AKIPS Network Monitor
Programming API
Version 18.x

AKIPS Pty Ltd

October 15, 2018

Copyright

Contact AKIPS

Web Site: https://www.akips.com
Email: info@akips.com

Contents

1 Syntax Conventions **6**
- 1.1 Literal Keywords . . . 6
- 1.2 Parameters . . . 6
- 1.3 Optional Parameters . . . 6
- 1.4 Regular Expressions . . . 6

2 Time Filter **7**
- 2.1 Keywords . . . 7
- 2.2 Syntax . . . 8
- 2.3 Examples . . . 9

3 IP Address Filter **11**
- 3.1 Syntax . . . 11
- 3.2 IPv4 Examples . . . 11
- 3.3 IPv6 Examples . . . 11

4 Database Overview **12**
- 4.1 Entities . . . 12
- 4.2 Groups . . . 14
- 4.3 Super Groups . . . 15
- 4.4 Profile Groups . . . 15

5 Entity Commands **16**
- 5.1 add . . . 16
- 5.2 set . . . 17
- 5.3 delete . . . 17
- 5.4 mdelete . . . 17
- 5.5 get . . . 18
- 5.6 mget . . . 19
- 5.7 mlist . . . 22
- 5.8 mtype . . . 23
- 5.9 rename . . . 24

6 Group Commands **25**
- 6.1 add . . . 25
- 6.2 count . . . 25
- 6.3 delete . . . 26
- 6.4 rename . . . 26
- 6.5 prune . . . 26
- 6.6 assign . . . 27
- 6.7 clear . . . 28

7 Profile Commands **29**
- 7.1 assign . . . 29
- 7.2 clear . . . 29

8 Event Commands **30**
- 8.1 Overview . . . 30
- 8.2 add . . . 30
- 8.3 set . . . 31
- 8.4 clear . . . 31
- 8.5 delete . . . 31
- 8.6 mdelete . . . 31
- 8.7 mget . . . 32
- 8.8 rget . . . 33

8.9 tget 33

9 Time-series **34**
9.1 Features 34
9.2 mget 34
9.3 calc 35
9.4 mcalc 35
9.5 series 35
9.6 cseries 36
9.7 top 36

10 Config and Events Web API **37**
10.1 Overview 37
10.2 Syntax 37
10.3 Using Curl 37
10.4 GET Examples 37
10.5 POST Examples 37

11 Syslog / Trap Reporter **38**
11.1 Overview 38
11.2 Command Line Syntax 38
11.3 Output Format 38
11.4 Examples 38
11.5 Web API 40

12 Netflow Reporter **41**
12.1 Overview 41
12.2 Command Line Syntax 41
12.3 Show Flags 41
12.4 Sort Flags 41
12.5 Output Format 41
12.6 Examples 42
12.7 Web API 43

13 Netflow Time-Series **44**
13.1 Command Line Syntax 44
13.2 Output Format 44
13.3 Examples 44
13.4 Web API 45

14 Switch Port Mapper **46**
14.1 Overview 46
14.2 Raw File Formats 46
14.3 Web API 46
14.4 Syntax 46
14.5 Examples 46

15 Unused Interfaces **47**
15.1 Overview 47
15.2 Syntax 47
15.3 CSV Output Format 47
15.4 Examples 47

16 TCP Socket API **48**
16.1 Overview 48
16.2 Syntax 48
16.3 Examples 48

17 Perl Module: Common **49**
17.1 Useful array definitions 49
17.2 BEGIN and END 49
17.3 PRINT_LINE 49
17.4 process_init () 50
17.5 process_signals () 50
17.6 process_lock () 50
17.7 errlog () 50
17.8 trim () 50
17.9 valid_regex () 51
17.10 get_localtime (epoch) 51
17.11 mail () 51
17.12 syslog () 51
17.13 http_send() and http_result() . . . 53

18 Perl Module: ADB **54**
18.1 adb_flush () 54
18.2 adb_send () 54
18.3 adb_result () 54

19 Perl Module: Discover **55**
19.1 discover_scan() 55
19.2 discover_config 55
19.3 discover_device_rewalk 56

20 Perl Scripting Guide **57**
20.1 Basic Structure 57

1 Syntax Conventions

1.1 Literal Keywords

Literal keywords are typed exactly as they appear in the API syntax.

1.2 Parameters

Parameters are fields which are expecting a value and are bound by curly braces.

```
{type} {value}
```

1.3 Optional Parameters

Optional parameters will be displayed inside square brackets:

```
[name {name}]
```

Optional parameters may be nested. For example:

```
mlist {type} [{parent regex} [{child regex} [{attribute regex}]]]
```

1.4 Regular Expressions

One of the tricks with regular expression syntax is you need to know what underlying language is compiling and executing the regex. The standard C regex library is different to Perl, Java, Javascript, etc.

AKIPS is mostly written in C and Perl, with a small amount of Javascript to handle browser interaction. For consistency, all C programs are linked against the Perl-Compatible Regular Expressions (PCRE) library.

1.4.1 Rules

- The AKIPS API requires all regex strings to be bound by a slash /.
- Any regex that contains white space must be quoted in either single or double quotes.
- Use the syntax /!{regex}/ to negate a regex
- Parent/child/attribute/value regex can be either:
  - Exact matching text string
  - A wildcard *
  - A Perl type regular expression

1.4.2 Examples

```
/NYC-RO[1-5]/
"/Cisco IOS/"
'/Cisco IOS/'
```

1.4.3 Regex Performance

The more precise you are with your regex syntax, the faster it will be when querying large configurations.

For example:

```
/^NewYork-Router/
```

will be faster than

```
/NewYork-Router/
```

because it is saying that the start of the match MUST start with *NewYork-Router*.

1.4.4 Reference

We recommend having a copy of *O'Reilly Regular Expressions Cookbook* on hand. It covers regex syntax for most programming languages.

2 Time Filter

AKIPS uses a common time filter syntax throughout the entire product.

The time filter syntax is used in:

- Reports and graphs
- Threshold rules
- Alerting rules
- Availability rules
- Syslog, SNMP Trap, and NetFlow reporters
- etc...

2.1 Keywords

Points in Time	
Now	
YYYY	
YYYY-MM	
YYYY-MM-DD	
YYYY-MM-DD HH:MM	
StartOfLastMinute	EndOfLastMinute
StartOfThisMinute	EndOfThisMinute
StartOfLastHour	EndOfLastHour
StartOfThisHour	EndOfThisHour
StartOfYesterday	EndOfYesterday
StartOfToday	EndOfToday
StartOfLastWeek	EndOfLastWeek
StartOfThisWeek	EndOfThisWeek
StartOfLastMonth	EndOfLastMonth
StartOfThisMonth	EndOfThisMonth
StartOfLastYear	EndOfLastYear
StartOfThisYear	EndOfThisYear

Time Ranges	
YYYY	
YYYY-MM	
YYYY-MM-DD	
ThisHour	LastHour
Today	Yesterday
ThisWeek	LastWeek
ThisMonth	LastMonth
ThisYear	LastYear

Days of Week	
Short	**Long**
Sun	Sunday
Mon	Monday
Tue	Tuesday
Wed	Wednesday
Thu	Thursday
Fri	Friday
Sat	Saturday

Duration			
Character	**Description**	**Example**	**Result**
m	minutes	7m	7 minutes
h	hours	2h	2 hours
d	days	5d	5 days
w	weeks	2w	2 weeks
M	months	3M	3 months
y	years	1y	1 year

2.2 Syntax

Duration

```
{number}{m h d w M y}
```

Time Range

```
last{duration}

{time range keyword}

yyyy q[1-4]

yyyy
yyyy-mm
yyyy-mm-dd
yyyy-mm-dd hh:mm
yyyy-mm-dd hh:mm to hh:mm

from yyyy to yyyy
from yyyy-mm to yyyy-mm
from yyyy-mm-dd to yyyy-mm-dd
from yyyy-mm-dd hh:mm to yyyy-mm-dd hh:mm

from yyyy for {duration}
from yyyy-mm for {duration}
from yyyy-mm-dd for {duration}
from yyyy-mm-dd hh:mm for {duration}

from {epoch} to {epoch}

from {point in time} to {point in time}
from {point in time} to {point in time} [+- duration]
from {point in time} [+- duration] to {point in time} [+- duration]
```

Weekday Filters

```
{wday} {hh:mm} to {hh:mm}
{wday} to {wday} {hh:mm} to {hh:mm}

not {wday} {hh:mm} to {hh:mm}
not {wday} to {wday} {hh:mm} to {hh:mm}
```

2.3 Examples

The best way of learning the AKIPS Time Filter API is to use the **Admin -> API -> Command Console**. Use the "**tf dump**" command to test your time filter rules. The output will contain a **span** and a list of **from / to** pairs.

NOTE: The time filter must be quoted with double quotes if it contains any spaces.

