Apex Filters 17.5.1.0

User Guide

1 Dec 2018

Table of Contents

1

Chapter 1: Apex filters

Use filters on dashboards or widgets to isolate just the object of interest for you, such as a specific application, IP or MAC address, or telephone number in a VoIP conversation.

Conventions used in this document:

Filters typed by the user will be shown in Bold Courier

- Filter names are written in `red`.
- Angled brackets < and > surround required arguments.
- Square brackets [and] surround optional arguments.
- Choices are separated by a vertical bar |.
- Single quotes ' ' are required for any parameter or text containing a space.
- Filter names are case sensitive (for instance, `fixRequest` or `networkType`).
- Parameters are case-*in*sensitive.

Boolean operators used in filter expressions

Use Boolean operators to build complex filters that find exactly what you are looking for in your dashboard.

These words or characters can be used to build very complex expressions. Filters containing multiple operators are evaluated using the order shown below. In other words, everything inside of parentheses is evaluated first starting from the innermost group and moving outwards, then any NOT operators are considered, then AND operators, and finally OR operators. To ensure that an expression containing multiple operators works as you intend, use parentheses to group items together.

( )	Used to group words or phrases and to show the order in which the groups should be evaluated (innermost group first). Example: ip 192.168.1.15 and app HTTP Example: ((ip 192.168.1.15 and app HTTP) or (ip 192.168.1.15 and app HTTPS)) and ((browser 'Chrome') and not (browser 'MSIE' or browser 'Firefox'))
not or **!**	Show only items that do *not* meet the expression. Example: not app http Example: ! app http
and or **&&**	Show only items that contain *all* of the expression items. Example: ip 192.168.1.15 and app HTTP and browser 'Chrome' Example: ip 192.168.1.15 && app HTTP && browser 'Chrome'
or or **||**	Show only items that contain *any* of the expression items. Example: browser 'MSIE' or browser 'Firefox' Example: browser 'MSIE' || browser 'Firefox'

app

Application or derived application being transported across the network.

If you created a derived application in Observer Analyzer that you want to filter on, enclose the name of the application in single quotes.

Usage

```
app  < 'AFP over TCP' |  'AmEx ISO 8583' |  'ArcaBook Multicast'
|  'BFD Control' |  'BFD Echo' |  'BGMP' |  'BGP' |  'Bit
Torrent' |  'CIFS/SMB' |  'Cisco SCCP' |  'Cisco SGM' |  'Cisco
Wireless IDS' |  'Cisco-RSB' |  'Citrix CGP' |  'Citrix ICA' |
'CME RLC' |  'DAAP' |  'DCE endpoint' |  'Derived Application'
|  'DIAMETER' |  'DirectEdge Multicast' |  'DNP3' |  'DNS' |
'DNS-LLMNR' |  'DNS-multicast' |  'EIP' |  'FCIP' |  'Finger' |
'FIX' |  'FLAP' |  'FTP control' |  'FTP data' |  'FTPS control'
|  'FTPS data' |  'GIOP' |  'H.245' |  'H.248' |  'H.323/H.225
gate disc' |  'H.323/H.225 gate stat' |  'H.323/H.225 host call'
|  'HOST2' |  'HTTP' |  'HTTPS' |  'IAX2' |  'IMAP' |  'IPFIX'
|  'IRC' |  'ISAKMP' |  'ISCSI' |  'ISO_TSAP' |  'Kerberos' |
'LBT-TCP' |  'LDAP' |  'LDP' |  'LDP/LDR' |  'Lotus Notes' |
'LSE Infolect' |  'M3UA' |  'MEGACO H.248' |  'MGCP call agent'
|  'MGCP gateway' |  'MINET' |  'MODBUS' |  'MS Web Discovery'
|  'MSN' |  'MSRPC' |  'Multicast Pitch' |  'MySQL' |  'NCP' |
'NetBIOS name' |  'NetBIOS session' |  'NetFlow' |  'NI Observer'
|  'NI OI' |  'NI Probe' |  'NNTP' |  'NNTP over SSL/TLS' |
'Other' |  'OUCH' |  'POP3' |  'POP3 over SSL/TLS' |  'PPTP'
|  'PTP' |  'RADIUS' |  'RLogin' |  'RTSP' |  'Shoutcast' |
'SIP' |  'SIP over SSL/TLS' |  'SOCKS' |  'SSH' |  'SUNRPC' |
'SUNRPC-RQuota' |  'T.120/X.224' |  'T.38 FAX (1998)' |  'T.38
FAX (2002)' |  'TACACS' |  'TDP' |  'TDS-SQL' |  'Telnet' |
'Telnet over SSL/TLS' |  'TNS-Oracle' |  'VISA Cardnet' |  'WAP-
WSP' |  'WAP-WTP' |  'WebSphere MQ' |  'WhoIs' |  'WhoIs++' |
'WMP-NSS' |  'X-Windows' |  'XFER' > [ /TCP |  /UDP |  /SCTP ]
```

Examples

- app HTTP
- app HTTP/TCP
- app 'WebSphere MQ'
- app 'WebSphere MQ/TCP'

browser

Brand of the web browser being used.

