RE2 syntax

Article • 04/11/2023

RE2 regular expression syntax describes the syntax of the regular expression library used by Kusto (re2). There are a few functions in Kusto that perform string matching, selection, and extraction by using a regular expression

- countof()
- extract()
- extract_all()
- matches regex
- parse operator
- replace_regex()
- trim()
- trimend()
- trimstart()

The regular expression syntax supported by Kusto is that of the re2 library. These expressions must be encoded in Kusto as `string` literals, and all of Kusto's string quoting rules apply. For example, the regular expression `\A` matches the beginning of a line, and is specified in Kusto as the string literal `"\\A"` (note the "extra" backslash (`\`) character).

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

RE2 library

Article • 05/01/2023

For information on the use of regular expressions with Kusto Query Language (KQL), see RE2 syntax.

Regular expressions are a notation for describing sets of character strings. When a particular string is in the set described by a regular expression, we often say that the regular expression matches the string.

The simplest regular expression is a single literal character. Except for the metacharacters like *+?()|, characters match themselves. To match a metacharacter, escape it with a backslash: \\+ matches a literal plus character.

Two regular expressions can be alternated or concatenated to form a new regular expression: if e1 matches s and e2 matches t, then e1|e2 matches s or t, and e1e2 matches st.

The metacharacters *, +, and ? are repetition operators: e1* matches a sequence of zero or more (possibly different) strings, each of which match e1; e1+ matches one or more; e1? matches zero or one.

The operator precedence, from weakest to strongest binding, is first alternation, then concatenation, and finally the repetition operators. Explicit parentheses can be used to force different meanings, just as in arithmetic expressions. Some examples: ab|cd is equivalent to (ab)|(cd); ab* is equivalent to a(b*).

The syntax described so far is most of the traditional Unix egrep regular expression syntax. This subset suffices to describe all regular languages: loosely speaking, a regular language is a set of strings that can be matched in a single pass through the text using only a fixed amount of memory. Newer regular expression facilities (notably Perl and those that have copied it) have added many new operators and escape sequences, which make the regular expressions more concise, and sometimes more cryptic, but usually not more powerful.

This page lists the regular expression syntax accepted by RE2.

It also lists some syntax accepted by PCRE, PERL, and VIM.

Single-character expressions

Example	Description

Example	Description
`.`	any character, possibly including newline (s=true)
`[xyz]`	character class
`[^xyz]`	negated character class
`\d`	Perl character class
`\D`	negated Perl character class
`[[:alpha:]]`	ASCII character class
`[[:^alpha:]]`	negated ASCII character class
`\pN`	Unicode character class (one-letter name)
`\p{Greek}`	Unicode character class
`\PN`	negated Unicode character class (one-letter name)
`\P{Greek}`	negated Unicode character class

Composites

Example	Description
`xy`	`x` followed by `y`
`x|y`	`x` or `y` (prefer `x`)

Repetitions

Example	Description
`x*`	zero or more `x`, prefer more
`x+`	one or more `x`, prefer more
`x?`	zero or one `x`, prefer one
`x{n,m}`	`n` or `n`+1 or ... or `m` `x`, prefer more
`x{n,}`	`n` or more `x`, prefer more
`x{n}`	exactly `n` `x`

Example	Description
`x*?`	zero or more `x`, prefer fewer
`x+?`	one or more `x`, prefer fewer
`x??`	zero or one `x`, prefer zero
`x{n,m}?`	`n` or `n`+1 or ... or `m` `x`, prefer fewer
`x{n,}?`	`n` or more `x`, prefer fewer
`x{n}?`	exactly `n` `x`
`x{}`	(≡ `x*`) (NOT SUPPORTED) VIM
`x{-}`	(≡ `x*?`) (NOT SUPPORTED) VIM
`x{-n}`	(≡ `x{n}?`) (NOT SUPPORTED) VIM
`x=`	(≡ `x?`) (NOT SUPPORTED) VIM

Implementation restriction: The counting forms x{n,m}, x{n,}, and x{n} reject forms that create a minimum or maximum repetition count above 1000. Unlimited repetitions aren't subject to this restriction.

Possessive repetitions

Example	Description
`x*+`	zero or more `x`, possessive (NOT SUPPORTED)
`x++`	one or more `x`, possessive (NOT SUPPORTED)
`x?+`	zero or one `x`, possessive (NOT SUPPORTED)
`x{n,m}+`	`n` or ... or `m` `x`, possessive (NOT SUPPORTED)
`x{n,}+`	`n` or more `x`, possessive (NOT SUPPORTED)
`x{n}+`	exactly `n` `x`, possessive (NOT SUPPORTED)

Grouping

Example	Description
`(re)`	numbered capturing group (submatch)

