Seq 2021 Cheat Sheet

INSTALLATION

Windows
Download the latest MSI from datalust.co/download.
Also available on Chocolatey and winget.

Docker
For Linux or Mac users, use the Docker command:

```
$ docker run --name seq -d --restart
unless-stopped -e ACCEPT_EULA=Y -p
5341:80 datalust/seq:latest
```

For more examples, see hub.docker.com/r/datalust/seq

Kubernetes
Helm instructions at docs.datalust.co/docs/using-helm.

BROWSE

The Seq UI is served at http://localhost:5341 by default.

NAVIGATION

Events
Search and query logs. Filter events by selecting signals. Create signals from search expressions to index all matching events, for better search performance.

Dashboards
Create and organize charts based on queries and signals.

Data > Ingestion
View the volume of logs sent to Seq in the last 24 hours. Create and manage API keys and input apps.

Data > Storage
View disk usage, check if retention policies are working as expected, add new retention policies.

Settings
Enabled authentication. Manage Seq configuration. Install and manage Seq apps. Add users (subscription required).

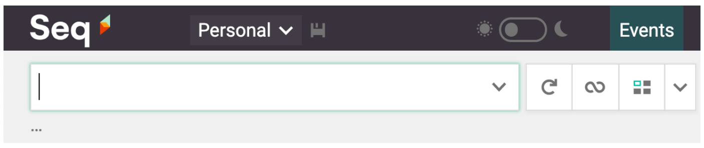
User Settings
Create and manage personal API keys, default workspace, and more.

SEARCH EXPRESSION OR QUERY?

There are two ways to explore log data in Seq:

- **Search expressions** for filtering and signals
- **Queries** for analyzing and dashboarding

Run search expressions and queries in the same "search box", by pressing Enter, or clicking .



TEXT FRAGMENTS

Text fragments use **"double-quotes"**, and are a shortcut to searching for logs containing the text.

AS FREE TEXT

```
New user created
```

See note below about no double-quotes

WITH LOGICAL OPERATORS

```
"operation" and not "timed out"
```

Double-quotes required when operators are used

ESCAPING

```
"New user \"Lucy\" created"
```

The text fragment `"prod"` is equivalent to the expression:

```
@Message like '%prod%' ci or @Exception like '%prod%' ci
```

Note: If there are **no double-quotes** around an input, Seq will try to determine if it is a valid search expression before treating it as free text.

Text λ You can click on the small *text* or *lambda* icon that appears below the filter bar, to inspect your input.

KEYWORDS

Keywords are **not** case-sensitive.

```
ci

and  or  not

for in

like

true  false

null

if  then  else

select  as  from  stream  where  group
by  having  order  asc  desc  limit
refresh  time  window
```

Machine data, for humans.

Seq 2021 Cheat Sheet

STRINGS

Unlike text fragments, string literals use **'single-quotes'** and are **case-sensitive**.

EXACT MATCH

```
Email = 'seq@example.com'
```

Single-quotes required for strings

CASE-INSENSITIVE

```
Environment = 'PRODUCTION' ci
```

LIKE OPERATOR

```
Environment like 'Prod%'
```

Single-character wildcard, underscore (_), also supported

REGULAR EXPRESSION

```
Source = /System.(Web|Api)/
```

ESCAPING

```
@Exception like '%can''t find host%'
```

Use a single-quote to escape a single-quote, %% escapes % in like expressions

STRING FUNCTIONS

```
=  <>  like                          COMPARATORS

Length(text)

ToLower(text)

ToUpper(text)

IndexOf(text, pattern)

StartsWith(text, pattern)

EndsWith(text, pattern)

Contains(text, pattern)

Substring(text, startIndex, length)
```

`pattern` can be a string or regex
Pass `length` as null to capture to end-of-string

NUMBERS

Integers, decimals (floats), and hexadecimal are all represented as 128-bit decimal values

EQUALITY

```
StatusCode = 200
```

COMPARISON

```
ElapsedMilliseconds > 1500
```

CONVERT STRING TO OBJECT

```
StatusCode = ToNumber('401')
```

NUMBER FUNCTIONS

```
=  <>  >  >=  <  <=                  COMPARATORS

+ - * / ^ %                          ARITHMETIC OPERATORS

Round(num, places)

ToNumber(text)
```

NULL

Like JavaScript, Seq treats null as a value which you can compare using "=", "<>", "in", and other operators. By contrast, the "is null" operator (inherited from SQL) checks for existence.

