SiLK Tool Suite Quick Reference

Binary Output SiLK Tools

(See back cover for list of text output tools)

Black tools produce flow binary.

Green tools produce bag binary. (B)

Blue tools produce pcap binary. (P)

Purple tools produce IP set binary. (I)

Orange tools produce other binary formats. (O)

SiLK Flow Record Fields

#	Name	Description	
1	sip	Source IP address	Five Tuple (key for flow)
2	dip	Destination IP address	
3	sport	Source port	
4	dport	Destination port	
5	proto	Protocol value in IP header	
6	packets	Packet count	
7	bytes	Byte count	
8	flags	TCP flags from all packets	
9	stime	Start time	
10	dur	Duration	
11	etime	End time	
12	sensor	Sensor number	
13	in	(Unused)	
14	out	(Unused)	
15	nhip	(Used for marking)	
16	stype	index from address_types.pmap for source IP address	
17	dtype	index from address_types.pmap for destination IP address	
18	scc	Country code of source IP address	
19	dcc	Country code of destination IP address	
20	class	Sensor category	
21	type	Flow category (`in`, `out`, `inweb`, `outweb`, etc.)	
22	stime+msec	Start time forcing milliseconds	
23	dur+msec	Duration forcing milliseconds	
24	etime+msec	End time forcing milliseconds	
25	icmptypecode	ICMP type & code	
26	initialflags	TCP flags for first packet	
27	sessionflags	TCP flags for later packets	
28	attributes	Termination conditions	
29	application	Service recognition	
	itype	ICMP type	
	icode	ICMP code	
	src-*mapname*	Label for source IP or proto/port from mapname	
	dst-*mapname*	Label for destination IP or proto/port from mapname	

SiLK Parameter Formats

Parameter order is up to the user except that parameters created via pmaps and plugins must be defined before they are referenced.

General Parameter Formats

--name=argument

Where name may be shortened to the minimum prefix not shared with another parameter (e.g., "protocol" can be "prot" but not "pro" for `rwfilter`)

filename

Where name follows Linux path formats, or may be `stdin` or `stdout` (as appropriate), or named pipe

Argument Formats

Attr-mask	*High/Care,* where both *High* and *Care* are a series of `FTCS` F = additional packets after FIN, T = active timeout, C = continued flow, S = equal size packets
Cc-list	Comma-separated list of top-level country code abbreviations
Cidr-list	Comma-separated list of IP addresses (in dotted-decimal notation) or CIDR blocks
Date	YYYY/MM/DD`T`HH or YYYY/MM/DD
Decimal	Any non-negative decimal number (e.g. 123.4)
Dec-range	*Decimal-Decimal* or *Decimal-*
Dirname	Local or full path naming directory
Fieldlist	Comma-separated list of field names or *Int-range*
Flag-mask	TCP flags as *High/Care* or comma-separated list of *High/Care* Where both *High* and *Care* are a series (no separator) of `SFARPECU`
Integer	Any positive whole number, range specified by context (e.g. 123 or aaa9)
Int-range	*Integer-Integer* or *Integer-*
Int-list	Comma-separated list of *Int-range* or *Integer*
Ip-addr	A CIDR block or a single IPv4 or IPv6 in canonical or integer notation, any field of a canonical address can be an *int-list* or the wild card `x`
Sensors	Comma-separated list of sensor names or *Int-range*
String	Sequence of characters between quotes
Time	YYYY/MM/DD`T`HH or YYYY/MM/DD`T`HH`:`MM or YYYY/MM/DD`T`HH`:`MM`:`SS or YYYY/MM/DD`T`HH`:`MM:SS`.`mmm (`T` can be replaced by `:`)
Time-range	*Time-Time*

Compression Options (Comp. Opt.)

none	No compression
zlib	Best compression, slower performance
lzo1x	Lesser compression, better performance (default)
best	Implementation defined (currently *lzo1x*)

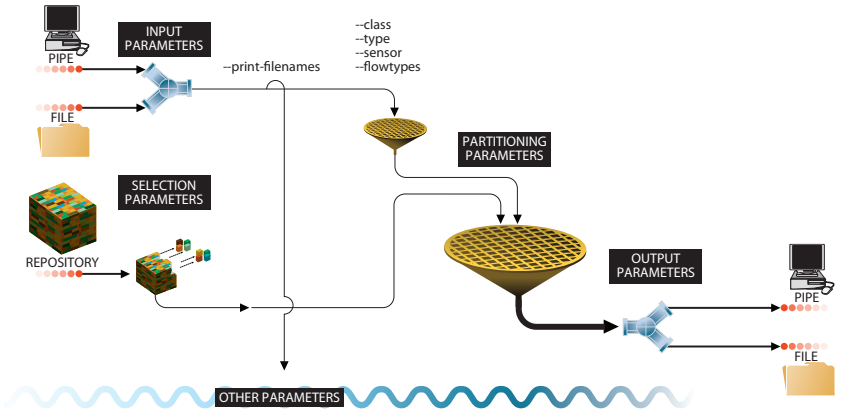
rwfilter

Retrieve flow records from pipe, file, or repository; select records of interest; and store to pipe or file.