Example	Description
`(?P<name>re)`	named & numbered capturing group (submatch)
`(?<name>re)`	named & numbered capturing group (submatch) (NOT SUPPORTED)
`(?'name're)`	named & numbered capturing group (submatch) (NOT SUPPORTED)
`(?:re)`	non-capturing group
`(?flags)`	set flags within current group; non-capturing
`(?flags:re)`	set flags during re; non-capturing
`(?#text)`	comment (NOT SUPPORTED)
`(?|x|y|z)`	branch numbering reset (NOT SUPPORTED)
`(?>re)`	possessive match of `re` (NOT SUPPORTED)
`re@>`	possessive match of `re` (NOT SUPPORTED) VIM
`%(re)`	non-capturing group (NOT SUPPORTED) VIM

Flags

Example	Description
`i`	case-insensitive (default false)
`m`	multi-line mode: `^` and `$` match begin/end line in addition to begin/end text (default false)
`s`	let `.` match `\n` (default false)
`U`	ungreedy: swap meaning of `x*` and `x*?`, `x+` and `x+?`, etc (default false)

Flag syntax is xyz (set) or -xyz (clear) or xy-z (set xy, clear z).

Empty strings

Example	Description
`^`	at beginning of text or line (`m`=true)
`$`	at end of text (like `\z` not `\Z`) or line (`m`=true)

Example	Description
`\A`	at beginning of text
`\b`	at ASCII word boundary (`\w` on one side and `\W`, `\A`, or `\z` on the other)
`\B`	not at ASCII word boundary
`\g`	at beginning of subtext being searched (NOT SUPPORTED) PCRE
`\G`	at end of last match (NOT SUPPORTED) PERL
`\Z`	at end of text, or before newline at end of text (NOT SUPPORTED)
`\z`	at end of text
`(?=re)`	before text matching `re` (NOT SUPPORTED)
`(?!re)`	before text not matching `re` (NOT SUPPORTED)
`(?<=re)`	after text matching `re` (NOT SUPPORTED)
`(?<!re)`	after text not matching `re` (NOT SUPPORTED)
`re&`	before text matching `re` (NOT SUPPORTED) VIM
`re@=`	before text matching `re` (NOT SUPPORTED) VIM
`re@!`	before text not matching `re` (NOT SUPPORTED) VIM
`re@<=`	after text matching `re` (NOT SUPPORTED) VIM
`re@<!`	after text not matching `re` (NOT SUPPORTED) VIM
`\zs`	sets start of match (= \K) (NOT SUPPORTED) VIM
`\ze`	sets end of match (NOT SUPPORTED) VIM
`\%^`	beginning of file (NOT SUPPORTED) VIM
`\%$`	end of file (NOT SUPPORTED) VIM
`\%V`	on screen (NOT SUPPORTED) VIM
`\%#`	cursor position (NOT SUPPORTED) VIM
`\%'m`	mark `m` position (NOT SUPPORTED) VIM
`\%231`	in line 23 (NOT SUPPORTED) VIM
`\%23c`	in column 23 (NOT SUPPORTED) VIM

Example	Description
`\%23v`	in virtual column 23 (NOT SUPPORTED) VIM

Escape sequences

Example	Description
`\a`	bell (≡ `\007`)
`\f`	form feed (≡ `\014`)
`\t`	horizontal tab (≡ `\011`)
`\n`	newline (≡ `\012`)
`\r`	carriage return (≡ `\015`)
`\v`	vertical tab character (≡ `\013`)
`\*`	literal `*`, for any punctuation character `*`
`\123`	octal character code (up to three digits)
`\x7F`	hex character code (exactly two digits)
`\x{10FFFF}`	hex character code
`\C`	match a single byte even in UTF-8 mode
`\Q...\E`	literal text `...` even if `...` has punctuation
`\1`	backreference (NOT SUPPORTED)
`\b`	backspace (NOT SUPPORTED) (use `\010`)
`\cK`	control char ^K (NOT SUPPORTED) (use `\001` etc)
`\e`	escape (NOT SUPPORTED) (use `\033`)
`\g1`	backreference (NOT SUPPORTED)
`\g{1}`	backreference (NOT SUPPORTED)
`\g{+1}`	backreference (NOT SUPPORTED)
`\g{-1}`	backreference (NOT SUPPORTED)
`\g{name}`	named backreference (NOT SUPPORTED)

Example	Description
`\g<name>`	subroutine call (NOT SUPPORTED)
`\g'name'`	subroutine call (NOT SUPPORTED)
`\k<name>`	named backreference (NOT SUPPORTED)
`\k'name'`	named backreference (NOT SUPPORTED)
`\lX`	lowercase `X` (NOT SUPPORTED)
`\ux`	uppercase `x` (NOT SUPPORTED)
`\L...\E`	lowercase text `...` (NOT SUPPORTED)
`\K`	reset beginning of `$0` (NOT SUPPORTED)
`\N{name}`	named Unicode character (NOT SUPPORTED)
`\R`	line break (NOT SUPPORTED)
`\U...\E`	upper case text `...` (NOT SUPPORTED)
`\X`	extended Unicode sequence (NOT SUPPORTED)
`\%d123`	decimal character 123 (NOT SUPPORTED) VIM
`\%xFF`	hex character FF (NOT SUPPORTED) VIM
`\%o123`	octal character 123 (NOT SUPPORTED) VIM
`\%u1234`	Unicode character 0x1234 (NOT SUPPORTED) VIM
`\%U12345678`	Unicode character 0x12345678 (NOT SUPPORTED) VIM