Usage

```
browser  <Avant Browser | Chrome | Comodo_Dragon | Deepnet Explorer | Firefox | GreenBrowser | Iron | K-Meleon | Konqueror | Maxthon | Mozilla | MSIE | Opera | RockMelt | Safari | SeaMonkey | SlimBrowser>
```

Examples

- `browser` Chrome
- `browser` Safari
- `browser` 'Deepnet Explorer'

browserPlatform

Type of device on which the web browser was used.

Usage

`browserPlatform <other | desktop | mobile>`

Examples

- `browserPlatform` desktop
- `browserPlatform` mobile
- `browserPlatform` other

browserVersion

Version of the web browser.

Usage

```
browserVersion  <Any text>
```

Examples

- `browserVersion` 36.0.1985.125
- `browserVersion` 11.0

dataSource

Business group or probe instance name. This is required if you want to drill into GigaStor data.

Usage

```
dataSource  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `dataSource` ProbeInstance1
- `dataSource` 'Probe Instance 1'

ethertype

Layer 3 protocol encapsulated in the first two octets of an Ethernet frame.

Usage

```
ethertype  <'3com loop detection' |  '3com tcp-ip' |  '3com
xns sys mgmt' |  '3com1' |  '3com2' |  '3com3' |  '802 local
exp' |  '802 oui extended' |  '802.11i' |  '802.11r frrr' |
'802.1ab lldp' |  '802.1ae macs' |  '802.1q mmrp' |  '802.1q
mvrp' |  '802.1q s-tag' |  '802.1qbe i-sid' |  '802.1qbg ecp'
|  '802.21 mihp' |  '802.3 epon' |  '802.3 slow protocol' |
'advanced encryption' |  'aeonic systems' |  'allen bradley'
|  'allied telesyn' |  'alpha micro' |  'apollo computer' |
'apollo domain' |  'appletalk' |  'appletalk aarp' |  'applitek
corp' |  'apricot comp' |  'arai bunkichi' |  'arp' |  'artisoft'
|  'ascom banking' |  'athena prog' |  'atomic' |  'att0' |
'att1' |  'att2' |  'att3' |  'autophon' |  'axis communications'
|  'banyan systems' |  'banyan vines' |  'banyan vines echo'
|  'banyan vines loopb' |  'bbn simnet' |  'beacon probe' |
'berkeley trailer' |  'biin' |  'bridge comm' |  'cabletron' |
'chaosnet' |  'charles river' |  'charles river data' |  'cisco
discovery' |  'cisco dtp' |  'cisco pagp' |  'cisco pvstp' |
'cisco stp' |  'cisco udld' |  'cisco vlan bridge' |  'cisco
vtp' |  'comdesign' |  'compr. rtp setup' |  'compressed rtp' |
'comptgraphiccorp' |  'computer network' |  'computer protocol'
|  'comsat labs' |  'control tech' |  'counterpoint comp1' |
'counterpoint comp2' |  'cray comm' |  'cronus direct' |  'cronus
vln' |  'dansk data elektro' |  'datability' |  'datability2'
|  'dca' |  'dca multicast' |  'dec customer' |  'dec decnet iv
route' |  'dec diag' |  'dec lan monitor' |  'dec lanbridge' |
'dec lat' |  'dec lavc sca' |  'dec mop dump load' |  'dec mop
remote con' |  'dec unassigned' |  'dec unassigned2' |  'dec
unassigned3' |  'dec unassigned4' |  'dec unassigned5' |  'delta
control' |  'dlog' |  'eapol' |  'ecma' |  'evans sutherland'
|  'excelan' |  'experdata' |  'foundry networks' |  'gateway
comm' |  'general dynamics' |  'geonetworking' |  'gsmp' |
'harris corp' |  'hayes' |  'hippi 6400' |  'hippi fp encap' |
'hp' |  'hp probe' |  'hypercom network sy' |  'idea courier'
|  'inst ind info' |  'intergraph corp' |  'invisible software'
|  'ip autonomous' |  'ipv4' |  'ipv6' |  'japan computer indu'
|  'kinetics' |  'kti' |  'l2-is-is' |  'lanbridge cache' |
'landis gyp power' |  'landmark graphics' |  'lansoft 0' |
'lansoft 1-5' |  'little machines' |  'logicraft' |  'lrt' |
'marathon tech' |  'matra' |  'mcap' |  'merit internodal' |
'motorola' |  'motorola computer' |  'mpls' |  'mpls multicast'
|  'nbs' |  'nestar' |  'network computing' |  'nixdorf' |
'nixdorf computers' |  'norand' |  'nortel slpp' |  'novell'
|  'os9 microware' |  'pacer software' |  'pcs basic block'
|  'phoenix microsys' |  'planning research ' |  'polygon' |
'ppp' |  'ppp ccp' |  'ppp chap' |  'ppp ipcp' |  'ppp lcp' |
'ppp lqr' |  'ppp mlcp' |  'ppp pap' |  'ppp spcp' |  'pppoe
discovery' |  'pppoe session' |  'proteon' |  'protocol engines'
```