Syntax summary: (input or selection [not both], partitioning, and output are required)

```
rwfilter input selection partitioning output other
```

Functional flow diagram:



Examples:

Pull outbound traffic to address block for 8 hours:

```
rwfilter --start=2011/04/15T00 --end=2011/04/15T07 \
  --sensor=SEN1 --type=out --daddress=10.5.x.x \
  --pass=10-5.rw
```

Pull all inbound traffic for 15 minutes:

```
rwfilter --start=2011/04/15T00 --sensor=SEN1 \
  --type=in,inweb \
  --stime=2011/04/15T00:00-2011/04/15T00:15 \
  --pass=first-quarter.rw
```

Choose completed TCP flows with content from a file:

```
rwfilter all-outbound.rw --proto=6 \
  --flags-all=SAF/SAF,SAR/SAR
  --packets=4- --bytes-per=65- --pass=comp-tcp.rw
```

For one hour of traffic, split common protocols from less common into two files:

```
rwfilter --start=2011/04/15T00 --sensor=SEN1 \
  --type=all --proto=1,6,17,50,51 \
  --pass=common.rw --fail=less_common.rw
```

`rwfilter` Parameters

Min-Name	Description	Arguments
	Input Parameters	
data	Root directory of repository	dirname
	Flow files or pipe to filter (no parameter prefix, multiple allowed)	
site-conf	Location of the site configuration file	filename
xarg	Read input file names from file or pipe	filename (opt.)
	Selection Parameters	
class	Class of sensor to process	string
end	Final hour of data to process	date
flowtype	Class/type pairs to process	class/type
sensor	Sensors to process	sensors
start	First hour of data to process	date
type	Types of flow records to process	type
	Output Parameters	
all	Destination for all records	filename
fail	Destination for records that fail	filename
pass	Destination for records that pass	filename
print-miss	Print the names of missing files	none
print-stat	Print a count of total flows	filename (opt.)
print-vol	Print count of flows/packets/bytes	filename (opt.)

Start-date and end-date:

		--start-date		
		Hour	**Day**	**None**
--end-date	**Hour**	Hours in explicit range	Ignore end-date hour. Whole days. (Don't do this)	Error
	Day	End-hour is the same as start-hour. #hours = 1, 25, 49, … (Don't do this)	Whole days.	Error
	None	1 hour	1 day	Current day to present time.

`rwfilter` Parameters (continued)

Min-Name	Description	Arguments
	Other Parameters	
compress	Set compression for output	comp. opt.
dry-run	Report command line errors	none
help	Print command info	none
max-fail	Write at most Arg records to fail	integer (0=all)
max-pass	Write at most Arg records to pass	integer (0=all)
note-add	Put arg in file header	string
note-file	Put content of arg in file header	filename
plugin	Use plugin to filter records	filename
print-file	Print names of input files	none
thread	Set filter threads	integer
version	Print version	none
	Partitioning Parameters	
	also class, flowtype, type, sensor from selection	
active	Flow active during this time window	time-range
any-addr	Src or dest address matches IP (and not-)	ip-addr
any-cc	Src or dest country code	country code list
any-cidr	Src or dest address in this list (and not-)	cidr-list
anyset	Src or dest address is in this IP set (and not-)	filename
aport	Source or destination port in this list	int-list
app	Flow signature label in list	int-list
attrib	Attribute field matches list	attr-mask
bytes	Byte count within this range	int-range
bytes-per	Byte-per-packet count within this range	dec-range
daddr	Destination address matches IP (and not-)	ip-addr
dcc	Destination address maps country code in list	cc-list
dcidr	Destination address in this list (and not-)	cidr-list
dipset	Destination address is in this IP set (and not-)	filename
dport	Destination port in this list	int-list
dtype	Destination address index in address_types.pmap matches Arg	integer
dur	Duration in seconds within this range	dec-range
etime	Ending time within this time window	time-range
flags-all	TCP flags match list	flags-mask
flags-init	Initial TCP flags match list	flags-mask
flags-sess	Session TCP flags match list	flags-mask

`rwfilter` Parameters (continued)