Character class elements

Example	Description
`x`	single character
`A-Z`	character range (inclusive)
`\d`	Perl character class
`[:foo:]`	ASCII character class `foo`
`\p{Foo}`	Unicode character class `Foo`

Example	Description
`\pF`	Unicode character class `F` (one-letter name)

Named character classes as character class elements

Example	Description
`[\d]`	digits (≡ `\d`)
`[^\d]`	not digits (≡ `\D`)
`[\D]`	not digits (≡ `\D`)
`[^\D]`	not not digits (≡ `\d`)
`[[:name:]]`	named ASCII class inside character class (≡ `[:name:]`)
`[^[:name:]]`	named ASCII class inside negated character class (≡ `[:^name:]`)
`[\p{Name}]`	named Unicode property inside character class (≡ `\p{Name}`)
`[^\p{Name}]`	named Unicode property inside negated character class (≡ `\P{Name}`)

Perl character classes

ASCII-only

Example	Description
`\d`	digits (≡ `[0-9]`)
`\D`	not digits (≡ `[^0-9]`)
`\s`	whitespace (≡ `[\t\n\f\r ]`)
`\S`	not whitespace (≡ `[^\t\n\f\r ]`)
`\w`	word characters (≡ `[0-9A-Za-z_]`)
`\W`	not word characters (≡ `[^0-9A-Za-z_]`)
`\h`	horizontal space (NOT SUPPORTED)
`\H`	not horizontal space (NOT SUPPORTED)
`\v`	vertical space (NOT SUPPORTED)

Example	Description
`\V`	not vertical space (NOT SUPPORTED)

ASCII character classes

Example	Description
`[[:alnum:]]`	alphanumeric (≡ `[0-9A-Za-z]`)
`[[:alpha:]]`	alphabetic (≡ `[A-Za-z]`)
`[[:ascii:]]`	ASCII (≡ `[\x00-\x7F]`)
`[[:blank:]]`	blank (≡ `[\t ]`)
`[[:cntrl:]]`	control (≡ `[\x00-\x1F\x7F]`)
`[[:digit:]]`	digits (≡ `[0-9]`)
`[[:graph:]]`	graphical (≡ `[!-~]` ≡ `[A-Za-z0-9!"#$%&'()*+,\-./:;<=>?@[\\\]^_ { |}~]`)
`[[:lower:]]`	lower case (≡ `[a-z]`)
`[[:print:]]`	printable (≡ `[ -~]` ≡ `[ [:graph:]]`)
`[[:punct:]]`	punctuation (≡ `[!-/:-@[-` `{-~]`)
`[[:space:]]`	whitespace (≡ `[\t\n\v\f\r ]`)
`[[:upper:]]`	upper case (≡ `[A-Z]`)
`[[:word:]]`	word characters (≡ `[0-9A-Za-z_]`)
`[[:xdigit:]]`	hex digit (≡ `[0-9A-Fa-f]`)

Unicode character class names

General

Example	Description
`C`	other
`Cc`	control
`Cf`	format

Example	Description
`Cn`	unassigned code points (NOT SUPPORTED)
`Co`	private use
`Cs`	surrogate
`L`	letter
`LC`	cased letter (NOT SUPPORTED)
`L&`	cased letter (NOT SUPPORTED)
`Ll`	lowercase letter
`Lm`	modifier letter
`Lo`	other letter
`Lt`	titlecase letter
`Lu`	uppercase letter
`M`	mark
`Mc`	spacing mark
`Me`	enclosing mark
`Mn`	non-spacing mark
`N`	number
`Nd`	decimal number
`Nl`	letter number
`No`	other number
`P`	punctuation
`Pc`	connector punctuation
`Pd`	dash punctuation
`Pe`	close punctuation
`Pf`	final punctuation
`Pi`	initial punctuation

Example	Description
`Po`	other punctuation
`Ps`	open punctuation
`S`	symbol
`Sc`	currency symbol
`Sk`	modifier symbol
`Sm`	math symbol
`So`	other symbol
`Z`	separator
`Zl`	line separator
`Zp`	paragraph separator
`Zs`	space separator

Scripts

Scripts
`Adlam`
`Ahom`
`Anatolian_Hieroglyphs`
`Arabic`
`Armenian`
`Avestan`
`Balinese`
`Bamum`
`Bassa_Vah`
`Batak`
`Bengali`

Scripts
`Bhaiksuki`
`Bopomofo`
`Brahmi`
`Braille`
`Buginese`
`Buhid`
`Canadian_Aboriginal`
`Carian`
`Caucasian_Albanian`
`Chakma`
`Cham`
`Cherokee`
`Chorasmian`
`Common`
`Coptic`
`Cuneiform`
`Cypriot`
`Cyrillic`
`Deseret`
`Devanagari`
`Dives_Akuru`
`Dogra`
`Duployan`
`Egyptian_Hieroglyphs`
`Elbasan`