CHECK IF A PROPERTY EXISTS

```
OrderId is not null
```

Has(OrderId) is also valid

CHECK IF A PROPERTY EXISTS AND HAS THE VALUE NULL

```
OrderId = null
```

CONDITIONAL IF/THEN/ELSE

```
if Quantity = 0 then 'None' else 'Some'
```

BOOLEAN AND NULL FUNCTIONS

```
if {expr} then {x} else {y}          IF/THEN/ELSE

is null/is not null                  CHECK FOR EXISTENCE

Coalesce(first, second)              IF FIRST IS NULL, RETURN SECOND
```

Machine data, for humans.

Seq 2021 Cheat Sheet

DATE AND TIME

Seq uses **ticks** — total number of 100 nanosecond intervals since 1 Jan, 0001 — as its time representation.

Note: ticks are **not** the same as milliseconds since Epoch.

There are 10,000 ticks in 1 millisecond.

GET EVENTS AFTER 2PM, 31 MAR 2020 GMT+10

```
@Timestamp > DateTime('2020-03-31
14:00:00 +10')
```

Most valid date and time string formats supported

Seq supports **duration literals** like `1d` or `30m` as a shorthand way of writing a duration in ticks.

GET EVENTS IN THE LAST DAY

```
@Timestamp >= Now() - 1d
```

GET EVENTS BEFORE 11AM LOCAL TIME

```
TimeOfDay(@Timestamp, 10) < 11h
```

DATE AND TIME FUNCTIONS

`Now()`	GET CURRENT TIME IN TICKS
`DateTime(text)`	CONVERT DATETIME STRING TO TICKS
`ToIsoString(ticks)`	CONVERT TICKS TO ISO-8601 STRING
`TimeOfDay(ticks, offset)`	CONVERT TICKS TO TIME OF DAY
`TimeSpan(timespan)`	CONVERT TIMESPAN STRING TO TICKS
`TotalMilliseconds(ticks)`	CONVERT TICKS TO MS

DURATION UNITS

`d h m s ms us`	DURATION UNITS

"us" is microseconds

COLLECTIONS (ARRAYS AND OBJECTS)

In Seq, collections include **arrays** and **objects**.

OBJECT SUB-PROPERTIES

```
User.Email = 'seq@example.com'
```

User['Email'] is also valid

MATCH AT ANY INDEX

```
Products[?].Name = 'Coffee'
```

Find events where at least one of the Products is 'Coffee'

MATCH ALL WILDCARD

```
Post.Tags[*] in ['Seq', '2021']
```

Find events where all Tags are either 'Seq', or '2021'

MATCH ALL WILDCARD, CHAINED

```
Order.Shipments[*].Items[*].Tax > 0
```

"An Order property where all items in all shipments were taxable."

"IN" OPERATOR

```
@Level in ['Error', 'Warning']
```

COLLECTION FUNCTIONS

`.  [?]  [*]  in`	ACCESSOR, WILDCARDS AND "IN" OPERATOR
`Keys(obj)`	RETURN ARRAY OF KEYS IN AN OBJECT
`Values(obj)`	RETURN ARRAY OF VALUES IN AN OBJECT
`FromJson(string)`	CONVERT STRING TO ARRAY OR OBJECT
`ToJson(obj)`	CONVERT ANY VALUE TO A STRING
`ElementAt(col, accessor)`	ELEMENT AT KEY OR INDEX

DEBUGGING OR TESTING EXPRESSIONS

Sometimes, we need to check if expressions are doing as we expect.

Debug an expression by wrapping it in `select <expr> from stream limit 1`

TESTING "1+1" EXPRESSION OUTPUT

```
select 1 + 1 from stream limit 1
```

Outputs 2

Seq 2021 Cheat Sheet

QUERIES

Seq uses SQL-like query syntax.