Min-Name	Description	Arguments
	Partitioning Parameters	
icmp-code	ICMP or ICMPv6 code is in this list	int-list
icmp-type	ICMP or ICMPv6 type is in this list	int-list
ip-v	IP version in list	int-list
ipa-any	Src or dest address matches expression	string
ipa-dst	Destination address matches expression	string
ipa-s	Source address matches expression	string
next	nhIP field matches IP (and not-)	ip-addr
nhcidr	nhIP field in this list (and not-)	cidr-list
nhipset	nhIP field is in this IP set (and not-)	filename
packet	Packet count within this range	int-range
pmap-any-MAPNAME	SRC or dest address map labeling matches argument	string
pmap-dst-MAPNAME	Destination address map labeling matches argument	string
pmap-file	Prefix map file to read	filename
pmap-src-MAPNAME	Source address map labeling matches argument	string
proto	Protocol in this list	int-list
python-exp	Run expression	string
python-file	Use Python code to extend processing	filename
saddr	Source address matches IP (and not-)	ip-addr
scc	Source address maps country code in list	cc-list
scidr	Source address in this list (and not-)	cidr-list
sipset	Source address is in this IP set (and not-)	filename
sport	Source port in this list	int-list
stime	Start time within this time window	time-range
stype	Source address index in address_types.pmap matches Arg	integer
tcp-flag	TCP flags are in the list	`FSRPAUEC`
tuple-del	Character separating the fields	character
tuple-dir	Specify IP-port mapping `forward` – as given `reverse` – flip source and destination `both` – do either matching	
tuple-field	Fields in five-tuple with values in tuple file	fieldlist
tuple-file	Record five-tuple fields match value combinations in file	filename

rwcut

Display network flow records as columnar or delimited text.

Syntax summary: (all parameters are optional)

```
rwcut formatting-parameters range-parameters
      output-parameters filename ...
```

Examples:

Quick overview of records in file:

```
rwcut --fields=1-6,stime flows.rw --pager=less
```

Output full records from file in csv format (`sed` command adds space after each comma):

```
rwcut --all-fields --delim=',' flows.rw \
| sed -e 's/,/, /g' >flows.csv
```

Output data with integer IP addresses (rather than dotted-quad) for sorting, plotting, etc.:

```
rwcut --ip-format=decimal --fields=sip,dip \
  flows.rw > flows.txt
```

Changing order of columnar display:

```
rwcut --fields=protocol,sip,sport,dip,dport \
  flows.rw >flows.txt
```

Labeling source addresses using a pmap:

```
rwcut --pmap-file=mal:malware.pmap --pmap-col=10 \
  --fields=src-mal,1-7,stime \
  flows.rw >mal-flows.txt
```

Parameters:

Min-Name	Description	Arguments
	Output Parameters	
copy	Copy all input SiLK flows to given pipe or file	filename
dry-run	Parse options and print column titles only	none
help	Print usage summary	none
help-fields	Print field descriptions	none
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
print-file	Print names of input files as they are opened	none
site-conf	Specify location of the site configuration file	filename
version	Print this program's version	none
	Range Parameters	
all-fields	Print all known fields to the output	none
end-rec	Specify ending record number	integer
fields	Specify fields to print	fieldlist

`rwcut` Parameters (continued)

Min-Name	Description	Arguments
	Range Parameters	
ipv6-policy	Specify how to handle IPv4 and IPv6 records `ignore` – drop IPv6 records `asv4` – convert v6 to v4 else ignore `mix` – allow both `force` – convert v4 to v6 `only` – drop IPv4	
num	Specify number of records to print	integer
pmap-file	Prefix map file to read	map:filename
start-rec	Specify starting record number	integer
tail-recs	Specify starting record number from end of file	integer
xarg	Read input file names from file or pipe	filename (opt.)
	Formatting Parameters	
col	Specify separation character between columns	character
delim	Shortcut for no-columns no-final-del column-sep	character (opt.)
icmp	Print ICMP type & code in sPort and dPort fields	none
integer-sen	Print sensor as an integer	none
integer-tcp	Print TCP flags as an integer	none
ip-format	Specify IP address print format `canonical` – dotted quad (IPv4) or hexadectet (IPv6) `decimal` – integers `force ipv6` – print all addresses as IPv6 `hexadecimal` – base-16 integers `zero-pad` add zeroes to fully fill column	
no-col	Disable fixed-width columnar output	none
no-final	Suppress column delimiter after last field	none
no-titles	Do not print column headers	none
plugin	Load given plugin to add fields	filename
pmap-col	Maximum column width to use for pmap value output	integer
python-file	Use Python code to extend processing	filename
timestamp	Time format options `default` – yyyy/mm/ddThh:mm:ss.sss `iso` – yyyy-mm-dd hh:mm:ss.sss `m/d/y` – mm/dd/yyyy hh:mm:ss.sss `epoch` – seconds since UNIX epoch; ignores timezone `utc` – use UTC timezone `local` – use local timezone `no-msec` – truncate milliseconds	

rwfileinfo

Print summary information about SiLK binary format files (flow, set, bag, etc.)