Scripts
`Elymaic`
`Ethiopic`
`Georgian`
`Glagolitic`
`Gothic`
`Grantha`
`Greek`
`Gujarati`
`Gunjala_Gondi`
`Gurmukhi`
`Han`
`Hangul`
`Hanifi_Rohingya`
`Hanunoo`
`Hatran`
`Hebrew`
`Hiragana`
`Imperial_Aramaic`
`Inherited`
`Inscriptional_Pahlavi`
`Inscriptional_Parthian`
`Javanese`
`Kaithi`
`Kannada`
`Katakana`

Scripts
`Kayah_Li`
`Kharoshthi`
`Khitan_Small_Script`
`Khmer`
`Khojki`
`Khudawadi`
`Lao`
`Latin`
`Lepcha`
`Limbu`
`Linear_A`
`Linear_B`
`Lisu`
`Lycian`
`Lydian`
`Mahajani`
`Makasar`
`Malayalam`
`Mandaic`
`Manichaean`
`Marchen`
`Masaram_Gondi`
`Medefaidrin`
`Meetei_Mayek`
`Mende_Kikakui`

Scripts
`Meroitic_Cursive`
`Meroitic_Hieroglyphs`
`Miao`
`Modi`
`Mongolian`
`Mro`
`Multani`
`Myanmar`
`Nabataean`
`Nandinagari`
`New_Tai_Lue`
`Newa`
`Nko`
`Nushu`
`Nyiakeng_Puachue_Hmong`
`Ogham`
`Ol_Chiki`
`Old_Hungarian`
`Old_Italic`
`Old_North_Arabian`
`Old_Permic`
`Old_Persian`
`Old_Sogdian`
`Old_South_Arabian`
`Old_Turkic`

Scripts
`Oriya`
`Osage`
`Osmanya`
`Pahawh_Hmong`
`Palmyrene`
`Pau_Cin_Hau`
`Phags_Pa`
`Phoenician`
`Psalter_Pahlavi`
`Rejang`
`Runic`
`Samaritan`
`Saurashtra`
`Sharada`
`Shavian`
`Siddham`
`SignWriting`
`Sinhala`
`Sogdian`
`Sora_Sompeng`
`Soyombo`
`Sundanese`
`Syloti_Nagri`
`Syriac`
`Tagalog`

Scripts
`Tagbanwa`
`Tai_Le`
`Tai_Tham`
`Tai_Viet`
`Takri`
`Tamil`
`Tangut`
`Telugu`
`Thaana`
`Thai`
`Tibetan`
`Tifinagh`
`Tirhuta`
`Ugaritic`
`Vai`
`Wancho`
`Warang_Citi`
`Yezidi`
`Yi`
`Zanabazar_Square`

Vim character classes

Example	Description
`\i`	identifier character (NOT SUPPORTED) VIM
`\I`	`\i` except digits (NOT SUPPORTED) VIM

Example	Description
`\k`	keyword character (NOT SUPPORTED) VIM
`\K`	`\k` except digits (NOT SUPPORTED) VIM
`\f`	file name character (NOT SUPPORTED) VIM
`\F`	`\f` except digits (NOT SUPPORTED) VIM
`\p`	printable character (NOT SUPPORTED) VIM
`\P`	`\p` except digits (NOT SUPPORTED) VIM
`\s`	whitespace character (≡ `[ \t]`) (NOT SUPPORTED) VIM
`\S`	non-white space character (≡ `[^ \t]`) (NOT SUPPORTED) VIM
`\d`	digits (≡ `[0-9]`) VIM
`\D`	not `\d` VIM
`\x`	hex digits (≡ `[0-9A-Fa-f]`) (NOT SUPPORTED) VIM
`\X`	not `\x` (NOT SUPPORTED) VIM
`\o`	octal digits (≡ `[0-7]`) (NOT SUPPORTED) VIM
`\O`	not `\o` (NOT SUPPORTED) VIM
`\w`	word character VIM
`\W`	not `\w` VIM
`\h`	head of word character (NOT SUPPORTED) VIM
`\H`	not `\h` (NOT SUPPORTED) VIM
`\a`	alphabetic (NOT SUPPORTED) VIM
`\A`	not `\a` (NOT SUPPORTED) VIM
`\l`	lowercase (NOT SUPPORTED) VIM
`\L`	not lowercase (NOT SUPPORTED) VIM
`\u`	uppercase (NOT SUPPORTED) VIM
`\U`	not uppercase (NOT SUPPORTED) VIM
`\_x`	`\x` plus newline, for any `x` (NOT SUPPORTED) VIM