1. Simple last 30 minutes range:

```
tf dump "last30m"
span    2017-10-20 15:03:03 to 2017-10-20 15:33:03
```

2. Simple last 24 hour range:

```
tf dump "last24h"
span    2017-10-19 15:36:39 to 2017-10-20 15:36:39
```

3. Simple last 1 day range:

```
tf dump "last1d"
span    2017-10-20 00:00:00 to 2017-10-20 23:59:59
```

4. Simple last 7 days range:

```
tf dump "last7d"
span    2017-10-14 00:00:00 to 2017-10-20 23:59:59
```

5. Business hours for last week:

```
tf dump "lastweek; mon to fri 8:00 to 17:00"
span    2017-10-12 00:00:00 to 2017-10-18 23:59:59
include 2017-10-13 08:00:00 to 2017-10-13 17:00:00
include 2017-10-14 08:00:00 to 2017-10-14 17:00:00
include 2017-10-15 08:00:00 to 2017-10-15 17:00:00
include 2017-10-16 08:00:00 to 2017-10-16 17:00:00
include 2017-10-17 08:00:00 to 2017-10-17 17:00:00
```

6. Business hours last week + longer on Thursday:

```
tf dump "lastweek; mon to fri 8:00 to 17:00; thu 8:00 to 21:00"
span    2017-10-12 00:00:00 to 2017-10-18 23:59:59
include 2017-10-13 08:00:00 to 2017-10-13 17:00:00
include 2017-10-14 08:00:00 to 2017-10-14 17:00:00
include 2017-10-15 08:00:00 to 2017-10-15 17:00:00
include 2017-10-16 08:00:00 to 2017-10-16 21:00:00
include 2017-10-17 08:00:00 to 2017-10-17 17:00:00
```

7. Outside business hours for last week:

```
tf dump "lastweek; not mon to fri 8:00 to 17:00; not sat 8:00 to 12:00;"
span    2018-05-24 00:00:00 to 2018-05-30 23:59:59
include 2018-05-24 00:00:00 to 2018-05-25 08:00:00
include 2018-05-25 17:00:00 to 2018-05-26 08:00:00
include 2018-05-26 17:00:00 to 2018-05-27 08:00:00
include 2018-05-27 17:00:00 to 2018-05-28 08:00:00
include 2018-05-28 17:00:00 to 2018-05-29 08:00:00
include 2018-05-29 17:00:00 to 2018-05-30 08:00:00
include 2018-05-30 12:00:00 to 2018-05-30 23:59:59
```

8. Business hours of 2017 Q1:

```
tf dump "2017 Q1; mon to fri 8:00 to 17:00"
span    2017-01-01 00:00:00 to 2017-03-31 23:59:59
include 2017-01-01 08:00:00 to 2017-01-01 17:00:00
include 2017-01-02 08:00:00 to 2017-01-02 17:00:00
include 2017-01-03 08:00:00 to 2017-01-03 17:00:00
...
include 2017-03-28 08:00:00 to 2017-03-28 17:00:00
include 2017-03-31 08:00:00 to 2017-03-31 17:00:00
```

9. Prior 30 minute period:

```
tf dump "from now - 1h to now - 30m"
span    2017-10-27 08:03:24 to 2017-10-27 08:33:24
```

10. Today until now:

```
tf dump "from StartOfToday to now;"
span    2017-10-27 00:00:00 to 2017-10-27 09:00:20
```

11. Business hours on the 1st June 2017:

```
tf dump "2017-06-01 8:00 to 17:00"
span    2017-06-01 08:00:00 to 2017-06-01 17:00:00
```

12. After hours on the 1st June 2017:

```
tf dump "from 2017-06-01 18:00 to 2017-06-02 06:00"
span    2017-06-01 18:00:00 to 2017-06-02 06:00:00
```

13. First two months of 2018:

```
tf dump "from 2018 for 2M"
span    2018-01-01 00:00:00 to 2018-03-01 00:00:00
```

14. Yesterday at midday for 3 hours:

```
tf dump "from startofyesterday + 12h to startofyesterday + 15h"
span    2018-05-31 12:00:00 to 2018-05-31 15:00:00
```

3 IP Address Filter

The IP address filter handles both IPv4 and IPv6. The IP address filter is used in the following:

- Ping Discover rules
- Netflow Reporter
- Syslog / SNMP Trap reporter

3.1 Syntax

NOTE: Range and wildcard can appear in any position of the IP address.

```
{address}/{mask}
{address}.*
{address}[{range}]
{address}[{range}]/{mask}
{address}[{range}].*
```

3.2 IPv4 Examples

1. A single IP address:
   ```
   10.1.1.50
   10.1.1.50/32
   ```
2. A single /24 subnet:
   ```
   10.0.0.0/24
   10.0.0.*
   ```
   from 10.0.0.0 to 10.0.0.255
3. Twenty /24 subnets:
   ```
   10.0.0-20.0/24
   10.0.0-20.*
   ```
   from 10.0.0.0 to 10.0.20.255
4. 256 subnets each with 1 address:
   ```
   10.0.0-255.1
   10.0.*.1
   ```
   ```
   10.0.0.1
   10.0.1.1
   10.0.2.1
   ...
   10.0.255.1
   ```

3.3 IPv6 Examples

1. A single IPv6 address:
   ```
   fd00:10:1:1::1
   ```
2. 255 IPv6 addresses on the same network:
   ```
   fd00:10:1:1::0-ff
   ```
   fd00:10:1:1::0 to fd00:10:1:1::ff
3. One address on multiple networks:
   ```
   fd00:10:1:1-50:1
   ```
   ```
   fd00:10:1:1::1
   fd00:10:1:2::1
   fd00:10:1:3::1
   ...
   fd00:10:1:50::1
   ```

4 Database Overview

Before starting with the API, you must have a clear understanding of:

- Entities
- Groups and Super Groups
- User Profiles

4.1 Entities

The AKIPS database architecture defines three levels:

- Level 1: Parent
- Level 2: Child
- Level 3: Attribute

Example:

Level 1	**Level 2**	**Level 3**
Parent	*Child*	*Attribute*
router1	sys	SNMPv2-MIB.sysName
router1	Se0/0	IF-MIB.ifHCInOctets
router1	Se0/0	IF-MIB.ifHCOutOctets
router2	Se0/0	IF-MIB.ifHCInOctets
router2	Se0/0	IF-MIB.ifHCOutOctets

4.1.1 Important Rules

1. Entities must be created in order of parent, child and then attribute.
   - The parent must exist before a child is added
   - The child must exist before an attribute is added
2. Entities are *typed*. See the table below.
3. Once added, the type of an entity can not be changed. It must be deleted and re-added.
4. A parent does not have a value.
5. A child can optionally have a value of *index,description*.
6. All attributes are assigned a value. Refer to the table below.

4.1.2 Entity Types

Parent Types	**Child Types**	**Attribute Types**	**Virtual Attribute Types**

`device`
`report`
`user`
`parent (generic)`
	`interface`
	`ipsla`
	`memory`
	`processor`
	`storage`
	`system`
	`temperature`
	`child (generic)`
		`counter`
		`enum`
		`gauge`
		`integer`
		`text`
		`timestamp`
		`uptime`
		`rtt`
		`attribute (generic)`
			`ifutil - interface utilisation`
			`ifrate - interface bps`
			`vutil - utilisation`
			`vnutil - calculate used from free`
			`vdiff - difference x = a - b`
			`temp_f - convert C to F`

4.1.3 Parent Values

A value can NOT be assigned to a parent entity.

4.1.4 Child Values

A value can optionally be assigned to a child entity. The value is in the format of "index,[description]", where description is optional. For example, a child called 'Fa0/1' may be assigned value of "1,Link to server", where '1' is the ifIndex of the interface in ifTable, and "Link to server" is the ifAlias.

4.1.5 Attribute Values

The following table lists the attribute types and values which can be assigned:

Type	Value	Example
counter	Must be '1'	1
gauge	scale (positive to multiply, negative to divide)	1
enum	number,text	1,up or 2,down
integer	number	1000000000
text	text	The quick brown fox
timestamp	seconds since epoch	1406787487
uptime	seconds since status change	12345

4.1.6 Conceptual Layout

"Conceptually" this is what the configuration looks like:

Parent		Child			Attribute		
type	Name	Type	Name	Value	Type	Name	Value
device	router1	system	sys	0	text	SNMPv2-MIB.sysName	router1.akips.com
device	router1	system	sys	0	text	SNMPv2-MIB.sysLocation	Test Rack
device	router1	system	sys	0	text	SNMPv2-MIB.sysContact	AKIPS Devel
device	router1	system	sys	0	text	SNMPv2-MIB.sysDescr	Cisco IOS C3560 ...
device	router1	system	sys	0	text	SNMPv2-MIB.sysObjectID	catalyst356024PS
device	router1	system	sys	0	text	ip4addr	10.1.8.250
device	router1	system	sys	0	text	ip6addr	fd00:10:1:8::250
device	router1	interface	Se1/1	1,Test network	enum	IF-MIB.ifAdminStatus	1,up
device	router1	interface	Se1/1	1,Test network	enum	IF-MIB.ifOperStatus	1,up
device	router1	interface	Se1/1	1,Test network	text	IF-MIB.ifAlias	Test network
device	router1	interface	Se1/1	1,Test network	text	IF-MIB.ifName	Se1/1
device	router1	interface	Se1/1	1,Test network	integer	IF-MIB.ifSpeed	2048000
device	router1	interface	Se1/1	1,Test network	enum	IF-MIB.ifType	22,propPointToPointSerial
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifHCInOctets	1
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifHCOutOctets	1
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifInErrors	1
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifOutErrors	1
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifInDiscards	1
device	router1	interface	Se1/1	1,Test network	counter	IF-MIB.ifOutDiscards	1

4.2 Groups

4.2.1 Important Rules

1. Groups are *typed*. See the table below for a list of group types.
2. An entity can only be assigned to a group with the same type:
   - An interface can only be assigned to an interface group
   - A device can only be assigned to a device group
3. Group inheritance
   - A child inherits group assignments from its parent
   - An attribute inherits group assignments from its child and parent
4. An entity can not be directly assigned to a super group. Only groups can be assigned to a super group.

