

Types

<code>100 0x64 0b1100100</code>	Long numbers
<code>3.1415 1e3 NaN Infinity -Infinity</code>	Double precision floating point numbers
<code>true false</code>	Booleans
<code>'hundred %25' "WarpScript"</code>	Text strings, URL encoded using UTF-8
<code><'
multiline
string
'></code>	<' and '> must appear alone on a line of their own
<code>[1 2.0 '3']</code>	List of objects
<code>{ 'key1' 42 'key2' 'value2' }</code>	Map of key-value pairs
<code>('unique' 'objects' 42)</code>	Set of objects
<code>36 'myvariable' STORE</code>	Store 36 in a variable
<code>\$myvariable !\$myvariable</code>	Use myvariable
<code><%
SAVE 'context' STORE
['arg1' 'arg2'] STORE
\$arg1 \$arg2 /
\$context RESTORE
%> 'mymacro' STORE</code>	Macro definition
<code>44 22 @mymacro</code>	Call 'mymacro' with parameters 44 and 22
<code>NEWGTS 'mygts' RENAME</code>	Create and name a Geo Time Series™ object
<code>NOW 24.4 -5.4 NaN 42 ADDVALUE</code>	Add a datapoint to a Geo Time Series™ with value 42 at the current timestamp at location 24.4 N and 5.4 W

Comments

<code>// WarpScript comment</code>	Single line comment
<code># WarpScript comment</code>	Single line comment
<code>/* WarpScript comment */</code>	Multi line comment

Function calls

<code>arg1 arg2 arg3 ... FUNCTION</code>	Call 'FUNCTION' with arguments arg1 arg2 arg3 ...
<code>PI COS</code>	Returns cos π
<code>1 1 +</code>	Returns 2

String Manipulation

<code>+, JOIN</code>	Concatenation
<code>SUBSTRING, SPLIT, TRIM</code>	Substring generation
<code>MATCH, REPLACE, REPLACEALL</code>	Regular expression or simple search

Operators

<code>+ - * / % **</code>	Add, subtract, multiply, divide, modulo, power
<code>== != < <= > >=</code>	Comparison operators
<code>&& ! AND OR NOT</code>	Boolean operators
<code>& ^ >> << >>> <<<</code>	Binary operators

Frameworks

<code>BUCKETIZE</code>	Apply a function to points in a bucket. Typically used for downsampling
<code>MAP</code>	Apply a function to a sliding window. Typically used for smoothing
<code>REDUCE</code>	For each timestamp, apply a function to each datapoint of input series, e.g. adding values one to one
<code>FILTER</code>	Used to select Geo Time Series™ from a larger set
<code>FILL</code>	Fill missing values in two Geo Time Series™

Composite Structures Manipulation

<code>+! APPEND</code>	Add elements to a list. +! will reuse the list
<code>[2 3 4] <% DROP 2 ** %> LMAP //returns [4 9 16]</code>	Map the square macro on every element of the list to build a new list
<code>{ } 'value' 'key' PUT</code>	Put a key value pair in an empty map
<code>index GET, 'key' GET</code>	Extract a value from a list or a map
<code>CONTAINS, CONTAINSKEY, CONTAINSVALUE</code>	Test presence in a list or a map
<code>LSORT, SORTBY, MSORT, REVERSE</code>	Sorting function for list and maps

Control / Flow

<code><% if-macro %> <% then-macro %> IFT</code>	If then
<code><% if-macro %> <% then-macro %> <% else-macro %> IFTE</code>	If then else
<code>0 10 <% 'i' STORE %> FOR</code>	For i = 0 to 10
<code>[2 3 4] <% 'arg' STORE \$arg 2 ** %> FOREACH</code>	Iterate on every element of the list
<code><% try-macro %> <% catch-macro %> <% finally-macro %> TRY</code>	Try catch. You can use ERROR function in the catch macro
<code>BREAK</code>	Break out of the current loop
<code>CONTINUE</code>	Go to the next iteration of the loop
<code>RETURN</code>	Return immediately from the currently executing macro

Debugging WarpScript™

STOP	Stop the WarpScript execution
FAIL, 'my error' MSGFAIL	Stop the WarpScript execution with an error
SECTION	Name a portion of code
LINEON	Enable automatic section naming based on line number