Example	Description
`\c`	ignore case (NOT SUPPORTED) VIM
`\C`	match case (NOT SUPPORTED) VIM
`\m`	magic (NOT SUPPORTED) VIM
`\M`	nomagic (NOT SUPPORTED) VIM
`\v`	verymagic (NOT SUPPORTED) VIM
`\V`	verynomagic (NOT SUPPORTED) VIM
`\Z`	ignore differences in Unicode combining characters (NOT SUPPORTED) VIM

Magic

Example	Description
`(?{code})`	arbitrary Perl code (NOT SUPPORTED) PERL
`(??{code})`	postponed arbitrary Perl code (NOT SUPPORTED) PERL
`(?n)`	recursive call to regexp capturing group `n` (NOT SUPPORTED)
`(?+n)`	recursive call to relative group `+n` (NOT SUPPORTED)
`(?-n)`	recursive call to relative group `-n` (NOT SUPPORTED)
`(?C)`	PCRE callout (NOT SUPPORTED) PCRE
`(?R)`	recursive call to entire regexp (≡ `(?0)`) (NOT SUPPORTED)
`(?&name)`	recursive call to named group (NOT SUPPORTED)
`(?P=name)`	named backreference (NOT SUPPORTED)
`(?P>name)`	recursive call to named group (NOT SUPPORTED)
`(?(cond)true|false)`	conditional branch (NOT SUPPORTED)
`(?(cond)true)`	conditional branch (NOT SUPPORTED)
`(*ACCEPT)`	make regexps more like Prolog (NOT SUPPORTED)
`(*COMMIT)`	(NOT SUPPORTED)
`(*F)`	(NOT SUPPORTED)

Example	Description
`(*FAIL)`	(NOT SUPPORTED)
`(*MARK)`	(NOT SUPPORTED)
`(*PRUNE)`	(NOT SUPPORTED)
`(*SKIP)`	(NOT SUPPORTED)
`(*THEN)`	(NOT SUPPORTED)
`(*ANY)`	set newline convention (NOT SUPPORTED)
`(*ANYCRLF)`	(NOT SUPPORTED)
`(*CR)`	(NOT SUPPORTED)
`(*CRLF)`	(NOT SUPPORTED)
`(*LF)`	(NOT SUPPORTED)
`(*BSR_ANYCRLF)`	set \R convention (NOT SUPPORTED) PCRE
`(*BSR_UNICODE)`	(NOT SUPPORTED) PCRE

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

JSONPath expressions

Article • 01/16/2023

JSONPath notation describes the path to one or more elements in a JSON document.

The JSONPath notation is used in the following scenarios:

- To specify data mappings for ingestion
- To specify data mappings for external tables
- In KQL functions that process dynamic objects, like bag_remove_keys() and extract_json()

The following subset of the JSONPath notation is supported:

Path expression	Description
`$`	Root object
`.`	Selects the specified property in a parent object. Use this notation if the property doesn't contain special characters.
`['property']` or `["property"]`	Selects the specified property in a parent object. Make sure you put single quotes or double quotes around the property name. Use this notation if the property name contains special characters, such as spaces, or begins with a character other than `A..Za..z_`.
`[n]`	Selects the n-th element from an array. Indexes are 0-based.

> ⓘ Note
>
> Wildcards, recursion, union, slices, and current object are not supported.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

SQL to Kusto Query Language cheat sheet

Article • 05/01/2023

If you're familiar with SQL and want to learn KQL, translate SQL queries into KQL by prefacing the SQL query with a comment line, `--`, and the keyword `explain`. The output will show the KQL version of the query, which can help you understand the KQL syntax and concepts.

Run the query

```
Kusto
```

```
--
explain
SELECT COUNT_BIG(*) as C FROM StormEvents
```

Output

Query
StormEvents | summarize C=count() | project C

SQL to Kusto cheat sheet

The table below shows sample queries in SQL and their KQL equivalents.

Category	SQL Query	Kusto Query
Select data from table	`SELECT * FROM dependencies`	`dependencies`
--	`SELECT name, resultCode FROM dependencies`	`dependencies | project name, resultCode`
--	`SELECT TOP 100 * FROM dependencies`	`dependencies | take 100`
Null evaluation	`SELECT * FROM dependencies` `WHERE resultCode IS NOT NULL`	`dependencies` `| where isnotnull(resultCode)`

Category	SQL Query	Kusto Query
Comparison operators (date)	`SELECT * FROM dependencies` `WHERE timestamp > getdate()-1`	`dependencies` `| where timestamp > ago(1d)`

--	`SELECT * FROM dependencies`
	`WHERE timestamp BETWEEN ... AND ...`
		`dependencies`
		`| where timestamp between`
		`(datetime(2016-10-01) ..`
		`datetime(2016-11-01))`

Comparison operators (string)	`SELECT * FROM dependencies` `WHERE type = "Azure blob"`	`dependencies` `| where type == "Azure blob"`

--	`-- substring`
	`SELECT * FROM dependencies`
	`WHERE type like "%blob%"`
		`// substring`
		`dependencies`
		`| where type contains "blob"`