```
| 'protocolengines xtp' | 'pupaddrtrans0' | 'pupaddrtrans1' |
'qualcomm1' | 'qualcomm2' | 'quantum software' | 'rad network
devices' | 'rarp' | 'rational' | 'raw frame relay' | 'realtek
rrcp' | 'retix' | 'rosemount corp' | 'saic' | 'sectra'
| 'secure data' | 'sgi' | 'sgi/time warner' | 'siemens
gammasonic' | 'silicon graphics' | 'sna over ethernet' | 'snmp
over ethernet' | 'spider systems ltd' | 'stanford v kernel'
| 'stp hippi st' | 'symbol tech' | 'symbolics private1' |
'symbolics private2' | 'taurus controls' | 'taylor instrument'
| 'tcp ip compression' | 'technically elite' | 'tigan inc'
| 'transetherbridging' | 'trfs' | 'trill' | 'trill fgl'
| 'trill rbridge chan' | 'tymshare' | 'ungerman bass net'
| 'ungermann bass dia' | 'ungermann bass dwn' | 'ungermann
bass niu' | 'univ mass amherst' | 'univ of utah' | 'valid
systems' | 'valis' | 'varian assoc' | 'veeco integ auto'
| 'vg analytical' | 'vg laboratory' | 'vita link' | 'vmtp'
| 'walker richer' | 'walker richer quin' | 'wellfleet comm'
| 'x.25 level 3' | 'x.75' | 'xerox ieee802.3 pup' | 'xerox
loopback' | 'xerox ns idp' | 'xerox pup' | 'xerox pup call'
| 'xns compatability' | 'xns frame relay arp' | 'xyplex1' |
'xyplex2' >
```

Examples

- `ethertype` 'cisco discovery'
- `ethertype` polygon
- `ethertype` 'cisco vlan bridge'

fixRequest

Request in the financial trading FIX protocol.

Usage

```
fixRequest  <'adjusted position report' |  'advertisement' |
'allocation' |  'allocation instruction ack' |  'allocation
instruction alert' |  'allocation report ack' |  'allocation
response' |  'application message report' |  'application message
request' |  'application message request ack' |  'assignment
report' |  'bid request' |  'bid response' |  'business message
reject' |  'ch' |  'ci' |  'collateral assignment' |  'collateral
inquiry' |  'collateral inquiry ack' |  'collateral report' |
'collateral request' |  'collateral response' |  'confirmation'
|  'confirmation ack' |  'confirmation request' |  'contrary
intention report' |  'cross order cancel request' |  'cross
order cancel/replace request' |  'derivative security list' |
'derivative security list request' |  'derivative security list
update report' |  'don't know trade' |  'email' |  'execution
ack' |  'execution report' |  'execution report-calculated'
|  'execution report-canceled' |  'execution report-done for
day' |  'execution report-expired' |  'execution report-fill,
replaced' |  'execution report-new' |  'execution report-
order status' |  'execution report-partial fill, replaced' |
'execution report-pending cancel' |  'execution report-pending
new' |  'execution report-pending replace' |  'execution report-
rejected' |  'execution report-replace' |  'execution report-
restated' |  'execution report-stopped' |  'execution report-
suspended' |  'execution report-trade cancel' |  'execution
report-trade correct' |  'execution report-trade, partial fill
or fill' |  'heartbeat' |  'indication of interest' |  'list
cancel request' |  'list execute' |  'list status' |  'list
status request' |  'list strike price' |  'logon' |  'logout'
|  'market data, Incremental refresh' |  'market data, request'
|  'market data, request reject' |  'market data, Snapshot/Full
Refresh' |  'market definition' |  'market definition request'
|  'market definition update report' |  'mass quote' |  'mass
quote acknowledgement' |  'multileg order cancel/replace' |
'network status request' |  'network status response' |  'new
order, cross' |  'new order, multileg' |  'news' |  'order
cancel reject' |  'order mass action report' |  'order mass
action request' |  'order mass cancel report' |  'order mass
cancel request' |  'order mass status request' |  'order, cancel
request' |  'order, cancel/replace request' |  'order, list' |
'order, single' |  'order, status request' |  'party details list
report' |  'party details list request' |  'position maintenance
report' |  'position maintenance request' |  'position report' |
'quote' |  'quote cancel' |  'quote request' |  'quote request
reject' |  'quote response' |  'quote status report' |  'quote
status request' |  'registration instructions' |  'registration
instructions response' |  'reject' |  'request for positions' |
'request for positions ack' |  'resend request' |  'rfq request'
```