Syntax summary: (all non-file parameters are optional)

```
rwfileinfo parameters filename ...
```

Examples:

Show all summary information on two files:

```
rwfileinfo flows.rw internal-ip.set
```

Show how generated and any comments:

```
rwfileinfo --fields=command-lines,annotations \
   flows.rw
```

Output info for loading into spreadsheet (without headings):

```
rwfileinfo --no-titles flows.rw
```

Parameters:

Min-Name	Description	Arguments
fields	List of fields to print 1 – format(id); 2 – version; 3 – byte-order 4 – compression(id); 5 – header-length; 6 – record-length 7 – count-records; 8 – file-size; 9 – command-lines 10 – record-version; 11 – silk-version; 12 – packed-file-info 13 – probe-name; 14 – annotations; 15 – prefix-map 16 – IP set; 17 – bag	fieldlist
help	Print usage summary	none
no-titles	Suppress file names and field names	none
site-conf	Specify location of the site configuration file	filename
summary	Print total files; file sizes; records	none
version	Print this program's version	none

rwsiteinfo

Displays information about site collection configuration, including sensor names and numbers. Replaces `mapsid` command from prior versions of SiLK.

Syntax summary: (`fields` parameter is required)

```
rwsiteinfo parameters --fields=site-fields
```

Examples:

Print list of all sensor names and numbers:

```
rwsiteinfo --fields=sensor,id-sensor
```

Print sensor name for two sensor numbers:

```
rwsiteinfo --fields=sensor --sensor=0,1
```

Print description of a sensor:

```
rwsiteinfo --fields=describe-sensor --sensor=SEN0
```

Parameters:

Min-Name	Description	Arguments
classes	Display listed classes	string
col	Specify separation character between columns	character
data	Root of directory containing repository	filename
delim	Shortcut for no-columns no-final-del column-sep	character (opt.)
fields	List of fields to print	site-fields
flowtypes	Display listed class/type pairs	class/type
help	Print usage summary	none
help-fields	Print field descriptions	none
list-delim	Use specified character in fields list	character
no-col	Disable fixed-width columnar output	none
no-final	Suppress column delimiter after last	none
no-titles	Do not print column headers	none
pager	Program to invoke to process output	filename
sensors	Display listed sensors	int-list or name list
site-conf	Specify location of the site configuration file	filename
timestamp-format	Specify formatting of times	(see `rwcut` description)
types	Display listed types	type list
version	Print this program's version	none

Site fields:

`class` – role of sensor as configured*	`flowtype` – class/type pair*
`default-class` – default sensor role*	`id-flowtype` – integer class/type pair*
`mark-defaults` – indicate use of defaults	`id-sensor` – integer sensor ID*
`default-type` – default flow category*	`sensor` – name of sensor*
`describe-sensor` – text description of sensor	`type` – flow category*
`repo-start-date` – time of first file	`repo-end-date` – time of latest file
`repo-file-count` – number of files	

** These fields also have a :list form (e.g. class:list) that formats the entry as a comma-separated list instead of across multiple lines.*

rwset

Read binary flow records and generate one or more IP sets.

Syntax summary: (option parameters and source are optional)

```
rwset option-parameters field-parameters source
```

Examples:

Generate set from source IP addresses or records in file:

```
rwset --sip-file=src.set flows.rw
```

Generate sets with refiltering:

```
rwfilter --start=2011/04/15T00 --end=2011/04/15T07 \
  --type=out --proto=6 --pass=stdout \
| rwset --sip=tcp-src.set --copy=stdout \
| rwfilter stdin --sport=0-1023 --pass=stdout \
| rwset --sip=tcp-rsvd-src.set
```

Parameters:

Min-Name	Description	Arguments
	Option Parameters	
compress	Select compression	comp. opt.
copy	Copy all input SiLK flows to given pipe or file	filename
help	Print usage summary	none
invocation-strip	Remove command history from file header	none
ipv6-policy	Specify how to handle IPv4/v6 records	(see `rwcut` description)
note-add	Put arg in file header	string
note-file	Put contents of arg in file header	filename
note-strip	Remove note entries from file header	none
print-file	Print names of input files as they are opened	none
record-v	IP set record version for compatibility	`0`, `2`, `3`, or `4`
site-conf	Specify location of configuration file	filename
version	Print this program's version	none
xarg	Read input file names from file or pipe	filename (opt.)
	Field Parameters (at least one needed)	
any-file	Store IP set of both source and destination addresses	filename
dip-file	Store IP set of destination addresses	filename
nhip-file	Store set of flow markings	filename
sip-file	Store IP set of source addresses	filename

rwsetbuild

Read text list of IP addresses and produce binary IP set.

