll	Returns all processes that loaded a module `kernel32.dll` (accepts path hierarchies).
modload:c:\windows\system32\sxs.dll	Returns all processes that loaded a module matching path and file `sxs.dll` (accepts path hierarchies).
path:c:\windows\system32\notepad.exe	Also returns all processes with the matching path (accepts path hierarchies).
regmod:\registry\machine\system\controlset001\control\deviceclasses*	Returns all processes that modified a registry entry with the matching path (accepts path hierarchies).
path:excel.exe	Returns all processes with the matching path (accepts path hierarchies).
cmdline:backup	Returns all processes with matching command line arguments.
hostname:win-5ikqdnf9go1	Returns all processes executed on the host with matching hostname.
group:"default group"	Returns all processes executed on hosts with matching group name (use of quotes are required when submitting two-word group names).
host_type:workstation	Returns all processes executed on hosts with matching type (use of quotes are required when submitting two-word host types).

Table 5: Process Search Query String Examples (continued)

Example Query Strings	Result
username:system	Returns all processes executed with the matching user context.
process_name:java.exe	Returns all processes with matching names.
parent_name:explorer.exe	Returns all processes executed by a parent process with matching names.
childproc_name:cmd.exe	Returns all processes that executed a child process with matching names.
md5:5a18f00ab9330ac7539675f3f326cf11	Returns all processes, modified files, or loaded modules with matching MD5 hash values.
process_md5:5a18f00ab9330ac7539675f3f326cf11	Returns all processes with matching MD5 hash values.
parent_md5:5a18f00ab9330ac7539675f3f326cf11	Returns all processes that have a parent process with the given MD5 hash value.
filewrite_md5:5a18f00ab9330ac7539675f3f326cf11	Returns all processes that modified a file or module with matching MD5 hash values.
childproc_md5:5a18f00ab9330ac7539675f3f326cf11	Returns all processes that executed a child process with matching MD5 hash values.
<type>_count:*	Returns all processes that have xxx_count field > 0, where type is one of modload, filemod, regmod, netconn, or childproc.
<type>_count:10	Returns all processes that have xxx_count field = 10, where type is one of modload, filemod, regmod, netconn, or childproc.
<type>_count:[10 TO 20]	Returns all processes that have xxx_count field >= 10 and <= 20, where type is one of modload, filemod, regmod, netconn, or childproc.
<type>_count:[10 TO *]	Returns all processes that have xxx_count field >= 10, where type is one of modload, filemod, regmod, netconn, or childproc.
<type>_count:[* TO 10]	Returns all processes that have xxx_count field < 10, where type is one of modload, filemod, regmod, netconn, or childproc.
start:2011-12-31	Returns all processes with a start date of 2011-12-31 (as observed on the host).
start:[* TO 2011-12-31]	Returns all processes with a start date earlier than or equal to 2011-12-31 (as observed on the host).
start:[* TO 2011-12-31T22:15:00]	Returns all processes with a start date earlier than or equal to 2011-12-31 at 22:15:00 (as observed on the host).

Table 5: Process Search Query String Examples (continued)

Example Query Strings	Result
start:[2011-12-31 TO *]	Returns all processes with a start date later than or equal to 2011-12-31 (as observed on the host).
start:[2011-12-31T09:45:00 TO *]	Returns all processes with a start date later than or equal to 2011-12-31 at 09:45:00 (as observed on the host).
start:*	Returns processes with any start date (as observed on the host).
start:[* TO *]	Returns processes with any start date (as observed on the host).
start:-10h	Returns all processes with a start time between NOW-10h and NOW. Units supported are, h: hours, m: minutes, s: seconds (as observed on the host).
last_update:2011-12-31	Returns all processes last updated on date 2011-12-31 (as observed on the host).
last_update:[* TO 2011-12-31]	Returns all processes last updated on a date earlier than or equal to 2011-12-31 (as observed on the host).
last_update:[* TO 2011-12-31T22:15:00]	Returns all processes last updated on a date earlier than or equal to 2011-12-31 at 22:15:00 (as observed on the host).
last_update:[2011-12-31 TO *]	Returns all processes last updated on a date later than or equal to 2011-12-31 (as observed on the host).
last_server_update:[2011-12-31T09:45:00 TO *]	Returns all processes last updated on a date later than or equal to 2011-12-31 at 09:45:00 (as observed at the server).
last_server_update:*	Returns processes with any update date (as observed on the server).
last_server_update:[* TO *]	Returns processes with any update date (as observed on the server) within the range provided.
last_server_update:-10h	Returns all processes last updated between NOW-10h and NOW. Units supported are h: hours, m: minutes, s: seconds (as observed on the server).
process_id:<guid>	Returns the process with the given process id, where `<guid>` is a signed 64-bit integer.
parent_id:<guid>	Returns the process with the given parent process id, where `<guid>` is a signed 64-bit integer.
sensor_id:<guid>	Returns processes executed on host with given sensor id, where `<guid>` is an unsigned 64-bit integer.