--	`-- wildcard`
	`SELECT * FROM dependencies`
	`WHERE type like "Azure%"`
		`// wildcard`
		`dependencies`
		`| where type startswith "Azure"`
		`// or`
		`dependencies`
		`| where type matches regex`
		`"^Azure.*"`

Comparison (boolean)	`SELECT * FROM dependencies` `WHERE !(success)`	`dependencies` `| where success == "False"`

Grouping, Aggregation	`SELECT name, AVG(duration) FROM`
	`dependencies`
	`GROUP BY name`
		`dependencies`
		`| summarize avg(duration) by`
		`name`

Distinct	`SELECT DISTINCT name, type FROM` `dependencies`	`dependencies` `| summarize by name, type`

--	`SELECT name, COUNT(DISTINCT type)`
	`FROM dependencies`
	`GROUP BY name`
		`dependencies`
		`| summarize by name, type |`
		`summarize count() by name`
		`// or approximate for large sets`
		`dependencies`
		`| summarize dcount(type) by name`

Column aliases, Extending	`SELECT operationName as Name,`
	`AVG(duration) as AvgD FROM dependencies`
	`GROUP BY name`
		`dependencies`
		`| summarize AvgD = avg(duration)`
		`by Name=operationName`

Category	SQL Query	Kusto Query

--	`SELECT conference, CONCAT(sessionid, ' ' , session_title) AS session FROM ConferenceSessions`	`ConferenceSessions`
		`| extend session=strcat(sessionid, " ", session_title)`
		`| project conference, session`

Ordering	`SELECT name, timestamp FROM dependencies`
	`ORDER BY timestamp ASC`
		`dependencies`
		`| project name, timestamp`
		`| order by timestamp asc nulls last`

Top n by measure	`SELECT TOP 100 name, COUNT(*) as Count`
	`FROM dependencies`
	`GROUP BY name`
	`ORDER BY Count DESC`
		`dependencies`
		`| summarize Count = count() by name`
		`| top 100 by Count desc`

Union	`SELECT * FROM dependencies`
	`UNION`
	`SELECT * FROM exceptions`
		`union dependencies, exceptions`

--	`SELECT * FROM dependencies`
	`WHERE timestamp > ...`
	`UNION`
	`SELECT * FROM exceptions`
	`WHERE timestamp > ...`
		`dependencies`
		`| where timestamp > ago(1d)`
		`| union`
		`  (exceptions`
		`  | where timestamp > ago(1d))`

Join	`SELECT * FROM dependencies`
	`LEFT OUTER JOIN exception`
	`ON dependencies.operation_Id = exceptions.operation_Id`
		`dependencies`
		`| join kind = leftouter`
		`  (exceptions)`
		`on $left.operation_Id == $right.operation_Id`

Nested queries	`SELECT * FROM dependencies`
	`WHERE resultCode ==`
	`(SELECT TOP 1 resultCode FROM dependencies`
	`WHERE resultId = 7`
	`ORDER BY timestamp DESC)`
		`dependencies`
		`| where resultCode == toscalar(`
		`  dependencies`
		`  | where resultId == 7`
		`  | top 1 by timestamp desc`
		`  | project resultCode)`

Having	`SELECT COUNT(*) FROM dependencies`
	`GROUP BY name`
	`HAVING COUNT(*) > 3`
		`dependencies`
		`| summarize Count = count() by name`
		`| where Count > 3`

Next steps

- Use T-SQL to query data

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Splunk to Kusto Query Language map

Article • 05/01/2023

This article is intended to assist users who are familiar with Splunk learn the Kusto Query Language to write log queries with Kusto. Direct comparisons are made between the two to highlight key differences and similarities, so you can build on your existing knowledge.

Structure and concepts

The following table compares concepts and data structures between Splunk and Kusto logs:

Concept	Splunk	Kusto	Comment
deployment unit	cluster	cluster	Kusto allows arbitrary cross-cluster queries. Splunk doesn't.
data caches	buckets	caching and retention policies	Controls the period and caching level for the data. This setting directly affects the performance of queries and the cost of the deployment.
logical partition of data	index	database	Allows logical separation of the data. Both implementations allow unions and joining across these partitions.
structured event metadata	N/A	table	Splunk doesn't expose the concept of event metadata to the search language. Kusto logs have the concept of a table, which has columns. Each event instance is mapped to a row.
data record	event	row	Terminology change only.
data record attribute	field	column	In Kusto, this setting is predefined as part of the table structure. In Splunk, each event has its own set of fields.
types	datatype	datatype	Kusto data types are more explicit because they're set on the columns. Both have the ability to work dynamically with data types and roughly equivalent set of datatypes, including JSON support.
query and search	search	query	Concepts essentially are the same between Kusto and Splunk.

Concept	Splunk	Kusto	Comment
event ingestion time	system time	`ingestion_time()`	In Splunk, each event gets a system timestamp of the time the event was indexed. In Kusto, you can define a policy called ingestion_time that exposes a system column that can be referenced through the ingestion_time() function.

