WATERSLIDE Kids

(for version 1.0.0)

This is a single-page quick reference to the WATERSLIDE (WS) stream processing system, and its modules, called 'kids.' It conforms to version 1.0.0, published on 17 March 2016. This document is version 0.1. It is stored in the WS code development repository, at `doc/developer/waterslide_refsheet.tex`.

To see further documentation, try '`wsman KID`' or '`wsman -v KID`' or '`wsman --help`'.

In general, the ordering of the kids within the groups below is from most commonly used to least commonly used.

Input and Output

`csv_in` - a source for event tuples based on character-delimited data

`file_in` - reads files as memmap-ed binary buffers

`wsproto_in` - reads data or files from stdin in Protocol Buffer formats (wsproto and pbmeta), creates metadata

`print` - prints tuple data to STDOUT or a file

`wsproto_out` - prints metadata in Protocol Buffer format

Data and Tuple Manipulation

`subtuple` - selects data based on presence of tuple members

`removefromtuple` - specifies items by label in a tuple to be removed from a tuple. The emitted tuple will not contain the specified items.

`tuplehash` - generates a hash based on specific items in a tuple. This is used to combine data from different data fields into a single key for state tracking algorithms.

`splittuple` - splits tuple into multiple tuples based on labels. If multiple items exist in a tuple, it will create a tuple for each item. It is also possible to specify data to keep/replicate for each new tuple.

`mergetuple` - merges data from multiple tuples based on a shared, specified key

`mklabelset` - create a label set from labels of tuple members

`appendfirstitem` - appends item to tuple based on key, useful for inference and merging

`appendlast` - appends last item to tuple based on key, useful for inference and merging

Counting

`bandwidth` - tracks items per second, counts all items. Reports output at end of stream

`keycount` - counts number of items with a specific key value. For instance the number of events with name ABCD

`keyadd` - adds values of a given labeled data field and accumulates these sums based on key

`keyaverage` - computes the average value of a data field based on sums of the same key

`labelstat` - counts the occurrence of each label within tuples

Numeric Calculation

`calc` - performs numerical calculations. Can also act as a filter based on numerical results.

Filtering

`uniq` - determines if labeled items are new. Multiple ports allows for removal and queries against the state of each labeled key.

`uniqexpire` - determines if items are new, expires out old items based on timestamp

`firstn` - selects the first N tuples from each key

`mostnew` - declares the first N items as new

`bloom` - determines if items are new, keep track of items using a bloom filter

`sample` - samples items based on probability

`cntquery` - determine if the value of a keys is mostly positive. This module has multiple ports, one for INCREMENTing, one for DECREMENTing, one for querying. This is for finding items sequence conditions that are biased.

Matching

`match` - finds strings in character buffers that match a dictionary of strings at arbitrary offset locations

`fixedmatch` - finds strings in character buffer that match a dictionary of strings at fixed, explicit locations in the character buffer. Can also be used for protocol detection.

`match_unit` - matches integers with specified properties and/or numeric ranges

`equal` - checks to see if two elements in a tuple are equal

`haslabel` - checks to see if a label or set of labels exists in an event or members of a tuple

`re2` - when WS is compiled with Googles re2 library, this allows you to specify perl compatible regular expressions including extracting content

State Tracking

`uniq` - only passes one uniq item. Items selected by label.

`appendfirstitem` - stores first item (value) at a key, subsequent queries of the key results in the stored value appended to the query tuple

`appendlast` - stores last item (value) at a key, subsequent queries of the key results in the stored value appended to the query tuple

`keyflow` - creates a temporal hash based on keys that occur close in time

`keeplast` - stores the last data item at the key. Only outputs data when table is full, or the table is flushed.

`cntquery` - determines of state of key has reached a counting threshold. Uses ports to increment or decrement state. Can also be used to check for imbalance of events, such as HTTP server hostnames occurring without a reference.

`firstn` - passes the first n events for each key

`keepn` - keeps the last n items at the key

`stateclimb` - user specifies labels and values (positive or negative). when label is found the value is added to accumulator for each key. If accumulator reaches a target value, then output is triggered.

Decoding

`base64` - takes an identified base64 encoded buffer and decodes it into another buffer

`asciihex` - takes an ASCII string with hex values and converts them to binary

Distributed Processing

`tcpthrow` - writes TCP messages, sends to connected client

`tcpcatch` - reads from TCP server, creates binary data for parsing

Miscellaneous Commands

`flush` - resets state hash tables, based on number of events seen or elapsed time