```
| 'security definition' | 'security definition request'
| 'security definition update report' | 'security list' |
'security list request' | 'security list update report' |
'security status' | 'security status request' | 'security type
request' | 'security types' | 'sequence reset' | 'settlement
instruction request' | 'settlement instructions' | 'settlement
obligation report' | 'stream assignment report' | 'stream
assignment report ack' | 'stream assignment request' | 'test
request' | 'trade capture report' | 'trade capture report
ack' | 'trade capture report request' | 'trade capture report
request ack' | 'trading session list' | 'trading session list
request' | 'trading session list update report' | 'trading
session status' | 'trading session status request' | 'user
notification' | 'user request' | 'user response' | 'xml
message'>
```

Examples

- `fixRequest` 'bid request'
- `fixRequest` 'control assignment'
- `fixRequest` 'order, cancel/replace request'

fixResponse

Response in the financial trading FIX protocol.

Usage

```
fixResponse  <'adjusted position report' |  'advertisement'
|  'allocation' |  'allocation instruction ack' |  'allocation
instruction alert' |  'allocation report ack' |  'allocation
response' |  'application message report' |  'application message
request' |  'application message request ack' |  'assignment
report' |  'bid request' |  'bid response' |  'business message
reject' |  'ch' |  'ci' |  'collateral assignment' |  'collateral
inquiry' |  'collateral inquiry ack' |  'collateral report' |
'collateral request' |  'collateral response' |  'confirmation'
|  'confirmation ack' |  'confirmation request' |  'contrary
intention report' |  'cross order cancel request' |  'cross
order cancel/replace request' |  'derivative security list' |
'derivative security list request' |  'derivative security list
update report' |  'don't know trade' |  'email' |  'execution
ack' |  'execution report' |  'execution report-calculated'
|  'execution report-canceled' |  'execution report-done for
day' |  'execution report-expired' |  'execution report-fill,
replaced' |  'execution report-new' |  'execution report-
order status' |  'execution report-partial fill, replaced' |
'execution report-pending cancel' |  'execution report-pending
new' |  'execution report-pending replace' |  'execution report-
rejected' |  'execution report-replace' |  'execution report-
restated' |  'execution report-stopped' |  'execution report-
suspended' |  'execution report-trade cancel' |  'execution
report-trade correct' |  'execution report-trade, partial fill
or fill' |  'heartbeat' |  'indication of interest' |  'list
cancel request' |  'list execute' |  'list status' |  'list
status request' |  'list strike price' |  'logon' |  'logout'
|  'market data, Incremental refresh' |  'market data, request'
|  'market data, request reject' |  'market data, Snapshot/Full
Refresh' |  'market definition' |  'market definition request'
|  'market definition update report' |  'mass quote' |  'mass
quote acknowledgement' |  'multileg order cancel/replace' |
'network status request' |  'network status response' |  'new
order, cross' |  'new order, multileg' |  'news' |  'order
cancel reject' |  'order mass action report' |  'order mass
action request' |  'order mass cancel report' |  'order mass
cancel request' |  'order mass status request' |  'order, cancel
request' |  'order, cancel/replace request' |  'order, list' |
'order, single' |  'order, status request' |  'party details list
report' |  'party details list request' |  'position maintenance
report' |  'position maintenance request' |  'position report' |
'quote' |  'quote cancel' |  'quote request' |  'quote request
reject' |  'quote response' |  'quote status report' |  'quote
status request' |  'registration instructions' |  'registration
instructions response' |  'reject' |  'request for positions' |
'request for positions ack' |  'resend request' |  'rfq request'
```

```
| 'security definition' | 'security definition request'
| 'security definition update report' | 'security list' |
'security list request' | 'security list update report' |
'security status' | 'security status request' | 'security type
request' | 'security types' | 'sequence reset' | 'settlement
instruction request' | 'settlement instructions' | 'settlement
obligation report' | 'stream assignment report' | 'stream
assignment report ack' | 'stream assignment request' | 'test
request' | 'trade capture report' | 'trade capture report
ack' | 'trade capture report request' | 'trade capture report
request ack' | 'trading session list' | 'trading session list
request' | 'trading session list update report' | 'trading
session status' | 'trading session status request' | 'user
notification' | 'user request' | 'user response' | 'xml
message'>
```

Examples

- `fixResponse` 'bid request'
- `fixResponse` 'control assignment'
- `fixResponse` 'order, cancel/replace request'

flowIdx

NetFlow interface.

Usage

```
flowIdx  <Unlimited number of digits (0-9).>
```

Examples

- `flowIdx` 1
- `flowIdx` 22
- `flowIdx` 333
- `flowIdx` 1234567890

flowIdxClient

NetFlow Client interface when using the **NetFlow Interface Index - Client** key.

The `flowIdxClient` filter displays the matching NetFlow Index only if the system made the request. The filter does not work with the **NetFlow Interface Index - Pair** key even though **NetFlow Interface Index - Pair** shows the client/server NetFlow Indexes in the conversation.

Usage

`flowIdxClient` *<Unlimited number of digits (0-9).>*

Examples

- `flowIdxClient` 1
- `flowIdxClient` 22
- `flowIdxClient` 333
- `flowIdxClient` 1234567890

flowIdxServer

NetFlow Server interface when using the **NetFlow Interface Index - Server** key.

The `flowIdxServer` filter displays the matching NetFlow Index only if the system received the request. The filter does not work with the **NetFlow Interface Index - Pair** key even though **NetFlow Interface Index - Pair** shows the client/server NetFlow Indexes in the conversation.

Usage

`flowIdxServer` *<Unlimited number of digits (0-9).>*

Examples

- `flowIdxServer` 1
- `flowIdxServer` 22
- `flowIdxServer` 333
- `flowIdxServer` 1234567890

flowSystem

NetFlow Autonomous System.

Usage