QUERY SYNTAX

```
select [<column> [as <label>],]
from stream
     [where <predicate>]
     [group by [<grouping>|time(<d>),]]
     [having <predicate>]
     [order by [time|<label>] [asc|desc]]
     [limit <n>]
     [for refresh]
```

"for refresh" is for bypassing the query cache, only use if data is stale.

EXAMPLE QUERY

```
select count(*) as count
from stream
where StatusCode > 399
group by RequestPath
order by count desc
limit 10
```

Find the top 10 RequestPaths that have returned a StatusCode of 400 or above

TIME SLICE QUERIES

Seq can generate timeseries data using the special `time()` grouping, in conjunction with a **duration literal** (1d, 30m) that determines the time interval to use.

COUNT BY TIME SLICE

```
select count(*)
from stream
group by time(15m)
```

COUNT BY TIME SLICE, GROUPED BY PROPERTY

```
select count(*)
from stream
group by Environment, time(1h)
```

USING THE INTERVAL() FUNCTION

```
select count(*)/interval() as
RateOfOrders
from stream
where @Properties['Order'] is not null
group by time(5m)
limit 1000
```

DURATION FUNCTIONS

`interval()` RETURNS THE CURRENT TIME GROUPING DURATION

When used in a query with group by time(x), returns x

Tip: Chart your query results as line, bar, and pie charts directly in the events screen.

AGGREGATE FUNCTIONS

Aggregate functions perform a calculation on a set of values and return a single value.

`any()` `all()`

```
select any(@Level = 'Error') from stream
```

Return true if expression is true

`count()`

```
select count(*) from stream
```

Counts all events

```
select count(IsAdmin) from stream
```

Counts number of events that contain IsAdmin = true.

`distinct()`

```
select distinct(@Level) from stream
```

List all distinct levels

```
select count(distinct(@EventType)) fr...
```

Count all distinct event types

`min()` `max()` `sum()` `mean()` `percentile()`

```
select min(Elapsed), max(Elapsed) fr...
```

Get smallest and largest Elapsed

```
select min(Elapsed), max(Elapsed),
mean(Elapsed), percentile(Elapsed, 90)
from stream group by time(1h)
```

Plot aggregates over time

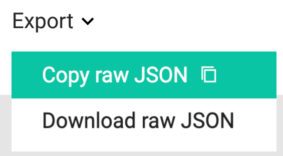
`first()` `last()`

```
select first(ExceptionType) from stream
where Application <> 'Admissions'
```

Machine data, for humans.

SEQ EVENT ANATOMY, BUILT-IN PROPERTIES, AND PROPERTIES

Seq uses Serilog's Compact Log Event Format (CLEF) as its native JSON event format. If you export any event from Seq, it will look something like the below:



```
{
  "@t":"2021-01-26T22:37:02.6297213Z",
  "@mt":"HTTP {RequestMethod} {RequestPath} responded {StatusCode}",
  "@m":"HTTP GET /example?q=123 responded 501",
  "@i":123456,
  "@l":"Error",
  "@x":"System.Threading.Tasks.TaskCancelledException\nAt line...",
  "RequestMethod":"GET",
  "RequestPath":"/example?q=123",
  "StatusCode":501,
  "Elapsed":0.75233,
  "RequestId":"0AB12C3DE4FGH:00000001",
  "Application":"MyExampleApp",
  "Site":"Production"
}
```

Here is a table that outlines the relationship between CLEF properties, and **built-in properties**. You can use built-in properties in search expressions and queries.

Note: All property names are case-sensitive.

	CLEF	Built-in property	Extra notes
Timestamp	`@t`	`@Timestamp`	Timestamp in ticks
Message	`@m`	`@Message`	
Event Type	`@i`	`@EventType`	Maybe a number or hexadecimal string
Event Id		`@Id`	e.g. 'event-313db38ac31d...'
Level	`@l`	`@Level`	
Exception	`@x`	`@Exception`	Usually a stack trace
Arrival		`@Arrived`	Number representing order of arrival
Properties		`@Properties`	All event properties without '@' prefix
Message Template	`@mt`	`@MessageTemplate`	
Data		`@Data`	Event as JSON object
Document		`@Document`	Event as JSON string

More details on CLEF at https://github.com/serilog/serilog-formatting-compact