4.2.2 Group Types

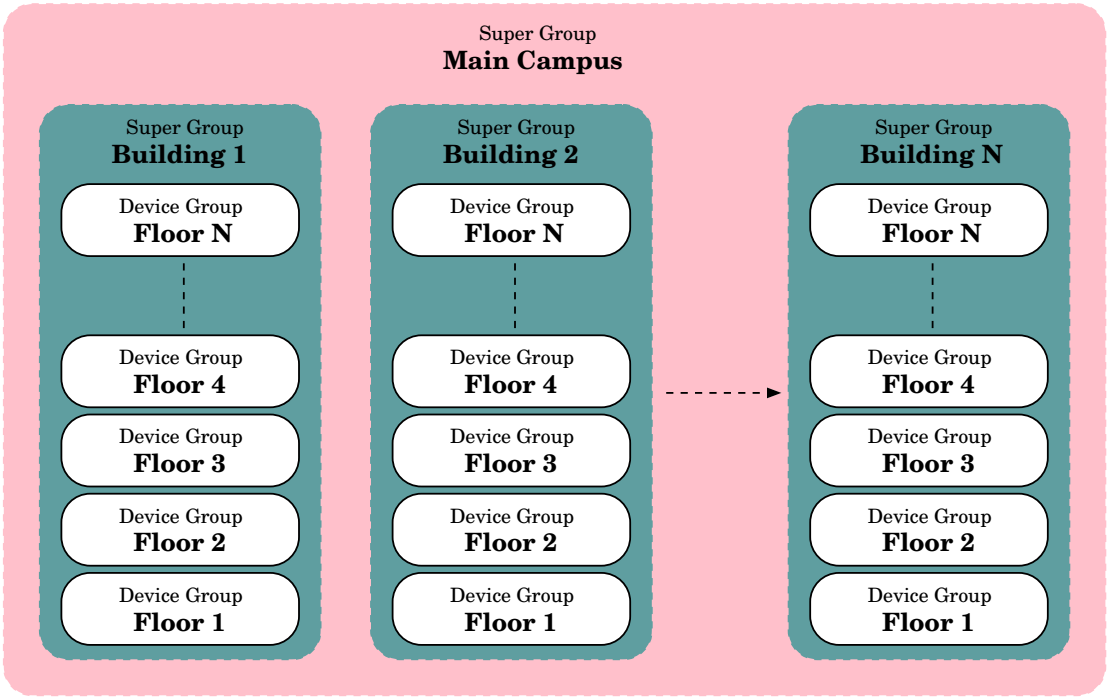
The group type has to be specified when creating a group. When assigning entities to a group, the assignment will only succeed if the entity and group types match. eg. if you attempt to assign an interface to a device group, then it will fail, and an error message will be logged to the AKIPS debug logs.

The following table lists the common group types that you may use.

Level	**Group Type**	**Comment**
n/a	Super	
n/a	Profile	
Parent	parent	generic parent type
Parent	device	
Child	child	generic child type
Child	system	
Child	interface	
Child	ipsla	
Child	storage	
Child	processor	
Child	memory	
Child	temperature	
Attribute	attribute	generic attribute type

4.3 Super Groups

Super groups are used to create *groups of groups*. In the following campus network example, a device group is created for each floor of a building, then all those device groups are assigned to a single super group called ***Building 1***, then all the building super groups are assigned to the ***Main Campus*** super group.



4.4 Profile Groups

TO BE COMPLETED.

5 Entity Commands

5.1 add

- The *add* command will add a new entity if it does not exist. If it does already exist, then only the value will be updated.
- Once an entity has been added, its type can NOT be changed. To change the type of an entity, it has to be deleted and re-added.
- If you know an entity already exists in the database and only want to update its value, use the *set* command described below instead.

5.1.1 Syntax

```
add {type} {parent name}
add {type} {parent name} {child name}
add {type} {parent name} {child name} = {index,[description]}
add {type} {parent name} {child name} {attribute name}
add {type} {parent name} {child name} {attribute name} = {value}
```

5.1.2 Examples

1. Add a router with an interface:

```
add device    Router1
add system    Router1 sys = 0
add text      Router1 sys SNMPv2-MIB.sysName = "Router1.akips.com"
add text      Router1 sys SNMPv2-MIB.sysLocation = "Test Rack"
add text      Router1 sys SNMPv2-MIB.sysDesc = "Cisco IOS 7200 Version 12.4(9)T RELEASE (fc1)"
add text      Router1 sys SNMPv2-MIB.sysContact = "AKIPS developers"
add uptime    Router1 sys SNMPv2-MIB.sysUpTime = 1432703430
add text      Router1 sys ip4addr = 10.1.8.250
add text      Router1 sys ip6addr = fd00:10:1:8::250
add system    Router1 ping4 = "4,10.1.8.4"
add rtt       Router1 ping4 PING.icmpRtt
add counter   Router1 ping4 PING.icmpDup
add counter   Router1 ping4 PING.icmpLost
add enum      Router1 ping4 PING.icmpState = 2,up
add system    Router1 ping6 = "6,fd00:10:1:8::250"
add rtt       Router1 ping6 PING.icmpRtt
add counter   Router1 ping6 PING.icmpDup
add counter   Router1 ping6 PING.icmpLost
add enum      Router1 ping6 PING.icmpState
add interface Router1 Fa0/0 = "1,Link to remote site"
add integer   Router1 Fa0/0 IF-MIB.ifSpeed = 1000000000
add text      Router1 Fa0/0 IF-MIB.ifAlias = "Link to remote site"
add counter   Router1 Fa0/0 IF-MIB.ifHCInOctets  = 1
add counter   Router1 Fa0/0 IF-MIB.ifHCOutOctets = 1
add enum      Router1 Fa0/0 IF-MIB.ifAdminStatus = 1,up
add enum      Router1 Fa0/0 IF-MIB.ifOperStatus  = 1,up
add counter   Router1 Fa0/0 IF-MIB.ifHCInBroadcastPkts = 1
add counter   Router1 Fa0/0 IF-MIB.ifHCOutBroadcastPkts = 1
add counter   Router1 Fa0/0 IF-MIB.ifHCInMulticastPkts = 1
add counter   Router1 Fa0/0 IF-MIB.ifHCOutMulticastPkts = 1
...
```

2. You might want to add some custom entries for asset information:

```
add child     Router1 asset
add text      Router1 asset serial_number = 123456789
add text      Router1 asset purchase_date = 2017/05/02
add integer   Router1 asset purchase_cost = 12013
```

5.2 set

If an entity already exists, then the value can be updated using the *set* command.

5.2.1 Syntax

```
set {parent name} {child name} = {value}
set {parent name} {child name} {attribute name} = {value}
```

5.2.2 Examples

```
set swt250 sys SNMPv2-MIB.sysLocation = "AKIPS Test Lab"
set swt250 sys SNMPv2-MIB.sysContact = "AKIPS developers"
set swt250 sys ip4addr = 10.1.8.250
```

5.3 delete

The *delete* command will delete a single entity and all its children/attributes.

5.3.1 Syntax

```
delete {type} {parent name}
delete {type} {parent name} {child name}
delete {type} {parent name} {child name} {attribute name}
```

5.3.2 Examples

1. Delete a device. This will also delete all underlying childs and attributes for the device:
   ```
   delete device CoreRouter1
   ```
2. Delete a child. This will also delete all underlying attributes for the child:
   ```
   delete interface CoreRouter1 Fa0/0
   ```
3. Delete a single attribute:
   ```
   delete integer CoreRouter1 Fa0/0 IF-MIB.ifSpeed
   ```

5.4 mdelete

WARNING: This command is not forgiving if you make a mistake. When deleting entities, it is best to do a *mget* command first to confirm your selection is correct.

mdelete stands for *multiple delete*. The mdelete command provides a quick method to delete large numbers of entities.

5.4.1 Syntax

```
mdelete {type} {parent regex} [{child regex} [{attribute regex}]]
        [any|all|not group {group name} ...]
```

5.4.2 Examples

1. Delete all devices (dangerous)
   ```
   mdelete device *
   ```
2. Delete all devices in the group "testlab"
   ```
   mdelete device * any group testlab
   ```
3. Delete all IF-MIB attributes:
   ```
   mdelete * CoreRouter1 Fa0/0 /IF-MIB/
   ```
4. Delete all Cisco IPSLA entities for a device:
   ```
   mdelete * Corerouter1 /ipsla/
   ```

5.5 get

The *get* command returns the value assigned to a child or attribute. Only the value is returned, not the parent/child/attribute names.

5.5.1 Syntax

```
get {parent name} {child name}
get {parent name} {child name} {attribute name}
```

5.5.2 Examples

1. A *get* on an interface will return the child value: *index,description*

```
get swt250 Fa0/1
10001,Link to swt251
```

2. A *get* on an attribute with a type of *text*:

```
get swt250 sys SNMPv2-MIB.sysLocation
Test Room
```

3. A *get* on an attribute with a type of *enum* returns five comma separated values:
   (a) Enumeration number (from MIB)
   (b) Text value (from MIB)
   (c) Create epoch time (ie. seconds since 1st Jan 1970 UTC)
   (d) Modified epoch time (ie. seconds since 1st Jan 1970 UTC)
   (e) Child description

```
get swt250 Fa0/1 IF-MIB.ifOperStatus
1,up,1431579447,1431579447,link to swt251
```

4. A *get* on an attribute with a type of *uptime* returns two comma separated values:
   (a) Epoch time when the value was reset to zero
   (b) Epoch time when the attribute value was last updated (ie. utime)

```
get swt250 sys SNMPv2-MIB.sysUpTime
1427283250,1433815768
```

5.6 mget

mget is the most commonly used API to retrieve large amounts of configuration information in a single command. The optional regex, profile and grouping parameters allow you to target the exact information to be retrieved.