```
flowSystem  <Unlimited number of digits (0-9).>
```

Examples

- `flowSystem` 1
- `flowSystem` 22
- `flowSystem` 333
- `flowSystem` 1234567890

flowSystemClient

Client of a NetFlow Autonomous System when using the **NetFlow Autonomous System - Client** key.

The `flowSystemClient` filter displays the matching NetFlow Autonomous System only if the system made the request. The filter does not work with the **NetFlow Autonomous System - Pair** key even though **NetFlow Autonomous System - Pair** shows the client/server NetFlow Indexes in the conversation.

Usage

`flowSystemClient` *<Unlimited number of digits (0-9).>*

Examples

- `flowSystemClient` 1
- `flowSystemClient` 22
- `flowSystemClient` 333
- `flowSystemClient` 1234567890

flowSystemServer

Server of a NetFlow Autonomous System when using the **NetFlow Autonomous System - Server** key.

The `flowSystemServer` filter displays the matching NetFlow Autonomous System only if the system received the request. The filter does not work with the **NetFlow Autonomous System - Pair** key even though **NetFlow Autonomous System - Pair** shows the client/server NetFlow Indexes in the conversation.

Usage

`flowSystemServer` *<Unlimited number of digits (0-9).>*

Examples

- `flowSystemServer` 1
- `flowSystemServer` 22
- `flowSystemServer` 333
- `flowSystemServer` 1234567890

ip

IPv4 or IPv6 address or range of addresses.

To filter an IP range or subnet, enter the first and last IP addresses in the range separated by a hyphen.

Usage

```
ip  <IPv4 or IPv6 address.>
```

Examples

- `ip` 192.168.1.15
- `ip` 10.1.54.212
- '`ip` 192.168.20.0-192.168.20.31'
- `ip` FE80:0000:0000:0000:0000:B4EF:EF8F:4819
- `ip` FE80::B4EF:EF8F:4819

ipClient

IPv4 or IPv6 address (or range of addresses) of a client if you are using the **IP - Client** key.

The `ipClient` filter displays the matching IP only if the request came from that IP address. The filter does work with the **IP - Pair** key field even though **IP - Pair** shows the client/server IP addresses in the conversation.

To filter an IP range or subnet, enter the first and last IP addresses in the range separated by a hyphen.

Usage

```
ipClient  <IPv4 or IPv6 address.>
```

Examples

- `ipClient` 192.168.1.15
- `ipClient` 10.1.54.212
- '`ip` 192.168.20.0-192.168.20.31'
- `ipClient` FE80:0000:0000:0000:0000:B4EF:EF8F:4819
- `ipClient` FE80::B4EF:EF8F:4819

ipProto

IP-based protocol that works with the **Application** and **Protocol** keys.

Usage

```
ipProto <'3pc' | 'a/n' | 'ah' | 'argus' | 'aris' | 'ax.25' | 'bbn-rcc-mon' | 'bna' | 'br-sat-mon' | 'cbt' | 'cftp' | 'chaos' | 'compaq-peer' | 'cphb' | 'cpnx' | 'crtp' | 'crudp' | 'dccp' | 'dcn-meas' | 'ddp' | 'ddx' | 'dgp' | 'dsr' | 'egp' | 'eigrp' | 'emcon' | 'encap' | 'esp' | 'etherip' | 'fc' | 'fire' | 'ggp' | 'gmtp' | 'gre' | 'hip' | 'hmp' | 'hopopt' | 'iatp' | 'icmp' | 'idpr' | 'idpr-cmtp' | 'idrp' | 'ifmp' | 'igmp' | 'igp' | 'il' | 'i-nlsp' | 'ipcomp' | 'ipcv' | 'ipip' | 'iplt' | 'ippc' | 'iptm' | 'ipv4' | 'ipv6' | 'ipv6-frag' | 'ipv6-icmp' | 'ipv6-nonxt' | 'ipv6-opts' | 'ipv6-route' | 'ipx-in-ip' | 'irtp' | 'isis over ipv4' | 'iso-ip' | 'iso-tp4' | 'kryptolan' | 'l2tp' | 'larp' | 'leaf-1' | 'leaf-2' | 'manet' | 'merit-inp' | 'mfe-nsp' | 'micp' | 'mobile' | 'mobility header' | 'mpls-in-ip' | 'mtp' | 'mux' | 'narp' | 'netblt' | 'nsfnet-igp' | 'nvp-ii' | 'ospfigp' | 'pgm' | 'pim' | 'pipe' | 'pnni' | 'prm' | 'ptp' | 'pup' | 'pvp' | 'qnx' | 'rdp' | 'reserved' | 'rohc' | 'rsvp' | 'rsvp-e2e-ignore' | 'rvd' | 'sat-expak' | 'sat-mon' | 'scc-sp' | 'scps' | 'sctp' | 'sdrp' | 'secure-vmtp' | 'shim6' | 'skip' | 'sm' | 'smp' | 'snp' | 'sprite-rpc' | 'sps' | 'srp' | 'sscopmce' | 'st' | 'stp' | 'sun-nd' | 'swipe' | 'tcf' | 'tcp' | 'tlsp' | 'tp' | 'trunk-1' | 'trunk-2' | 'ttp' | 'udp' | 'udplite' | 'uti' | 'vines' | 'visa' | 'vmtp' | 'vrrp' | 'wb-expak' | 'wb-mon' | 'wesp' | 'wsn' | 'xnet' | 'xns-idp' | 'xtp' >
```

Examples

- `ipProto` ipv4
- `ipProto` ipv6-route
- `ipProto` 'trunk-l'
- `ipProto` 'xnet'

ipServer

IPv4 or IPv6 address (or range of addresses) of a server if you are using the **IP - Server** key.

The `ipServer` filter displays the matching IP only if the request was received by that IP address. The filter does work with the **IP - Pair** key field even though **IP - Pair** shows the client/server IP addresses in the conversation.

To filter an IP range or subnet, enter the first and last IP addresses in the range separated by a hyphen.

Usage