Syntax summary: (can use `stdin` for input and `stdout` for output, otherwise filenames)

`rwsetbuild` *`parameters input output`*

Sample Input File: (`list.set.txt` – containing 10 addresses)

```
10.1.1.1
10.2.2.2
192.168.12.0/29
```

Examples:

Generate IP set from one-address-per-line file:

```
rwsetbuild list.set.txt list.set
```

Generate IP set from file with address ranges (colon-separated):

```
rwsetbuild --ip-range=':' ranges.set.txt ranges.set
```

Produce sorted list of unique IP addresses in file:

```
rwsetbuild input.txt stdout | rwsetcat
```

Parameters: (all optional)

Min-Name	Description	Arguments
compress	Select compression	comp. opt.
help	Print usage summary	none
invocation-strip	Remove command history from file header	none
ip-ranges	Allow input of address ranges in IP or integer format (no wildcards)	character (opt.)
note-add	Put arg in file header	string
note-file	Put contents of arg in file header	filename
record-v	IP set record version for compatibility	`0`, `2`, `3`, or `4`
version	Print this program's version	none

rwsettool

Perform operations on set files to produce new set files.

Syntax summary: (operation and arg-sets are required, parameters are optional)

```
rwsettool operation arg-sets parameters
```

where *arg-sets* is a blank-delimited list of IP set file names

Examples:

Merging two sets:

```
rwsettool --union day1.set day2.set \
  --output=either.set
```

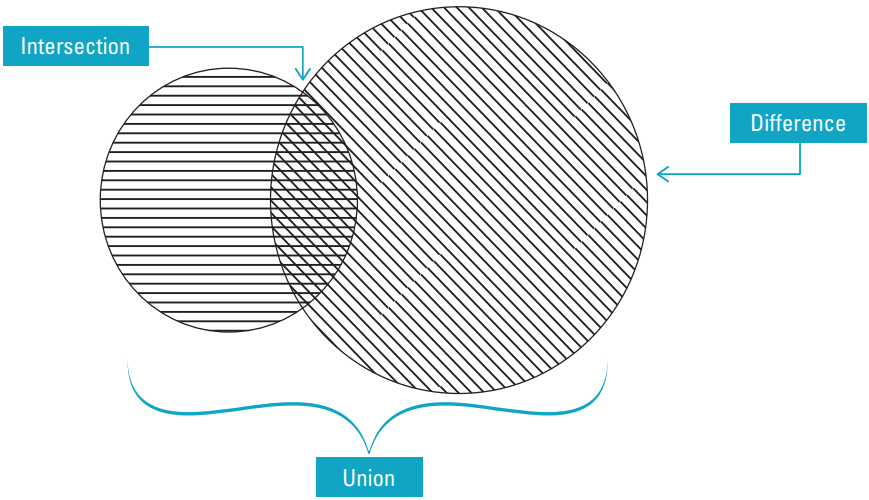
Finding common elements:

```
rwsettool --intersect day1.set day2.set \
  --output=both.set
```

Finding non-common elements:

```
rwsettool --diff day1.set day2.set --output=only1.set
rwsettool --diff day2.set day1.set \
| rwsettool --union stdin only1.set --output=not-comm.set
```

Set operations:



`rwsettool` Parameters

Min-Name	Description	Arguments
	Operations	
diff	Gathers IPs from first input not in other input sets	none
intersect	Gathers IPs that exist in ALL the input sets	none
mask	Only one IP per block of size in Arg	integer
sample	Only random samples from input; need size or ratio	none
union	Gathers IPs that exist in ANY input set	none
	Option Parameters	
compress	Compression for output	comp. opt.
fill	Use complete blocks of given prefix length	integer
help	Print usage summary	none
invocation-strip	Remove command history from file header	none
note-add	Use Arg as annotation	string
note-file	Use Arg contents as annotation	filename
note-strip	Do not copy notes from the input to the output	none
output	Specify output location	filename
ratio	The probability that an individual IP will be sampled	0.0-1.0
record-v	IP set record version for compatibility	`0`, `2`, `3`, or `4`
seed	Random number seed for `--sample`	decimal
size	The sample size for `--sample` per input	integer
version	Print this program's version	none

rwsetcat

Read binary IP set and produce text.