5.6.1 Syntax

```
mget {type} [{parent regex} [{child regex} [{attribute regex}]]]
     [value {text|integer|/regex/}]
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

`{type}`

- Query with just the *type*

```
mget device
mget user
mget interface
mget processor
mget system
```

- If a parent is requested, then the *type* must be a wildcard or a parent type:

```
mget * /^nyc-.*-rtr/
mget device *
mget user *
```

- If a child is requested, then the *type* must be a wildcard or a child type:

```
mget * swt250 /^Fa/
mget system swt250 *
mget interface * *
mget interface * /^Fa0/1/
```

- If an attribute is requested, then the type must be a wildcard or an attribute type:

```
mget counter swt250 * *
mget counter swt250 Fa0/1 /IF-MIB/
```

`[{parent regex} [{child regex} [{attribute regex}]]]`

Each of these can be either:

- Wildcard
- Exact string match
- Regular expression enclosed in forward slash characters

`[value {text|integer|/regex/}]`

- *text* - match on the extact text string
- *integer* - match on the extact number (eg. IF-MIB.ifSpeed)
- *regex* - match on a Perl text regular express

`[profile {profile name}]`

The *profile* option is typically used by the GUI backend to restrict what data a user profile has access to.

`[any|all|not group {group name} ...]`

- *any* is a logical OR
- *all* is a logical AND

5.6.2 Return Values

The format of the return values depends on the entity level requested (ie. parent, child or attribute).

1. Parent
   ```
   {parent}
   ```
2. Child
   ```
   {parent} {child} = {index},{description}
   ```
3. Attribute
   ```
   {parent} {child} {attribute} = {values}
   ```

5.6.3 Examples

1. Retrieve the entire configuration for a single device:
   ```
   mget * swt250 * *
   ```

2. Retrieve the sysName for each device. SNMPv2-MIB.sysName has an attribute type of *text*.
   ```
   mget text * * SNMPv2-MIB.sysName
   Atlanta-ro sys SNMPv2-MIB.sysName = Atlanta-ro.akips.com
   Baltimore-ro sys SNMPv2-MIB.sysName = Baltimore-ro.akips.com
   Boston-ro sys SNMPv2-MIB.sysName = Boston-ro.akips.com
   Charlotte-ro sys SNMPv2-MIB.sysName = Charlotte-ro.akips.com
   Chicago-ro sys SNMPv2-MIB.sysName = Chicago-ro.akips.com
   Cincinnati-ro sys SNMPv2-MIB.sysName = Cincinnati-ro.akips.com
   cisco-74-1-1 sys SNMPv2-MIB.sysName = cisco-74-1-1
   Cleveland-ro sys SNMPv2-MIB.sysName = Cleveland-ro.akips.com
   Columbus-ro sys SNMPv2-MIB.sysName = Columbus-ro.akips.com
   Detroit-ro sys SNMPv2-MIB.sysName = Detroit-ro.akips.com
   ...
   ```

3. Retrieve all IF-MIB objects for a single interface:
   ```
   mget * swt250 Fa0/1 /^IF-MIB.*/
   swt250 Fa0/1 IF-MIB.ifAdminStatus = 1,up,1434416774,1434416774
   swt250 Fa0/1 IF-MIB.ifAlias = Link to swt251
   swt250 Fa0/1 IF-MIB.ifDescr = FastEthernet0/1
   swt250 Fa0/1 IF-MIB.ifDuplex = 3,fullDuplex,1434416774,1434416774
   swt250 Fa0/1 IF-MIB.ifHCInBroadcastPkts = 1
   swt250 Fa0/1 IF-MIB.ifHCInMulticastPkts = 1
   swt250 Fa0/1 IF-MIB.ifHCInOctets = 1
   swt250 Fa0/1 IF-MIB.ifHCInUcastPkts = 1
   swt250 Fa0/1 IF-MIB.ifHCOutBroadcastPkts = 1
   swt250 Fa0/1 IF-MIB.ifHCOutMulticastPkts = 1
   swt250 Fa0/1 IF-MIB.ifHCOutOctets = 1
   swt250 Fa0/1 IF-MIB.ifHCOutUcastPkts = 1
   swt250 Fa0/1 IF-MIB.ifInBitRate = IF-MIB.ifHCInOctets
   swt250 Fa0/1 IF-MIB.ifIndex = 10001
   swt250 Fa0/1 IF-MIB.ifInDiscards = 1
   swt250 Fa0/1 IF-MIB.ifInErrors = 1
   swt250 Fa0/1 IF-MIB.ifInUtil = IF-MIB.ifHCInOctets,IF-MIB.ifSpeed
   swt250 Fa0/1 IF-MIB.ifName = Fa0/1
   swt250 Fa0/1 IF-MIB.ifOperStatus = 1,up,1434416774,1434416774
   swt250 Fa0/1 IF-MIB.ifOutBitRate = IF-MIB.ifHCOutOctets
   swt250 Fa0/1 IF-MIB.ifOutDiscards = 1
   swt250 Fa0/1 IF-MIB.ifOutErrors = 1
   swt250 Fa0/1 IF-MIB.ifOutUtil = IF-MIB.ifHCOutOctets,IF-MIB.ifSpeed
   swt250 Fa0/1 IF-MIB.ifPhysAddress = 0016c701ae03
   swt250 Fa0/1 IF-MIB.ifSpeed = 100000000
   swt250 Fa0/1 IF-MIB.ifType = 6,ethernetCsmacd,1434416774,1434416774
   ```

4. Retrieve the IPv4 address used for each device:

```
mget text * sys ip4addr
cisco-74-1-1 sys ip4addr = 10.74.1.1
cisco-74-1-10 sys ip4addr = 10.74.1.10
cisco-74-1-11 sys ip4addr = 10.74.1.11
cisco-74-1-12 sys ip4addr = 10.74.1.12
cisco-74-1-13 sys ip4addr = 10.74.1.13
cisco-74-1-14 sys ip4addr = 10.74.1.14
...
```

5. Retrieve the IPv6 address used for each device:

```
mget text * sys ip6addr
Atlanta-ro sys ip6addr = fd00:10:4:26::1
Baltimore-ro sys ip6addr = fd00:10:4:22::1
Boston-ro sys ip6addr = fd00:10:4:23::1
Charlotte-ro sys ip6addr = fd00:10:4:27::1
Chicago-ro sys ip6addr = fd00:10:4:40::1
Cincinnati-ro sys ip6addr = fd00:10:4:45::1
LittleRock-ro sys ip6addr = fd00:10:4:49::1
...
```

6. Retrieve both the IPv4 and IPv6 addresses used for each device:

```
mget text * sys /ip.addr/
Atlanta-ro sys ip4addr = 10.4.26.1
Atlanta-ro sys ip6addr = fd00:10:4:26::1
Baltimore-ro sys ip4addr = 10.4.22.1
Baltimore-ro sys ip6addr = fd00:10:4:22::1
Boston-ro sys ip4addr = 10.4.23.1
Boston-ro sys ip6addr = fd00:10:4:23::1
Charlotte-ro sys ip4addr = 10.4.27.1
Charlotte-ro sys ip6addr = fd00:10:4:27::1
Chicago-ro sys ip4addr = 10.4.40.1
...
```

7. Retrieve a list of devices in the 'data_center' device group:

```
mget device * any group data_center
```

8. Retrieve a list of devices in the 'data_center' *AND* 'cisco' device groups:

```
mget device * all group data_center cisco
```

9. Retrieve a list of devices in the 'cisco' *OR* 'juniper' device groups:

```
mget device * any group cisco juniper
```

10. Retrieve a list of interfaces in the 'uplinks' interface group:

```
mget interface * * any group uplinks
```

5.7 mlist

The *mlist* command returns a list of matching entities. The syntax is the same as the *mget* command but the output does not contain any entity values.

5.7.1 Syntax

```
mlist {type} [{parent regex} [{child regex} [{attribute regex}]]]
      [value {text|integer|/regex/}]
      [profile {profile name}]
      [any|all|not group {group name} ...]
```

5.7.2 Examples

1. Retrieve a list of all interfaces on a device:

```
mlist interface swt250 *
swt250 Fa0/1
swt250 Fa0/10
swt250 Fa0/11
swt250 Fa0/12
swt250 Fa0/13
swt250 Fa0/14
swt250 Fa0/15
swt250 Fa0/16
...
```

2. Retrieve a list of interface attributes on a device with a type counter:

```
mlist counter swt250 * /^IF-MIB./
swt250 Fa0/1 IF-MIB.ifHCInBroadcastPkts
swt250 Fa0/1 IF-MIB.ifHCInMulticastPkts
swt250 Fa0/1 IF-MIB.ifHCInOctets
swt250 Fa0/1 IF-MIB.ifHCInUcastPkts
swt250 Fa0/1 IF-MIB.ifHCOutBroadcastPkts
swt250 Fa0/1 IF-MIB.ifHCOutMulticastPkts
swt250 Fa0/1 IF-MIB.ifHCOutOctets
swt250 Fa0/1 IF-MIB.ifHCOutUcastPkts
swt250 Fa0/1 IF-MIB.ifInDiscards
swt250 Fa0/1 IF-MIB.ifInErrors
...
```

3. Retrieve a list of enumerated attribute types configured for a device:

```
mlist enum swt250 * *
swt250 cstack.1 CISCO-STACK-MIB.moduleStandbyStatus
swt250 cstack.1 CISCO-STACK-MIB.moduleStatus
swt250 Fa0/1 IF-MIB.ifAdminStatus
swt250 Fa0/1 IF-MIB.ifDuplex
swt250 Fa0/1 IF-MIB.ifOperStatus
swt250 Fa0/1 IF-MIB.ifType
swt250 Fa0/10 IF-MIB.ifAdminStatus
swt250 Fa0/10 IF-MIB.ifDuplex
swt250 Fa0/10 IF-MIB.ifOperStatus
swt250 Fa0/10 IF-MIB.ifType
...
```

5.8 mtype

The *mtype* command returns the entity type in the AKIPS configuration. The syntax is the same as the *mget* command but the output contains the entity type instead of a entity value.

5.8.1 Syntax

```
mtype {type} [{parent regex} [{child regex} [{attribute regex}]]]
      [value {text|integer|/regex/}]
      [profile {profile name}]
      [any|all|not group {group name} ...]
```

5.8.2 Examples

1. Retrieve a list of child types for a device:

```
mtype * swt250 *
swt250 cpu.1 processor
swt250 cstack.1 child
swt250 Fa0/1 interface
swt250 Fa0/10 interface
swt250 Fa0/11 interface
swt250 Fa0/12 interface
...
swt250 Gi0/1 interface
swt250 Gi0/2 interface
swt250 ipsla.1 ipsla
swt250 ipsla.2 ipsla
swt250 ping6 system
swt250 psu.1003 system
swt250 ram.1 memory
swt250 ram.2 memory
swt250 sys system
```

2. Retrieve a list of attribute types for a device:

```
mtype * swt250 * *
swt250 cpu.1 CISCO-PROCESS-MIB.cpmCPUTotal1minRev gauge
swt250 cstack.1 CISCO-STACK-MIB.moduleModel text
swt250 cstack.1 CISCO-STACK-MIB.moduleStandbyStatus enum
swt250 cstack.1 CISCO-STACK-MIB.moduleStatus enum
swt250 Fa0/1 IF-MIB.ifAdminStatus enum
swt250 Fa0/1 IF-MIB.ifAlias text
swt250 Fa0/1 IF-MIB.ifDescr text
swt250 Fa0/1 IF-MIB.ifDuplex enum
swt250 Fa0/1 IF-MIB.ifHCInBroadcastPkts counter
swt250 Fa0/1 IF-MIB.ifHCInMulticastPkts counter
...
swt250 sys ip6addr text
swt250 sys mac_md5 text
swt250 sys SNMP.auth text
swt250 sys SNMP.auth_password text
swt250 sys SNMP.ipaddr text
swt250 sys SNMP.maxrep integer
swt250 sys SNMP.priv text
swt250 sys SNMP.priv_password text
swt250 sys SNMP.snmpState enum
swt250 sys SNMP.user text
swt250 sys SNMP.version integer
swt250 sys SNMPv2-MIB.sysContact text
swt250 sys SNMPv2-MIB.sysDescr text
swt250 sys SNMPv2-MIB.sysLocation text
swt250 sys SNMPv2-MIB.sysName text
swt250 sys SNMPv2-MIB.sysObjectID text
swt250 sys SNMPv2-MIB.sysUpTime uptime
```

5.9 rename

The *rename* command allows you to rename either a parent or child.

5.9.1 Syntax

```
rename {type} {parent name from} {parent name to}
rename {type} {parent name from} {child name from} {parent name to} {child name to}
```

5.9.2 Examples

1. Rename a device:
   ```
   rename device nyc-rtr usa-nyc-rtr1
   ```
2. Move an interface from one device to another:
   ```
   rename interface nyc-rtr1 Se0/1  nyc-rtr2 Se0/1
   ```

6 Group Commands

6.1 add

- The *add* command will add a new group if it does not exist.
- Adding the same group twice has no affect.
- Once a group has been added, its *type* can NOT be changed. To change the *type* of a group, it has to be deleted and re-added.

6.1.1 Syntax

```
add {type} group {group name}
```

6.1.2 Examples

1. Add a *core-routers* group:
   ```
   add device group core-routers
   ```
2. Add an interface group for all switch uplink ports:
   ```
   add interface group uplinks
   ```
3. Add an IPSLA group:
   ```
   add ipsla group Jitter-tests
   ```
4. Add device and super groups for the San Francisco campus network:
   ```
   add device group SFO-building1-fl1
   add device group SFO-building1-fl2
   add device group SFO-building1-fl3
   ...
   add device group SFO-building2-fl1
   add device group SFO-building2-fl2
   add device group SFO-building2-fl3
   ...
   add super  group SFO-building1
   add super  group SFO-building1
   add super  group SFO-campus
   ```
5. Add interface and super groups for the San Francisco campus network:
   ```
   add interface group SFO-building1-switch-uplinks
   add interface group SFO-building2-switch-uplinks
   add interface group SFO-building3-switch-uplinks
   ...
   add super group SFO-campus-switch-uplinks
   ```

6.2 count

The *count* command returns the number of entities in a group.

6.2.1 Syntax

```
count {type} group {group name}
```

6.2.2 Examples

1. Count the number of devices in the *Cisco* device group:
   ```
   count device group Cisco
   ```
2. Count the number of interfaces in the *Uplinks* interface group:
   ```
   count interface group Uplinks
   ```

6.3 delete

6.3.1 Syntax

```
delete {type} group {group name}
```

6.3.2 Examples

1. Delete a device group:
   ```
   delete device group core-routers
   ```
2. Delete an interface group:
   ```
   delete interface group uplinks
   ```
3. Delete a super group:
   ```
   delete super group SFO-Campus
   ```

6.4 rename

The group *rename* command allows you to rename a group and still keep all its associations.

6.4.1 Syntax

```
rename {type} group {group name from} {group name to}
```

6.4.2 Examples

1. Rename a profile group:
   ```
   rename profile group Support Network-Support
   ```
2. Rename a device group:
   ```
   rename device  group core-routers USA-Core-Routers
   ```
3. Rename an interface group:
   ```
   rename interface group uplinks Switch-Uplinks
   ```

6.5 prune

The *prune* command removes empty groups. This is typically used in the auto grouping.

6.5.1 Syntax

```
prune {type} group
```

6.5.2 Examples

1. Remove all empty device groups:
   ```
   prune device group
   ```
2. Remove all empty interface groups:
   ```
   prune interface group
   ```

6.6 assign

The *assign* command is for group assignments, typically used in the auto grouping section.

6.6.1 Syntax

```
assign group {group name} = {super group name}

assign {type} {parent regex} [{child regex} [{attribute regex}]]
       [value {text|integer|/regex/}]
       [profile {profile name}]
       [any|all|not group {group name} ...] = {target group} ...
```

6.6.2 Examples

1. Create some vendor groups and assign devices to them based on attribute values:
   ```
   add device group Cisco
   add device group Extreme
   assign * * sys SNMPv2-MIB.sysDescr  value /Cisco/      = Cisco
   assign * * sys SNMPv2-MIB.sysDescr  value /Cabletron/  = Extreme
   assign * * sys SNMPv2-MIB.sysDescr  value /Enterasys/  = Extreme
   assign * * sys SNMPv2-MIB.sysDescr  value /Extreme/    = Extreme
   ```

2. Put different Cisco models into groups using sysDescr:
   ```
   add device group Cisco-3600
   add device group Cisco-3700
   add device group Cisco-7200
   assign * * sys SNMPv2-MIB.sysDescr value "/Cisco IOS 3600/" = Cisco-3600
   assign * * sys SNMPv2-MIB.sysDescr value "/Cisco IOS 3700/" = Cisco-3700
   assign * * sys SNMPv2-MIB.sysDescr value "/Cisco IOS 7200/" = Cisco-7200
   ```

3. Put different Cisco models into groups using sysObjectID:
   ```
   add device group Cisco-3600
   add device group Cisco-3700
   add device group Cisco-7200
   assign * * sys SNMPv2-MIB.sysObjectID value /CISCO-PRODUCTS-MIB.cisco36/ = Cisco-3600
   assign * * sys SNMPv2-MIB.sysObjectID value /CISCO-PRODUCTS-MIB.catalyst37xxStack/ = Cisco-3700
   assign * * sys SNMPv2-MIB.sysObjectID value /CISCO-PRODUCTS-MIB.cisco72/ = Cisco-7200
   ```

4. Create some useful device groups:
   ```
   add device group routers
   add device group switches
   add device group core-routers
   add device group regional-routers
   add device group core-switches
   add device group regional-switches
   ```

5. Create some useful interface groups:
   ```
   add interface group core-links
   add interface group core-switch-uplinks
   add interface group regional-links
   add interface group servers
   ```

6. Assign all devices that end in "-rtr" to the routers group:
   ```
   assign device /-rtr\$/ = routers
   ```
7. Assign devices that start with "nyc-" to the group "NewYork":
   ```
   assign device /^nyc-/ = NewYork
   ```

8. Assign all serial interfaces to the serial-links group:
   ```
   add interface group serial-links
   assign interface * /^Se/ = serial-links
   ```

9. Assign all links with an ifAlias of "uplink" to a group:
```
add interface group Switch-Uplinks
assign interface * * IF-MIB.ifAlias value /uplink/ = Switch-Uplinks
```

10. Create a hierarchy from a building floor all the way up to a country
```
add device group central-office-fl1
assign device /central-office-fl1-.*/ = central-office-fl1

add device group central-office-fl2
assign device /central-office-fl2-.*/ = central-office-fl2

add super group central-office
assign group central-office-fl1 = central-office
assign group central-office-fl2 = central-office

add super group Brisbane
assign group central-office = Brisbane

add super group Queensland
assign group Brisbane = Queensland

add super group Australia
assign group Queensland = Australia
```

6.7 clear

The *clear* command is for removing group assignments, typically used in the auto grouping section.

6.7.1 Syntax