Functions

The following table specifies functions in Kusto that are equivalent to Splunk functions.

Splunk	Kusto	Comment
`strcat`	`strcat()`	(1)
`split`	`split()`	(1)
`if`	`iff()`	(1)

`tonumber`	`todouble()`
	`tolong()`
	`toint()`
		(1)

`upper` `lower`	`toupper()` `tolower()`	(1)
`replace`	`replace_string()`, `replace_strings()` or `replace_regex()`	(1) Although `replace` functions take three parameters in both products, the parameters are different.
`substr`	`substring()`	(1) Also note that Splunk uses one-based indices. Kusto notes zero-based indices.
`tolower`	`tolower()`	(1)
`toupper`	`toupper()`	(1)
`match`	`matches regex`	(2)
`regex`	`matches regex`	In Splunk, `regex` is an operator. In Kusto, it's a relational operator.
`searchmatch`	`==`	In Splunk, `searchmatch` allows searching for the exact string.

Splunk	Kusto	Comment
`random`	rand() rand(n)	Splunk's function returns a number between zero to 2^{31}-1. Kusto's returns a number between 0.0 and 1.0, or if a parameter is provided, between 0 and n-1.
`now`	`now()`	(1)
`relative_time`	`totimespan()`	(1) In Kusto, Splunk's equivalent of `relative_time(datetimeVal, offsetVal)` is `datetimeVal + totimespan(offsetVal)`. For example, `search | eval n=relative_time(now(), "-1d@d")` becomes `... | extend myTime = now() - totimespan("1d")`.

(1) In Splunk, the function is invoked by using the `eval` operator. In Kusto, it's used as part of `extend` or `project`.
(2) In Splunk, the function is invoked by using the `eval` operator. In Kusto, it can be used with the `where` operator.

Operators

The following sections give examples of how to use different operators in Splunk and Kusto.

> ⓘ Note
>
> In the following examples, the Splunk field `rule` maps to a table in Kusto, and Splunk's default timestamp maps to the Logs Analytics `ingestion_time()` column.

Search

In Splunk, you can omit the `search` keyword and specify an unquoted string. In Kusto, you must start each query with `find`, an unquoted string is a column name, and the lookup value must be a quoted string.

Product	Operator	Example
Splunk	`search`	`search Session.Id="c8894ffd-e684-43c9-9125-42adc25cd3fc" earliest=-24h`

Product	Operator	Example
Kusto	`find`	`find Session.Id=="c8894ffd-e684-43c9-9125-42adc25cd3fc" and ingestion_time()> ago(24h)`

Filter

Kusto log queries start from a tabular result set in which `filter` is applied. In Splunk, filtering is the default operation on the current index. You also can use the `where` operator in Splunk, but we don't recommend it.

Product	Operator	Example
Splunk	`search`	`Event.Rule="330009.2" Session.Id="c8894ffd-e684-43c9-9125-42adc25cd3fc" _indextime>-24h`
Kusto	`where`	`Office_Hub_OHubBGTaskError` `| where Session_Id == "c8894ffd-e684-43c9-9125-42adc25cd3fc" and ingestion_time() > ago(24h)`

Get *n* events or rows for inspection

Kusto log queries also support `take` as an alias to `limit`. In Splunk, if the results are ordered, `head` returns the first *n* results. In Kusto, `limit` isn't ordered, but it returns the first *n* rows that are found.

Product	Operator	Example
Splunk	`head`	`Event.Rule=330009.2` `| head 100`
Kusto	`limit`	`Office_Hub_OHubBGTaskError` `| limit 100`

Get the first *n* events or rows ordered by a field or column

For the bottom results, in Splunk, you use `tail`. In Kusto, you can specify ordering direction by using `asc`.

Product	Operator	Example

Product	Operator	Example

Splunk	`head`	`Event.Rule="330009.2"`
		`| sort Event.Sequence`
		`| head 20`

Kusto	`top`	`Office_Hub_OHubBGTaskError` `| top 20 by Event_Sequence`

Extend the result set with new fields or columns

Splunk has an `eval` function, but it's not comparable to the `eval` operator in Kusto. Both the `eval` operator in Splunk and the `extend` operator in Kusto support only scalar functions and arithmetic operators.

Product	Operator	Example
Splunk	`eval`	`Event.Rule=330009.2` `| eval state= if(Data.Exception = "0", "success", "error")`
Kusto	`extend`	`Office_Hub_OHubBGTaskError` `| extend state = iff(Data_Exception == 0,"success" ,"error")`

Rename

Kusto uses the `project-rename` operator to rename a field. In the `project-rename` operator, a query can take advantage of any indexes that are prebuilt for a field. Splunk has a `rename` operator that does the same.

Product	Operator	Example
Splunk	`rename`	`Event.Rule=330009.2` `| rename Date.Exception as execption`
Kusto	`project-rename`	`Office_Hub_OHubBGTaskError` `| project-rename exception = Date_Exception`

Format results and projection

Splunk uses the `table` command to select which columns to include in the results. Kusto has a `project` operator that does the same and more.