```
ipServer  <IPv4 or IPv6 address.>
```

Examples

- `ipServer` 192.168.1.15
- `ipServer` 10.1.54.212
- '`ip` 192.168.20.0-192.168.20.31'
- `ipServer` FE80:0000:0000:0000:0000:B4EF:EF8F:4819
- `ipServer` FE80::B4EF:EF8F:4819

link

Link protocol from the logical link control layer of a switch.

Usage

```
link  <'802.11 Control' |  '802.11 Keep Alive' |  '802.11 Mgmt' |
'802.11 WEP Encoded Data' |  '802.2 LLC' |  '802.2 LLC SNAP' |
'802.3 Raw' |  'ATM CTRL' |  'Ethernet' |  'FDDI SMT' |  'FIBRE
CHANNEL' |  'OEL2' |  'SNA' |  'Token Ring FDDI' |  'WAN CTRL' >
```

Examples

- `link` '802.11 WEP Encoded Data'
- `link` 'Ethernet'

mac

MAC address

Usage

```
mac  <Any valid MAC address.>
```

Examples

- `mac` 00:08:5C:00:00:01
- `mac` 00:0F:EA:91:04:07

macClient

MAC address of a client when using the **MAC - Client** key.

The `macClient` filter displays the matching MAC address only if the system made the request. The filter does not work with the **MAC - Pair** key even though **MAC - Pair** shows the client/server MAC addresses in the conversation.

Usage

`macClient` *<Any valid MAC address.>*

Examples

- `macClient` 00:08:5C:00:00:01
- `macClient` 00:0F:EA:91:04:07

macServer

MAC address of a server when using the **MAC - Server** key.

The `macServer` filter displays the matching MAC address only if the system received the request. The filter does not work with the **MAC - Pair** key even though **MAC - Pair** shows the client/server MAC addresses in the conversation.

Usage

`macServer` *<Any valid MAC address.>*

Examples

- `macServer` 00:08:5C:00:00:01
- `macServer` 00:0F:EA:91:04:07

mpls

MPLS label.

Usage

```
mpls  <Any number of digits (0-9); up to a maximum of seven
(e.g., 9999999).>
```

Examples

- `mpls` 1
- `mpls` 86
- `mpls` 75863
- `mpls` 9827263

mplsClient

MPLS label for a client when using the **MPLS Label - Client** key.

The `mplsClient` filter displays the matching MPLS identifier only if the system made the request. The filter does not work with the **MPLS Label - Pair** key even though **MPLS Label - Pair** shows the client/server MPLS identifier in the conversation.

Usage

`mplsClient` *<Any number of digits (0-9); up to a maximum of seven (e.g., 9999999).>*

Examples

- `mplsClient` 1
- `mplsClient` 86
- `mplsClient` 75863
- `mplsClient` 9827263

mplsServer

MPLS label for a server when using the **MPLS Label - Server** key.

The `mplsServer` filter displays the matching MPLS identifier only if the system made the request. The filter does not work with the **MPLS Label - Pair** key even though **MPLS Label - Pair** shows the client/server MPLS identifier in the conversation.

Usage

`mplsServer` *<Any number of digits (0-9); up to a maximum of seven (e.g., 9999999).>*

Examples

- `mplsServer` 1
- `mplsServer` 86
- `mplsServer` 75863
- `mplsServer` 9827263

multicastPort

Multicast port when using any of the **Port**, **Port - Client**, or **Port - Server** keys.

Usage