```
clear group {group name} = {super group name}

clear {type} {parent regex} [{child regex} [{attribute regex}]]
      [value {text|integer|/regex/}]
      [profile {profile name}]
      [any|all|not group {group name} ...] = {target group} ...
```

6.7.2 Examples

1. Add all Cisco devices to the *Cisco* group and then clear all 3560 switches from the group:
```
assign * * sys SNMPv2-MIB.sysDescr value /Cisco/ = Cisco
clear  * * sys SNMPv2-MIB.sysDescr value "/Cisco IOS C3560/" = Cisco
```

7 Profile Commands

7.1 assign

7.1.1 Syntax

```
assign group {group name} = {profile name}
assign user  {user name}  = {profile name}
```

7.1.2 Examples

1. Create a WAN-Tech-Support profile and allow access to all Cisco devices:

```
add profile group WAN-Tech-Support
add device  group WAN-Devices
assign * * sys SNMPv2-MIB.sysDescr value /Cisco/ = WAN-Devices

add report group WAN-Reports
assign * * config vendor value /cisco/ = WAN-Reports

assign group WAN-Devices = WAN-Tech-Support
assign group WAN-Reports = WAN-Tech-Support
```

2. Create a helpdesk profile, assign the profile to user *fred*, and allow access to all device which contain *-swt* or *-rtr* in their sysName:

```
add device group Helpdesk-Devices
assign * /-swt|-rtr/ = Helpdesk-Devices

add profile group Helpdesk-Profile
assign user fred = Helpdesk-Profile
assign group helpdesk-devices = Helpdesk-Profile
```

7.2 clear

7.2.1 Syntax

```
clear group {group name} = {profile name}
```

7.2.2 Examples

```
clear group WAN-Device = WAN-Tech-Support
```

8 Event Commands

8.1 Overview

- Event records are stored in daily chronological ordered blocks.
- Events older than three days are archived and compressed using AKIPS proprietary compression algorithms.
- Once an event has been archived into a compressed block, it can not be modified (eg. flags) or deleted.

An event record can be one of the following:

- Enumeration change
- Threshold above or below
- Uptime reset

The list of polled *enumerated* and *uptime* attributes are listed in the **Admin -> Status Alerts** help.

The list of polled *threshold* attributes are listed in the **Admin -> Threshold Alerts** help.

Each record in the event database includes the following information:

- Event time (seconds since epoch)
- Entity (ie. parent child attribute)
- Flags (comma seperated)
- Value
- Child description

Event flags are stored internally as a bit field. Multiple bits can be set simultaneously (eg. critical,enum). The following flags are defined:

- critical
- enum
- threshold
- uptime

Additional flags will be introduced in the future (eg. ack and suppress).

8.2 add

8.2.1 Syntax

```
add event {time} {parent name} {child name} {attribute name} {event flags} = {value}
add event {time} {parent name} {child name} {attribute name} = {value}
```

Notes:

- An entity (ie. parent child attribute) must exist before an event can be added.
- If {*time*} is set to zero, then the current time is used.
- Only the *critical* flag can be used in the *add* command. The event type (enum,threshold,uptime) is automatically resolved internally.

8.2.2 Examples

1. Add ifOperStatus down and up events:
   ```
   add event 0 swt250 Se0/1 IF-MIB.ifOperStatus = down
   add event 0 swt250 Se0/1 IF-MIB.ifOperStatus = up
   ```

2. Add Ping and SNMP down events:
   ```
   add event 0 swt250 ping4 PING.icmpState = down
   add event 0 swt250 ping6 PING.icmpState = down
   add event 0 swt250 sys   snmp.snmpState = down
   ```

3. Create a site specific admin state for a device that might be set by another system:
   ```
   add enum swt250 sys admin_state
   add event 0 swt250 sys admin_state = maintenance
   add event 0 swt250 sys admin_state = online
   ```

8.3 set

8.3.1 Syntax

```
set event {time} {parent name} {child name} {attribute name} = {event flags}
```

8.3.2 Examples

1. Set the *critical* flag on an existing ping down event:
   ```
   set event 1435822220 swt250 ping4 ping.icmpState = critical
   ```

8.4 clear

8.4.1 Syntax

```
clear event {time} {parent name} {child name} {attribute name} = {event flags}
```

8.4.2 Examples

1. Clear the *critical* flag on an existing ping down event:
   ```
   clear event 1435822220 swt250 ping4 ping.icmpState = critical
   ```

8.5 delete

8.5.1 Syntax

```
delete event {time} {parent name} {child name} {attribute name}
```

8.5.2 Examples

1. Delete a single down event:
   ```
   delete event 1435822225 swt250 ping4 PING.icmpState
   ```

8.6 mdelete

8.6.1 Syntax

```
mdelete event time {time filter}
        [{parent regex} [{child regex} [{attribute regex}]]]
        [profile {profile name}]
        [any|all|not group {group name} ...]
```

8.6.2 Examples

1. Delete all ping up/down events for a device that occurred in the last hour:
   ```
   mdelete event time last1h swt250 ping4 PING.icmpState
   ```

2. Delete all IF-MIB.ifOperStatus events for devices in the 'Edge-Switches' device group for yesterday:
   ```
   mdelete event time yesterday * * IF-MIB.ifOperStatus any group Edge-Switches
   ```

8.7 mget

The *mget* command retrieves matching event records in chronological order.

8.7.1 Syntax

```
mget event {all,critical,enum,threshold,uptime}
     time {time filter}
     [{parent regex} {child regex} {attribute regex}]
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

8.7.2 Return Values

Enumerated events:

```
{epoch} {parent} {child} {attribute} enum {flags} {value} [{child description}]
```

Uptime events:

```
{epoch} {parent} {child} {attribute} uptime {flags} {last uptime in seconds} [{child description}]
```

Threshold events:

```
{epoch} {parent} {child} {attribute} threshold {flags} {rule exceeded} [{child description}]
```

Note: The *child description* is only displayed if one has been set in the configuration. Some child technologies do not have an appropriate description.

8.7.3 Examples

1. Retrieve all ping outage events for the Columbus router over the last hour:

```
mget event enum time last1h Columbus-ro /ping/ *
1435894798 Columbus-ro ping4 PING.icmpState enum none down 10.4.1.22
1435894799 Columbus-ro ping6 PING.icmpState enum none down fd00:10:4:1::22
1435895128 Columbus-ro ping4 PING.icmpState enum none up 10.4.1.22
1435895129 Columbus-ro ping6 PING.icmpState enum none up fd00:10:4:1::22
```

2. Retrieve all ifOperStatus events for the last day:

```
mget event enum time last1d * * IF-MIB.ifOperStatus
1435846269 NewYork-ro Se2/2 IF-MIB.ifOperStatus enum none down Link to San Francisco
1435846509 NewYork-ro Se2/2 IF-MIB.ifOperStatus enum none up Link to San Francisco
1435848305 Chicago-ro Se1/4 IF-MIB.ifOperStatus enum none down Link to Dallas
1435848365 Chicago-ro Se1/4 IF-MIB.ifOperStatus enum none up Link to Dallas
...
```

3. Retrieve all sysUpTime resets:

```
mget event uptime time last1d
1435846486 Toronto-ro sys SNMPv2-MIB.sysUpTime uptime none 19044
1435848309 Columbus-ro sys SNMPv2-MIB.sysUpTime uptime none 7863
1435849250 Detroit-ro sys SNMPv2-MIB.sysUpTime uptime none 10363
1435849830 Cleveland-ro sys SNMPv2-MIB.sysUpTime uptime none 25704
...
```

4. Retrieve all the threshold events for the last hour:

```
mget event threshold time last1h
1436104800 cisco-74-1-19 cpu.2 CISCO-PROCESS-MIB.cpmCPUTotal1minRev threshold critical,above last5m,avg,60
1436104800 Chicago-ro ping4 PING.icmpRtt threshold critical,below last30m,avg,40000
1436104800 cisco-74-1-38 cpu.26 CISCO-PROCESS-MIB.cpmCPUTotal1minRev threshold critical,above last5m,avg,60
1436105101 SanFrancisco-ro ping4 PING.icmpRtt threshold critical,above last30m,avg,40000
1436105101 cisco-74-1-17 cpu.4 CISCO-PROCESS-MIB.cpmCPUTotal1minRev threshold critical,below last5m,avg,60
1436105101 cisco-74-1-29 cpu.2 CISCO-PROCESS-MIB.cpmCPUTotal1min threshold critical,above last5m,avg,60
1436105101 cisco-74-1-30 cpu.2 CISCO-PROCESS-MIB.cpmCPUTotal1min threshold critical,above last5m,avg,60
1436105101 NewYork-ro ping6 PING.icmpRtt threshold critical,above last30m,avg,40000
1436105101 NewYork-ro ping4 PING.icmpRtt threshold critical,above last30m,avg,40000
1436105101 Chicago-ro ping4 PING.icmpRtt threshold critical,above last30m,avg,40000
```

8.8 rget

The *rget* command is similar to *mget* but it returns all events in reverse chronological order.