Product	Operator	Example

Product	Operator	Example
Splunk	`table`	`Event.Rule=330009.2` `| table rule, state`
Kusto	`project`	`Office_Hub_OHubBGTaskError` `| project exception, state`

Splunk uses the `field -` command to select which columns to exclude from the results. Kusto has a `project-away` operator that does the same.

Product	Operator	Example
Splunk	`fields -`	`Event.Rule=330009.2` `| fields - quota, hightest_seller`
Kusto	`project-away`	`Office_Hub_OHubBGTaskError` `| project-away exception, state`

Aggregation

See the list of summarize aggregations functions that are available.

Splunk operator	Splunk example	Kusto operator	Kusto example
`stats`	`search (Rule=120502.*)` `| stats count by OSEnv, Audience`	`summarize`	`Office_Hub_OHubBGTaskError` `| summarize count() by App_Platform, Release_Audience`
`evenstats`	`...` `| stats count_i by time, category` `| eventstats sum(count_i) AS count_total by _time_`	`join`	`T2` `| join kind=inner (T1) on _time` `| project _time, category, count_i, count_total`

Join

`join` in Splunk has substantial limitations. The subquery has a limit of 10,000 results (set in the deployment configuration file), and a limited number of join flavors are available.

Product	Operator	Example

Product	Operator	Example
Splunk	`join`	`Event.Rule=120103*	stats by Client.Id, Data.Alias` `| join Client.Id max=0 [search earliest=-24h Event.Rule="150310.0"` `Data.Hresult=-2147221040]`

Kusto	`join`	`cluster("OAriaPPT").database("Office`
		`PowerPoint").Office_PowerPoint_PPT_Exceptions`
		`| where Data_Hresult== -2147221040`
		`| join kind = inner (Office_System_SystemHealthMetadata`
		`| summarize by Client_Id, Data_Alias)on Client_Id`

Sort

In Splunk, to sort in ascending order, you must use the `reverse` operator. Kusto also supports defining where to put nulls, either at the beginning or at the end.

Product	Operator	Example

Splunk	`sort`	`Event.Rule=120103`
		`| sort Data.Hresult`
		`| reverse`

Kusto	`order by`	`Office_Hub_OHubBGTaskError` `| order by Data_Hresult, desc`

Multivalue expand

The multivalue expand operator is similar in both Splunk and Kusto.

Product	Operator	Example
Splunk	`mvexpand`	`mvexpand solutions`
Kusto	`mv-expand`	`mv-expand solutions`

Result facets, interesting fields

In Log Analytics in the Azure portal, only the first column is exposed. All columns are available through the API.

Product	Operator	Example

Product	Operator	Example
Splunk	`fields`	`Event.Rule=330009.2` `| fields App.Version, App.Platform`
Kusto	`facets`	`Office_Excel_BI_PivotTableCreate` `| facet by App_Branch, App_Version`

Deduplicate

In Kusto, you can use `summarize arg_min()` to reverse the order of which record is chosen.

Product	Operator	Example
Splunk	`dedup`	`Event.Rule=330009.2` `| dedup device_id sortby -batterylife`
Kusto	`summarize arg_max()`	`Office_Excel_BI_PivotTableCreate` `| summarize arg_max(batterylife, *) by device_id`

Next steps

- Walk through a tutorial on the Kusto Query Language.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Timezones

Article • 03/09/2023

The following is a list of timezones supported by the Internet Assigned Numbers Authority (IANA) Time Zone Database.

Related functions:

- datetime_local_to_utc()
- datetime_utc_to_local()

Timezone
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam

Timezone
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena

Timezone
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman

Timezone
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba

Timezone
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Nelson
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana

Timezone
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes

Timezone
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem

Timezone
America/Nuuk
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Punta_Arenas
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock

Timezone
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo

Timezone
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Atyrau
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Barnaul
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Kolkata
Asia/Chita

Timezone
Asia/Choibalsan
Asia/Chongqing
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Famagusta
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh_City
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu

Timezone
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qostanay
Asia/Qyzylorda
Asia/Yangon (Rangoon)
Asia/Riyadh

Timezone
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Tomsk
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yangon
Asia/Yekaterinburg
Asia/Yerevan

Timezone
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW

Timezone
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba

Timezone
EET
EST
EST5EDT
Egypt
Eire
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5
Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14

Timezone
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/GMT0
Etc/Greenwich
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Astrakhan
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen

Timezone
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kyiv
Europe/Kirov
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica

Timezone
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Saratov
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Ulyanovsk
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB

Timezone
GB-Eire
GMT
GMT+0
GMT-0
GMT0
Greenwich
HST
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya

Timezone
MET
MST
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam

Timezone
Pacific/Honolulu
Pacific/Johnston
Pacific/Kanton
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk

Timezone
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROK
Singapore
Türkiye
UCT
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Samoa
UTC
Universal
W-SU
WET
Zulu