Syntax summary: (all optional)

```
rwsetcat parameters filename ...
```

Examples:

List IP addresses in set into text file:

```
rwsetcat list.set >list.set.txt
```

Count IP addresses from standard input:

```
rwsetcat --count
```

Summarize CIDR /16 blocks (class B subnets) in set:

```
rwsetcat --net=B list.set
```

List IP addresses in set as integers (for plotting):

```
rwsetcat --ip-format=decimal list.set
```

Parameters: (also format parameters as `rwcut` takes, but no plugin, pmap, or python)

Min-Name	Description	Arguments
cidr	Print IPs in CIDR block notation	zero or one (opt.)
count	Print the number of IPs	none
help	Print usage summary	none
ip-range	Print IPs as ranges of count|low|high|	none
net	Summarize CIDR blocks in set	ProtoDispSumm (opt.)
	Proto is `v4`: (uses `asv4` policy), `v6:` (uses `force` policy), or blank (counts as `v4:`)	
	Disp is a comma-separated list of CIDR lengths, letters, or blank (counts as `H`)	
	For v4, letters are `T`=0, `A`=8, `B`=16, `C`=24, `X`=27, or `H`=32	
	For v6, letters are `T`=0, or `H`=128	
	Summ is blank (for no summary) or a combination of	
	S – use later list or defaults for v4, `A,B,C,X,` for v6, `48,64`	
	/ - (if no later list) previous Disp as summary blocks	
	/list – specify another list in Disp format as summary blocks	
pager	Program to invoke to process output	filename
print-file	Print filename as they are processed	zero or one (opt.)
print-ips	Also print IPs if count or statistics parameter present	none
print-stat	Print set statistics	none
version	Print this program's version	none

rwsort

Sort binary flow records, merging files if required.

Syntax summary: (parameters and flow-files are optional, `fields` is required)
`rwsort --fields=key-fields` *`parameters filename ...`*

Examples:

Ordering flow file by start time:

```
rwsort --fields=stime flows.rw >sorted.rw
```

Ordering flow file by source IP address, then by time:

```
rwsort --fields=sip,stime --output=src-time.rw flows.rw
```

Merging two flow files and ordering by start time:

```
rwsort --fields=stime one.rw two.rw >time-order.rw
```

Parameters:

Min-Name	Description	Arguments
compress	Compression for output	comp. opt.
help	Print usage summary	none
help-fields	Print field descriptions	none
note-add	Store Arg as an annotation	string
note-file	Store contents of Arg as an annotation	filename
output	Output destination	filename
plugin	Load given plugin(s) to add fields	filename
pmap-file	Prefix map file to read. Use before `fields`	map:filename
presort	Merge only (do not sort)	none
print-file	Print names of input files	none
python	Use Python code to extend processing	filename
reverse	Reverse the sort order	none
site-conf	Site configuration file	filename
sort-buff	Memory allocation for sort buffer	integer[k,M,G]
temp	Store temporary files here	dirname
version	Print this program's version	none
xarg	Read input file names from file or pipe	filename (opt.)

rwcount

Summarize binary flow records across time.

Syntax summary: (all parameters optional)

```
rwcount parameters filename ...
```

Examples:

Generate 30-second counts of records from standard input, with data proportional to time:

```
rwcount >30-sec.txt
```

Generate five-minute counts from file, with data proportional to time:

```
rwcount --bin-size=300 flows.rw >five-min.txt
```

Generate hourly counts in csv format, with data only in start time block, from file (including `sed` command to add space after comma):

```
rwcount --bin-size=3600 --delim=',' \
  --load-scheme=1  flows.rw \
| sed –e 's/,/, /g'  >hr.csv
```

Generate 12-minute counts (calculated in line), with data proportional to time:

```
rwcount --bin-size=$((12*60)) flows.rw >12-min.txt
```

Common bin-size values:

Interval	bin-size Value
5 min	300
10 min	600
15 min	900
30 min	1800
Hour	3600
Day	86400
Week	604800

`rwcount` Parameters

(Also format parameters as rwcut takes, but no plugin, pmap, or python)

Min-Name	Description	Arguments
bin-size	Size of bins in seconds (default 30.000)	decimal
bin-slots	Print bin labels using the internal bin index	none
copy	Copy all input SiLK flows to given pipe or file	filename
end	Print bins until this time	time
epoch	Print bin labels using epoch time	none
help	Print usage summary	none
load	Specifies handling of flows that span bins `0` – split volume EVENLY across the bins `1` – fill FIRST appropriate bin with complete volume `2` – fill LAST appropriate bin with complete volume `3` – fill CENTERMOST bin with complete volume `4` – split volume into bins proportional to time ACTIVE `5` – assign MAXIMUM possible volume for each bin `6` – assign MINIMUM possible volume for each bin (for `5` and `6`, sum of all bin values may not match total volume)	integer
output	Send output to given file path (default stdout)	filename
pager	Program to invoke to process output	filename
print-file	Print names of input files as they are opened	none
site-conf	Specify location of the site configuration file	filename
skip-zero	Don't print bins that have no flows	none
start	Print bins from this time forward	time
version	Print this program's version	none
xarg	Read input file names from file or pipe	filename (opt.)

rwstats

Generate top N, bottom N, or descriptive statistics from file.