Binary Search Examples

Table 6: Binary Search Query String Examples

Example Query Strings	Result
md5:5a18f00ab9330ac7539675f326cf11	Returns all binaries with matching MD5 hash values.
digsig_publisher:Oracle	Returns all binaries with a digital signature publisher field with a matching name.
digsig_issues:VeriSign	Returns all binaries with a digital signature issuer field with a matching name.
digsig_subject:Oracle	Returns all binaries with a digital signature subject field with a matching name.
digsig_prog_name:Java	Returns all binaries with a digital signature program name field with a matching name.
digsig_result:Expired	Returns all binaries with a digital signature status of `<status>`.
digsig_sign_time:2011-12-31	Returns all binaries with a digital signature date of 2011-12-31.
digsig_sign_time:[* TO 2011-12-31]	Returns all binaries with a digital signature date earlier than or equal to 2011-12-31.
digsig_sign_time:[2011-12-31 TO *]	Returns all binaries with a digital signature date later than or equal to 2011-12-31.
digsig_sign_time:*	Returns binaries with any digital signature date.
digsig_sign_time:[* TO *]	Returns binaries with any digital signature date within the range provided.
digsig_sign_time:-10h	Returns all binaries with a start time between NOW-10h and NOW. Units supported are h: hours, m: minutes, s: seconds.
<type>_version:7.0.170.2	Returns all binaries with matching version, where `<type>` is product or file.
product_name:Java	Returns all binaries with matching product name.
company_name:Oracle	Returns all binaries with matching company name.
internal_name:java	Returns all binaries with matching internal name.
original_filename:mtxoci.dll	Returns all binaries with matching filename.
observed_filename:c:\windows\system32\mtxoci.dll	Returns all binaries that have been observed to run on or were loaded with the given path.
<type>_mod_len:[* TO 10]	Returns all binaries that have `<type>_mod_len` (module length in bytes) field < 4096, where type is original or copied.

Table 6: Binary Search Query String Examples (continued)

Example Query Strings	Result
<type>_desc:“database support”	Returns all binaries that have `<type>_desc` field with matching text, where type is file or product.
legal_<type>:Microsoft	Returns all binaries with matching `legal_<type>` field text, where type is trademark or copyright.
<type>_build:“Public version”	Returns all binaries with matching `<type>_build` field text, where type is special or private.
is_executable_image:True or False	Boolean search (case insensitive) returning all binaries that are executable or not executable.
is_64bit_:True or False	Boolean search (case insensitive) returning all binaries that are 64-bit or not 64-bit.
watchlist_4:[2014-04-01 TO 2014-09-31]	Returns all binaries that matched watchlist 4 during the time period shown.

Threat Intelligence (Alliance) Search Examples

Any document matching a threat intelligence feed is tagged with an alliance_score_<feed> field, where the value is a score from -100 to 100. <feed> is the “short name” of the threat intelligence feed, such as “nvd”, “isight, or “virustotal.” For any threat intelligence feed, you can click the **View Hits** button to discover the feed’s short name.

Table 7: Threat Intelligence Search Examples

Example Query Strings	Result
alliance_score_<feed>:*	Returns all binaries that have `<feed>` score > 0.
alliance_score_<feed>:10	Returns all binaries that have `<feed>` score = 10.
alliance_score_<feed>:[10 TO 20]	Returns all binaries that have `<feed>` score >= 10 and <= 20.
alliance_score_<feed>:[10 TO *]	Returns all binaries that have `<feed>` score >= 10.
alliance_score_<feed>:[* TO 10]	Returns all binaries that have `<feed>` score < 10.