```
multicastPort  <Any valid port number up to 65535.>
```

Examples

- `multicastPort` 80
- `multicastPort` 462
- `multicastPort` 25901

multicastPortClient

Multicast port on a client when using the **Port - Client** key.

The `multicastPortClient` filter displays the matching port only if the system made the request. The filter does not work with the **Port - Client** key even though **Port - Client** shows the client/server Trading Multicast ports in the conversation.

Usage

`multicastPortClient` *<Any valid port number up to 65535.>*

Examples

- `multicastPortClient` 80
- `multicastPortClient` 462
- `multicastPortClient` 25901

multicastPortServer

Multicast port on a server when using the **Port - Client** key.

The `multicastPortServer` filter displays the matching port only if the system made the request. The filter does not work with the **Port - Server** key even though **Port - Server** shows the client/server Trading Multicast ports in the conversation.

Usage

`multicastPortServer` *<Any valid port number up to 65535.>*

Examples

- `multicastPortServer` 80
- `multicastPortServer` 462
- `multicastPortServer` 25901

multicastStreamId

Trading multicast stream ID when using the **Trading Multicast Stream Identifier** key.

Usage

```
multicastStreamId  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `multicastStreamId` 436f6e67726174756c6174696f6e7321
- `multicastStreamId` 671111101031149711611710897116105111111011533

multicastStreamName

Trading multicast stream name when using the **Trading Multicast Stream Name** key.

Usage

```
multicastStreamName  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `multicastStreamName` youtube
- `multicastStreamName` 'datacenter 8'
- `multicastStreamName` 'Pat Smith'

networkType

Kind of network generating the traffic when using the **Network Type** key.

Usage

```
networkType  <'ethernet' |  'netFlow' |  'netFlowTrend' |
'sFlow' |  'wireless' >
```

Examples

- `networkType` 'ethernet'
- `networkType` 'netFlow'
- `networkType` 'wireless'

pattern

Case-insensitive ASCII strings can be searched for using the pattern keyword.

Usage

```
pattern  "search pattern" [(ip|tcp|
```

Examples

- `pattern` "HTTP/1.1"
- `pattern` "HTTP/1.1" [tcpData]
- `pattern` "HTTP/1.1" [14]
- `pattern` "HTTP/1.1" [14-20]
- `pattern` "HTTP/1.1" [14-32767]
- `pattern` "HTTP/1.1" [tcpData:14-20]

patternSensitive

Case-sensitive ASCII strings can be searched for using the patternSensitive keyword.

Usage

```
patternSensitive  "search pattern" [(ip|tcp|
```

Examples

- `patternSensitive` “SELECT Query”
- `patternSensitive` “SELECT Query” [udpData]
- `patternSensitive` “SELECT Query” [14]
- `patternSensitive` “SELECT Query” [14-20]
- `patternSensitive` “SELECT Query” [14-32767]
- `patternSensitive` “SELECT Query” [udpData:14-20]

patternBin

Binary bits can be searched for using the patternBin keyword.

Usage

```
patternBin  "search pattern" [(ip|tcp|
```

Examples

- `patternBin` xxx1xxxx
- `patternBin` xxx1xxxx [tcp]
- `patternBin` xxx1xxxx [tcp:13]
- `patternBin` "0101xxxx xxx1xxxx" [tcp:10-15]

patternHex

Hex characters can be searched for using the patternHex keyword.

Usage

```
patternHex  "search pattern" [(ip|tcp|
```

Examples

- `patternHex` 41706578
- `patternHex` 41706578 [ip]
- `patternHex` "56 69 61 76 69" [13]
- `patternHex` "56 69 61 76 69" [tcp:10-15]

patternRegex

Regular expressions can be used for a search using the patternRegex keyword.

Usage