8.8.1 Syntax

```
rget event time {time filter}
     [{parent regex} [{child regex} [{attribute regex}]]]
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

8.9 tget

The *tget* command returns time-series values for the number of events per *interval*. The output is in CSV format. This data is typically used to graph the number of events over time.

8.9.1 Syntax

```
tget event {all,critical,enum,threshold,uptime} {interval}
     time {time filter}
     [{parent regex} [{child regex} [{attribute regex}]]]
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

8.9.2 Examples

1. Retrieve time-series values for all ping events for yesterday in 1 hour intervals. A total of 24 values will be returned.

```
tget event all 3600 time yesterday * /ping/ *
506,426,460,458,440,760,315,301,232,421,332,288,196,299,380,381,495,448,497,570,386,430,362,53
```

9 Time-series

9.1 Features

- 3 years of historical data
- 60 second values (No rollup or averaging of stored data)
- Values stored in 30 day blocks using multi stage lossless data compression
- SHA-1 digest calculated on every data block for data integrity
- Time-series data types:
  - Counters
  - Gauges
  - RTT (microseconds)
- Rolling averages calculated for:
  - last 5 minutes
  - last 30 minutes
  - last 1 hour
  - last 8 hours
  - last 24 hours
  - last 48 hours
  - last 7 days
  - last 30 days

9.2 mget

The *mget* command is used by the threshold alerting to test if a lastN rolling average has gone above or below any of the user defined threshold rules. Refer to the help screen in the **Admin -> Threshold Alerting** for threshold examples.

9.2.1 Syntax

```
mget {lastN} avg|nonzero [above|below {value}[%]]
     [time {time filter}]
     {type} {parent regex} {child regex} {attribute regex}
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

9.2.2 Examples

1. Retrieve a list of devices in the *data-center* device group whose average Ping RTT values have gone above 10,000 microseconds (10ms) for the last hour:
   ```
   mget last1h avg above 10000 rtt * /ping/ * any group data-center
   ```

2. Retrieve a list of interfaces in the *wan-links* interface group whose utilisation percentage gone above 80% of the last 5 minuntes:
   ```
   mget last5m avg above 80% * * * /ifInUtil|ifOutUtil/ any group wan-links
   ```

3. Retrieve the list of Cisco devices whose CPU utilisation has gone above 60% in the last 5 minutes. In this example, the attribute is a regex because there two possible MIB object of *cpmCPUTotal1min* and *cpmCPUTotal1minRev*.
   ```
   mget last5m avg above 60 * * * /CISCO-PROCESS-MIB.cpmCPUTotal1min/
   ```

9.3 calc

The *calc* command calculates a single total, average or median value.

9.3.1 Syntax

```
calc  total|avg|median {NN}
      time {time filter}
      {type} {parent name} {child name} {attribute regex}
      [profile {profile name}]
      [any|all|not group {group name} ...]
```

Note: the attribute is a regex to cater for matching on things like ifInOctets and ifHCInOctets. Only one of these will exist in the database for any one interface.

9.3.2 Examples

1. Calculate the *total* InOctets for a single switch port for yesterday:
   ```
   calc total time yesterday counter swt250 Fa0/1 /InOctets/
   ```

2. Calculate the *average* InOctets for a single switch port for yesterday:
   ```
   calc avg time yesterday counter swt250 Fa0/1 /InOctets/
   ```

9.4 mcalc

The *mcalc* command calculates multiple total, average or median values.

9.4.1 Syntax

```
mcalc  total|avg|median {NN}
       time {time filter}
       {type} {parent regex} {child regex} {attribute regex}
       [profile {profile name}]
       [any|all|not group {group name} ...]
```

9.4.2 Examples

1. Calculate the total in and out octets for all interfaces in the wlan-links interface group:
   ```
   mcalc total time yesterday counter * * /^IF-MIB.*InOctets|^IF-MIB.*OutOctets/ any group wan-links
   Chicago-ro Se1/0 IF-MIB.ifInOctets = 90922240
   Chicago-ro Se1/0 IF-MIB.ifOutOctets = 112385280
   Chicago-ro Se1/1 IF-MIB.ifInOctets = 0
   Chicago-ro Se1/1 IF-MIB.ifOutOctets = 0
   Chicago-ro Se1/2 IF-MIB.ifInOctets = 11374963
   Chicago-ro Se1/2 IF-MIB.ifOutOctets = 10454049
   Chicago-ro Se1/3 IF-MIB.ifInOctets = 11409152
   ```

9.5 series

The series command returns time-series values for the selected time range.

9.5.1 Syntax

```
series  [interval total|avg {aggregate secs}]
        time {time filter}
        {type} {parent regex} {child regex} {attribute regex}
        [profile {profile name}]
        [any|all|not group {group name} ...]
```

9.5.2 Examples

1. Retrieve the total In/Out Octet values for an interface:
   ```
   series interval total 3600 time yesterday counter swt250 Fa0/1 /InOctets|OutOctets/
   swt250 Fa0/1 IF-MIB.ifHCInOctets = 2524672,2463360,2449920,2447488,2486656,2488064,2457856,...
   swt250 Fa0/1 IF-MIB.ifHCOutOctets = 975552,806976,803392,822016,802496,894144,820480,820608,...
   ```

9.6 cseries

The *cseries* command returns all data in CSV format, including the parent, child and attribute fields. The first line contains a descriptive header for each column.

9.6.1 Syntax

```
cseries  [interval total|avg {aggregate secs}]
         time {time filter}
         {type} {parent regex} {child regex} {attribute regex}
         [profile {profile name}]
         [any|all|not group {group name} ...]
```

9.6.2 Examples

1. Retrieve the total In/Out Octet values for an interface:

```
cseries interval total 3600 time yesterday counter swt250 Fa0/1 /InOctets|OutOctets/
parent,child,child description,attribute,2018-07-05 00:00:00,2018-07-05 01:00:00,...
swt250,Fa0/1,Link to swt251,IF-MIB.ifHCInOctets,2524672,2463360,2449920,2447488,2486656,...
swt250,Fa0/1,Link to swt251,IF-MIB.ifHCOutOctets,975552,806976,803392,822016,802496,894144,...
```

9.7 top

9.7.1 Syntax

```
top  {N} [reverse] [interval avg {secs}] total|max|avg|median {NN}
     time {time filter}
     {type} {parent regex} {child regex} {attribute regex}
     [profile {profile name}]
     [any|all|not group {group name} ...]
```

9.7.2 Performance

When the time filter is one of the lastN types, the data is retrieved from the rolling averages, otherwise the top command needs to do large amounts data mining.

9.7.3 Examples

1. Retrieve the interfaces with the highest In/Out Errors for the last 5 minutes. This will be quick because it is accessing the rolling average data values.

```
top 20 total time last5m counter * * /IF-MIB.if.*Errors/
```

2. Retrieve the interfaces with the highest In/Out Octets for yesterday. This will be slow because the command has to do massive amounts of data mining, extracting the totals for every monitored interface so it can calculate the top 20.

```
top 20 total time yesterday counter * * /IF-MIB.if.*Octets/
cisco-74-1-12 Te1/1 IF-MIB.ifHCInOctets = 134374989234176
cisco-74-1-16 Te1/1/1 IF-MIB.ifHCOutOctets = 92564298924032
juniper-74-2-7 xe-0/0/10.0 IF-MIB.ifHCInOctets = 92329710977024
juniper-74-2-12 ae47.0 IF-MIB.ifHCOutOctets = 92308460535808
cisco-74-1-24 Ethernet1/23 IF-MIB.ifHCOutOctets = 89412016799744
cisco-74-1-24 Ethernet1/24 IF-MIB.ifHCOutOctets = 88919019356160
juniper-74-2-10 reth0.2213 IF-MIB.ifHCOutOctets = 88832066191360
arista-74-0-40 Ethernet5/6 IF-MIB.ifHCInOctets = 88685106167808
cisco-74-1-17 Po55.152 IF-MIB.ifHCOutOctets = 88431959736320
...
```

10 Config and Events Web API

10.1 Overview

The **Config and Events** Web API allows you to get/set data using the same API syntax described in the sections above, the only difference being a transport mechanism of HTTP/HTTPS instead of SSH or via the Command Console.

To activate the Config and Events Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Config and Events** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-rw** with an appropriate password.

10.2 Syntax

```
http://{server}/api-db?password={pw};cmds={query}
```

- Replace *server* and *pw* with appropriate values.
- For GET requests, *query* typically needs to be URL encoded (e.g. spaces turned into +).

10.3 Using Curl

Curl is typical of many open source programs - it has a ridiculous number of command line options.
These are some useful options:

Option	Comment
-s	Silent mode, to suppress annoying HTTP headers being printed
-k	Insecure mode, to suppress error messages about self signed certificates
–data-binary	Send POST data in binary form - don't strip NL characters !!
–compressed	Tell the HTTP server to compress the output (e.g. gzip)
@{filename}	Read POST data from {filename}
@-	Read POST data from stdin

NOTE: Curl has bizarre behaviour when doing POST requests where it strips newline characters if you don't use the –data-binary option. You will need to use this option when sending multi line content, as per the example below.

10.4 GET Examples

1. Retrieve a list of devices:
   ```
   curl -s "http://{server}/api-db?password={pw};cmds=mget+device+*"
   ```
2. Retrieve a list of unreachable IPv4 Ping devices:
   ```
   curl -s "http://{server}/api-db?password={pw};cmds=mget+*+*+ping4+PING.icmpState+value+/down/"
   ```

10.5 POST Examples

1. Retrieve a list of devices:
   ```
   curl -s -d "cmds=mget device *" "http://{server}/api-db?password={pw}"
   ```
2. Retrieve a list of unreachable IPv4 Ping devices:
   ```
   curl -s -d "cmds=mget * * ping4 PING.icmpState value /down/" "http://{server}/api-db?password={pw}"
   ```
3. Run a bunch of commands from a file
   ```
   curl -s --data-binary @commands "http://{server}/api-db?password={pw}"
   ```
   where the file called *commands* contains ADB syntax
   ```
   cmds=
   mget device *
   mget device * any group cisco
   mget * * ping4 PING.icmpState value /down/
   ```
4. Run a bunch of commands reading from stdin
   ```
   cat commands | curl -s --data-binary @- "http://{server}/api-db?password={pw}"
   ```

11 Syslog / Trap Reporter

11.1 Overview

All Syslog and SNMP Traps are stored in a single message database. The **nm-msg-reporter** command line tool can be used to extract and filter messages from the database. *nm-msg-reporter* takes arguments from either the command line or via stdin.

11.2 Command Line Syntax