Syntax summary: (two alternative forms)

Generate descriptive statistics: (overall or by protocol, omit filename to use `stdin`)

```
rwstats --overall filename ...

rwstats --detail=protocol-list filename ...
```

Generate top/bottom N lists:

```
rwstats --fields=fieldlist --values=vallist --top bound
  options filename ...

rwstats --fields=fieldlist --values=vallist --bottom bound
  options filename ...
```

where *vallist* is a comma-separated list of bytes, packets, flows, records, sip-distinct, dip-distinct or distinct:*KEYFIELD*

Examples:

Find 10 highest-volume IP pairs:

```
rwstats --fields=sip,dip --values=bytes --top \
  --count=10 flows.rw
```

Find all destination ports that get more than ten percent of the traffic by frequency:

```
rwstats --fields=dport --values=records --top \
  --percent=10 flows.rw
```

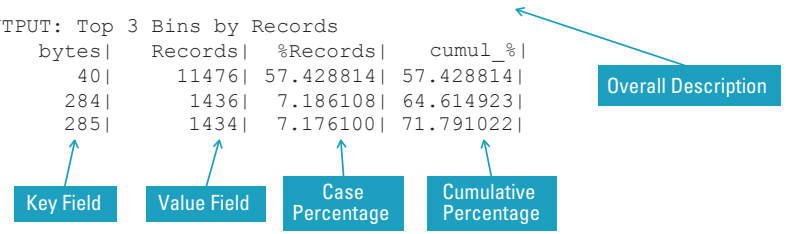
Print descriptive statistics on traffic volumes:

```
rwstats --overall flows.rw
```

Output sample:

```
rwstats --fields=bytes --values=records --top --count=3 maybe.rw

INPUT: 19983 Records for 18 Bins and 19983 Total Records
OUTPUT: Top 3 Bins by Records
     bytes|   Records|  %Records|    cumul_%|
        40|     11476| 57.428814| 57.428814|
       284|      1436|  7.186108| 64.614923|
       285|      1434|  7.176100| 71.791022|
```



`rwstats` Parameters

(Also all formatting parameters from `rwcut`*)*

Min-Name	Description	Arguments
	Bounds	
bottom	Apply bounds from key with lowest value	none
count	Specify N (0 means print all)	integer
percent	Specify percent for bound. Only for bytes, packets, or flows	decimal
threshold	Specify value for bound (not from plugins)	integer
top	Apply bounds from key with highest value	none
	Options	
bin-time	Specify bin size for time keys	integer
copy	Copy all input SiLK flows to given pipe or file	filename
help	Print usage summary	none
help-fields	Print field descriptions	none
ipv6-policy	Specify how to handle IPv4/v6	(see `rwcut` description)
legacy-help	Print help for legacy switches	none
no-per	Don't print the percentage columns	none
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
presort	Assume input has been presorted for fields	none
print-file	Print names of input files	none
site-conf	Specify location of the site configuration file	filename
temp	Store temporary files in this directory	dirname
version	Print this program's version	none
xarg	Read input file names from file or pipe	filename (opt.)

rwuniq

Summarize traffic volumes based on unique combinations of flow record fields.

Syntax summary: (options and filename are optional; values may be replaced by counting parameters)

```
rwuniq --fields=fieldlist --values=vallist options filename ...
```

Examples:

Generate byte count totals of protocols grouped by hour from a file:

```
rwuniq --fields=proto,stime --bin-time=3600 \
  --values=bytes flows.rw
```

Generate byte count totals and number of source addresses of high-volume flows by destination ports from a file:

```
rwuniq --fields=dport --values=bytes,distinct:sip \
  --bytes=10000- maybe.rw
```

Generate contrasting views of traffic by size and by source port:

```
rwuniq --fields=bytes --values=records,distinct:dip \
  --output=bytes.txt --copy=stdout flows.rw \
| rwuniq --fields=sport --values=records,distinct:dip \
  --output=sport.txt
```

Count source ports per source address:

```
rwuniq --fields=sip --values=distinct:sport flows.rw
```

Count bytes, packets, and flow records by protocol, reporting in protocol number order:

```
rwuniq --fields=protocol \
  --values=bytes,packets,flows --sort flows.rw
```