```
patternRegex  "search pattern" [(ip|tcp|
```

Examples

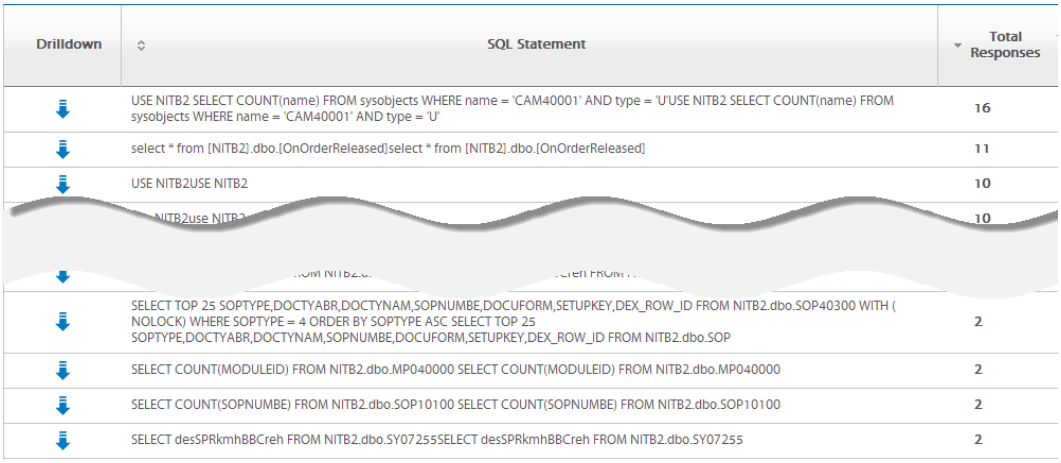
- `patternRegex` /\d+ABCD/

sql

SQL query when using the **SQL Statement** key.

You are not using SQL 'select' statements to match content in a database. Instead you are searching within SQL statements that already exist. The filter you create narrows down the list of possible matches from the **SQL Statement** key. The **SQL Statement** key can contain any characters or numbers.

Figure 1: SQL Statement



Drilldown	SQL Statement	Total Responses
	USE NITB2 SELECT COUNT(name) FROM sysobjects WHERE name = 'CAM40001' AND type = 'U'USE NITB2 SELECT COUNT(name) FROM sysobjects WHERE name = 'CAM40001' AND type = 'U'	16
	select * from [NITB2].dbo.[OnOrderReleased]select * from [NITB2].dbo.[OnOrderReleased]	11
	USE NITB2USE NITB2	10
	NITB2use NITB2	10
	SELECT TOP 25 SOPTYPE,DOCTYABR,DOCTYNAM,SOPNUMBE,DOCUFORM,SETUPKEY,DEX_ROW_ID FROM NITB2.dbo.SOP40300 WITH (NOLOCK) WHERE SOPTYPE = 4 ORDER BY SOPTYPE ASC SELECT TOP 25 SOPTYPE,DOCTYABR,DOCTYNAM,SOPNUMBE,DOCUFORM,SETUPKEY,DEX_ROW_ID FROM NITB2.dbo.SOP	2
	SELECT COUNT(MODULEID) FROM NITB2.dbo.MP040000 SELECT COUNT(MODULEID) FROM NITB2.dbo.MP040000	2
	SELECT COUNT(SOPNUMBE) FROM NITB2.dbo.SOP10100 SELECT COUNT(SOPNUMBE) FROM NITB2.dbo.SOP10100	2
	SELECT desSPRkmhBBCreh FROM NITB2.dbo.SY07255SELECT desSPRkmhBBCreh FROM NITB2.dbo.SY07255	2

Usage

```
sql  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `sql` NITB2
- `sql` MP04000
- `sql` dbo

subnetRangeName

A range of subnet IP addresses when using the **Subnet Range** key.

The `subnetRangeName` does not work with **IP Address**, **IP Address - Pair**, **IP Address - Client** and others. The only key it works with is **Subnet Range**.

Usage

```
subnetRangeName  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `subnetRangeName` '192.168.0.126 - 192.168.1.15'
- `subnetRangeName` '10.1.54.1 - 10.1.55.254'

url

URL, such as for a website or FTP location.

Usage

```
url  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `url` http://observer.viavisolutions.com
- `url` observer.viavisolutions.com
- `url` ftp://ftp.observer.viavisolutions.com

vlan

VLAN ID when using the **VLAN Tag ID** key.

Usage

```
vlan  <Any number of digits (0-9); up to a maximum of four (e.g., 9999).>
```

Examples

- `vlan` 1
- `vlan` 86
- `vlan` 7586
- `vlan` 9827

voipCall

Target phone number of VoIP call.

You can get a list of calls through a server, jitter and MOS scores, and other statistics if you create a widget with both `voipCall` and `voipStationIp` and then filter the VoIP server IP addresses.

Usage

```
voipCall  <Unlimited amount of any text, digits, or characters.>
```

Examples

- `voipCall` '43896758-3-1699564929'
- `voipCall` '67139887-0-5804930681'

voipDeviceIp

IP address (or range of addresses) of a VoIP server.

The `voipDeviceIp` filter cannot pass results to a table with voice station details. If you want to correlate individual calls, use voipStationIp (page 52) instead.

Usage

```
voipDeviceIp  <IPv4 or IPv6 address.>
```

Examples

- `voipDeviceIp` 192.168.1.15
- `voipDeviceIp` 10.1.54.212
- `voipDeviceIp` FE80:0000:0000:0000:0000:B4EF:EF8F:4819
- `voipDeviceIp` FE80::B4EF:EF8F:4819

voipIpType

Kind of device used in VoIP call when using the **VoIP IP Type** key.

If you want to see performance sorted by server, use the *'server-known'* option; for just phones, the *'phone'* field.

```
voipIpType  <'admin-reg server' |  'control processor' |
'gateway' |  'media processor' |  'non-server' |  'outside ip' |
'phone' |  'router' |  'server-inferred' |  'server-known' >
```

Examples

- `voipIpType` 'control processor'
- `voipIpType` 'gateway'
- `voipIpType` 'router'

voipStationIp

IP address of any system used in VoIP call.

Combine `voipStationIp` and `voipIpType` to see how a specific machine is defined.

Usage

`voipStationIp` *<IPv4 or IPv6 address.>*

Examples

- `voipStationIp` 192.168.1.15
- `voipStationIp` 10.1.54.212
- `voipStationIp` FE80:0000:0000:0000:0000:B4EF:EF8F:4819
- `voipStationIp` FE80::B4EF:EF8F:4819