Stack Manipulation

SWAP	Swap the top 2 elements of the stack
DUP	Duplicate the top element of the stack
DROP	Drop the top element of the stack
CLEAR	Empty the stack
ROT	Cycle the top 3 elements of the stack
N ROLL	Cycle up one level the top N elements of the stack
N ROLLD	Cycle down one level the top N elements of the stack
SNAPSHOT	Generate WarpScript code which will regenerate the content of the stack

Conversions

->JSON, JSON->	Serialize / deserialize JSON structures
->B64, B64->	Encode / decode string in base64

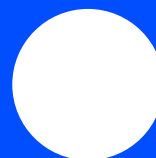
Date and Time functions

NOW	Push onto the stack the current timestamp in platform time units
ps, ns, us, ms, s, m, h, d, w	Push onto the stack the number of platform time units for the specified duration, e.g. 4 d will push the number of time units in 4 days
NOW 'optional_timezone' TSELEMENTS	Convert a timestamp into a list of timestamp elements
NOW 5 ADDDAYS	Add days to a timestamp or a list of timestamp elements
NOW 4 ADDMONTHS	Add months to a timestamp or a list of timestamp elements
[2009 02 14 6 01 31] 'optional_timezone' TSELEMENTS->	Convert a list of timestamp elements into a timestamp
'2016-01-19T16:07:37Z' TOTIMESTAMP	Convert a date in ISO8601 format into a timestamp in the platform time units
\$ts 'optional_timezone' ISO8601	Convert a timestamp into its ISO8601 representation using UTC or the specified time zone

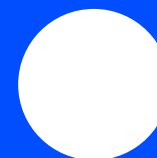
Geo Time Series™ (GTS) functions

NEWGTS	Push onto the stack an empty GTS instance
\$gts 'foo' RENAME	Change the name of a GTS
\$gts { 'foo' 'bar' 'id' '42' } RELABEL	Change the labels of a GTS
\$gts WRAP	Pack a GTS (or a list thereof) or a GTS Encoder into a STRING
\$wrapped_gts UNWRAP	Unpack a GTS (or a list thereof) or a GTS Encoder from a STRING
\$gts [[\$t0 \$t1] [\$t2 \$t3] [...]] CLIP	Clip a GTS according to a series of tick intervals
\$gts \$end_tick \$duration TIMECLIP	Clip GTS by restricting their ticks to those within a time interval
\$bucketized_gts FILLNEXT	Fill gaps in a bucketized GTS by re-using the value/location/elevation of the next non empty bucket to fill each empty bucket
\$bucketized_gts FILLPREVIOUS	Fill gaps in a bucketized GTS by re-using the value/location/elevation of the previous non empty bucket to fill each empty bucket
\$bucketized_gts [\$lat \$long \$elevation \$value] FILLVALUE	Fill gaps in a bucketized GTS with a fixed value/location/elevation
\$gts DEDUP	Remove duplicate ticks
\$gts_list MERGE	Merge multiple GTS
\$gts \$size SHRINK	Shrink the number of values of a GTS to size
\$gts \$n LTTB	Select n meaningful values in a GTS
\$gts COMPACT	Remove measurements which have the same value, location and elevation as the previous one
\$gts RANGECOMPACT	Retain only the first and last measurement of ranges with identical value location and elevation
COMMONTICK	Keep only the common timestamps of all input GTS
[\$token 'classname' { 'label1' 'value1' } \$end \$duration] FETCH	Retrieve data according to fetch criteria
[\$token 'classname' { 'label1' 'value1' }] FIND	Identify GTS according to criteria on classes and labels / attributes
[\$token 'classname' { 'label1' 'value1' }] FINDSETS	Return sets of values for the classes, labels and attributes
[\$token 'classname' { 'label1' 'value1' }] FINDSTATS	Compute statistics on matching GTS

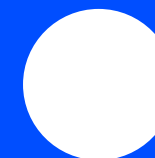
Collect WarpScript badges. If you succeed to collect them all, ask for your diploma!



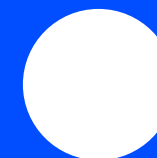
Hello World



FETCH/UPDATE



BUCKETIZE/MAP



Processing



WarpScript on Spark