Output sample:

```
rwuniq --fields=dport --values=bytes,distinct:sip maybe.rw
```

```
dPort|                Bytes|sIP-Distin|
   22|              2492768|         1|
 7051|               636478|         1|
 7052|               635862|         1|
```

Key Field

Value Fields

`rwuniq` Parameters

(Also all formatting parameters from `rwcut`*)*

Min-Name	Description	Arguments
	Option Parameters	
bin-time	Specify bin size for time keys	integer
copy	Copy all input SiLK flows to given pipe or file	filename
epoch-time	Print times as count of seconds since epoch	none
fields	Field combination for bins	fieldlist
help	Print usage summary	none
help-fields	Print field descriptions	none
ipv6-policy	Specify how to handle IPv4/v6	(see `rwcut` description)
output	Send output to given file path	filename
pager	Program to invoke to process output	filename
presort	Assume input has been presorted with fields	none
print-file	Print names of input files as they are opened	none
site-conf	Specify location of the site configuration file	filename
sort-out	Present the output sorted by key fields	none
temp	Store temporary files here	dirname
version	Print this program's version	none
	Counting Parameters	
all	Bytes, packets, flows, stime, and etime	none
bytes	Sum bytes in each bin	int-range
dip-dist	Count distinct destination addresses in each bin	int-range
flows	Count flow records in each bin	int-range
packets	Sum packets in each bin	int-range
sip-dist	Count distinct source addresses in each bin	int-range
values	Value(s) to compute: `bytes`, `packets`, `records`, `distinct:`*`KEYFIELD`*`, stime-earliest, flows, etime-latest`	valuelist
xarg	Read input file names from file or pipe	filename (opt.)

Notes

IP Protocols

Num	Name	Description	Header bytes	
			IPv4	IPv6
0	HOPOPT	IPv6 Hop-by-Hop		48
1	ICMP	Internet Control Messages	28	
2	IGMP	Internet Group Management	28	
3	GGP	Gateway-to-Gateway	24	
4	IPv4	v4 Encapsulation	40	60
6	TCP	Transmission Control	40	60
8	EGP	Exterior Gateway	30	50
9	IGRP	Interior Gateway	28	
17	UDP	User Datagram	28	48
27	RDP	Reliable Data	38	58
28	IRTP	Internet Reliable Transaction	28	48
41	IPv6	IPv6 Encapsulation	60	80
43	IPv6-Route	IPv6 Routing Header		48
44	IPv6-Frag	IPv6 Fragment Header		48
46	RSVP	Reservation Protocol	28	48
47	GRE	Generic Route Encapsulation	24	44
50	ESP	Encap Security Payload	28	48
51	AH	Authentication Header	32	52
53	SWIPE	IP with Encryption	28	48
58	ICMP	ICMP for IPv6		44
59	NoNxt	No Next Header for IPv6		40
60	IPv6-Opts	Destination Options for IPv6		48
88	EIGRP	Enhanced Interior Gateway Routing	40	60
98	ENCAP	Encapsulation Header	28	48
99		Private Encryption	20	40
132	SCTP	Stream Control Transmission	32	52
143-252		*Unassigned*	--	
253-254		*Experimental*	--	
255		*Reserved*	--	

SiLK Commands (continued)

Text Output SiLK Tools

page — Tool summary *(tools without page numbers are not in guide)*

- rwbagcat – display and characterize bag content
- rwcompare – determine if two flow files are identical
- 18 rwcount – time-series counts
- 8 rwcut – text from flows
- rwfglob – list repository files from rwfilter selection parameters
- 10 rwfileinfo – describe file contents
- rwpcut – display packet fields of PCAP data
- rwpmapcat – display pmap content
- rwpmaplookup – display pmap label for IP addresses
- rwresolve – perform DNS lookup from IP address text
- rwscan – apply scan detection models to flows
- rwscanquery – query the network scan database
- 16 rwsetcat – display IP set content
- rwsetmember – determine which IP sets have this address
- 11 rwsiteinfo – display repository information as configured
- 20 rwstats – generate top N/bottom N counts or protocol statistics
- 22 rwuniq – generate aggregate counts

(See front cover for binary output tools)

For More Information

http://tools.netsa.cert.org/silk/docs.html

Analyst's Handbook: Using SiLK for Network Traffic Analysis - tutorial on the SiLK tools and on using them for analyzing network traffic

PySiLK: SiLK in Python - reference guide for manipulating SiLK Flow data from within Python

The SiLK Reference Guide - every SiLK manual page in a single document

SiLK Installation Handbook - instructions on configuring, building, and installing SiLK at your site