```
nm-msg-reporter
 time {filter}
 [type {syslog|trap}]
 [addr {filter}]
 [limit {num}]
 [regex {filter}]
 [mute {file}]
 [trim {file}]
 [opt stats]
 [interval {sec}]
```

11.3 Output Format

Each syslog or trap message contains:

1. Header line
   `{epoch time}  {type}  {IP version}  {IP Address}`
2. Message text
3. Blank terminating line

Example syslog message:

```
1436223614 syslog 4 10.4.2.26
notice local7 134: Jul 7 09:00:12.772: OSPF-5-ADJCHG: Process 1, Nbr 10.4.49.1 on Serial2/2 ...
```

Example SNMP Trap message:

```
1436223756 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 45484
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier CISCO-SYSLOG-MIB.clogMessageGenerated
CISCO-SYSLOG-MIB clogHistFacility 111 DisplayString OSPF
CISCO-SYSLOG-MIB clogHistSeverity 111 ENUM 6,notice
CISCO-SYSLOG-MIB clogHistMsgName 111 DisplayString ADJCHG
CISCO-SYSLOG-MIB clogHistMsgText 111 DisplayString Process 1, Nbr 10.4.49.1 on Serial2/2 ...
CISCO-SYSLOG-MIB clogHistTimestamp 111 TimeTicks 45484
```

11.4 Examples

1. Retrieve all syslog messages for the last hour:

```
time last1h type syslog
1436229532 syslog 4 10.4.2.130
error local7 67: *Mar 3 09:22:52.269: SYS-3-CPUHOG: Task is running for (3180)msecs, more ...

1436229532 syslog 4 10.4.2.130
error local7 68: -Traceback= 0x60C35A94 0x60C35C84 0x60C34688 0x60C34B84 0x60C38BC8 0x60C2950C ...

1436229532 syslog 4 10.4.2.130
error local7 69: *Mar 3 09:22:53.109: SYS-3-CPUHOG: Task is running for (4024)msecs, more ...

1436229532 syslog 4 10.4.2.130
error local7 70: -Traceback= 0x60C35A98 0x60C35C84 0x60C34688 0x60C34B84 0x60C38BC8 0x60C2950C ...
```

2. Retrieve all trap messages for the last day which contain the string 'OSPF':

```
type trap time last1d regex OSPF
1436232075 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 53803
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier CISCO-SYSLOG-MIB.clogMessageGenerated
CISCO-SYSLOG-MIB clogHistFacility 122 DisplayString OSPFv3
CISCO-SYSLOG-MIB clogHistSeverity 122 ENUM 6,notice
CISCO-SYSLOG-MIB clogHistMsgName 122 DisplayString ADJCHG
CISCO-SYSLOG-MIB clogHistMsgText 122 DisplayString Process 1, Nbr 10.4.45.1 on Serial1/6 from ...
CISCO-SYSLOG-MIB clogHistTimestamp 122 TimeTicks 53803

1436232075 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 53803
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier OSPF-TRAP-MIB.ospfNbrStateChange
OSPF-MIB ospfRouterId ??? IPAddress 10.4.40.1
OSPF-MIB ospfNbrIpAddr ??? IPAddress 10.4.2.166
OSPF-MIB ospfNbrAddressLessIndex ??? Integer 0
OSPF-MIB ospfNbrRtrId ??? IPAddress 10.4.45.1
OSPF-MIB ospfNbrState ??? ENUM 1,down
```

3. Retrieve all OSPF traps for yesterday, and report time-series values in 1 hour intervals. The time-series values are typically useful for graphing messages over time.

```
time yesterday type trap regex OSPF opt stats interval 3600
...
1436190031 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 11722
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier CISCO-SYSLOG-MIB.clogMessageGenerated
CISCO-SYSLOG-MIB clogHistFacility 83 DisplayString OSPF
CISCO-SYSLOG-MIB clogHistSeverity 83 ENUM 6,notice
CISCO-SYSLOG-MIB clogHistMsgName 83 DisplayString ADJCHG
CISCO-SYSLOG-MIB clogHistMsgText 83 DisplayString Process 1, Nbr 10.1.8.170 on Serial1/0 from LOADING ...
CISCO-SYSLOG-MIB clogHistTimestamp 83 TimeTicks 11722

#total,1436104800,3600,78,39,117,1817,105,94,86,67,73,69,156,73,101,70,74,98,107,43,203,105,147,115,71,131
#match,1436104800,3600,15,0,35,15,32,25,20,15,15,16,84,16,30,15,15,15,35,0,110,30,77,35,16,16
#syslog,1436104800,3600,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
#trap,1436104800,3600,15,0,35,15,32,25,20,15,15,16,84,16,30,15,15,15,35,0,110,30,77,35,16,16
```

4. Retrieve all syslog and traps for today from 10.4.2.26

```
time today addr 10.4.2.26
1436232275 syslog 4 10.4.2.26
notice local7 149: Jul 7 11:24:34.476: LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/6, changed...

1436232275 syslog 4 10.4.2.26
notice local7 150: Jul 7 11:24:34.572: OSPF-5-ADJCHG: Process 1, Nbr 10.4.45.1 on Serial1/6 from LOADING...

1436232275 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 54003
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier OSPF-TRAP-MIB.ospfNbrStateChange
OSPF-MIB ospfRouterId ??? IPAddress 10.4.40.1
OSPF-MIB ospfNbrIpAddr ??? IPAddress 10.4.2.166
OSPF-MIB ospfNbrAddressLessIndex ??? Integer 0
OSPF-MIB ospfNbrRtrId ??? IPAddress 10.4.45.1
OSPF-MIB ospfNbrState ??? ENUM 8,full

1436232276 trap 4 10.4.2.26
SNMPv2-MIB sysUpTime 0 TimeTicks 54004
SNMPv2-MIB snmpTrapOID 0 ObjectIdentifier OSPF-TRAP-MIB.ospfOriginateLsa
OSPF-MIB ospfRouterId ??? IPAddress 10.4.40.1
OSPF-MIB ospfLsdbAreaId ??? IPAddress 0.0.0.0
OSPF-MIB ospfLsdbType ??? ENUM 1,routerLink
```

11.5 Web API

11.5.1 Overview

This is a Web API wrapper around the nm-msg-reporter program.
To activate the Syslog and Traps Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Syslog and Traps** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-ro** with an appropriate password.

11.5.2 Syntax

`https://{server}/api-msg?password={pw};{option}={value};...`

Option	Value	Required	Default
time	time filter	required	
addr	ip filter	optional	
type	syslog|trap	optional	both
device	name|regex	optional	
regex	regex filter	optional	
limit	num messages	optional	

11.5.3 Examples

1. Retrieve all syslog and trap messages for the last 30 minutes
   `http://{server}/api-msg?password={pw};time=last30m`
2. Retrieve all syslog messages for devices which match a regex for the last 30 minutes
   `http://{server}/api-msg?password={pw};time=last30m;type=syslog;device=/^swt/;`
3. Retrieve all syslog messages for devices in the *core-routers* device group for the last 1 hour
   `http://{server}/api-msg?password={pw};time=last1h;type=syslog;group=core-routers;`

12 Netflow Reporter

12.1 Overview

Netflow records are stored in a dedicated database. The **nm-flow-reporter** command line tool can be used to query the database and generate CSV output. *nm-flow-reporter* takes arguments from the command line.

12.2 Command Line Syntax

```
nm-flow-reporter [options] time {timefilter} meter {meter}
    [src|dst|any|both {IP filter}]
    [limit {num}]
    [proto {protocol.service}]
    [show src,dst,pro,pkt,oct,flo,con]
    [sort pkt|oct|flo|con]

Flow Meter Sources:
    ...
```

12.3 Show Flags

- src - v4/v6 source address
- dst - v4/v6 destination address
- pro - protocols
- pkt - packets
- oct - octets
- flo - flows
- con - number of conversations
- tsf - filter that can be passed to *nm-flow-timeseries*
- url - URL (useful for building web interface drilldowns)

12.4 Sort Flags

By default, the output is sorted by:

1. source address
2. destinations address
3. protocol

The sort order can be changed by specifying one of the following flags:

1. pkt - packets
2. oct - octets
3. flo - flows
4. con - conversations

12.5 Output Format

The first line of output contains a CSV formatted header with 13 fields that describe the fields displayed in the data.

```
#Source,Destination,Protocol,Packets,Bytes,Flows,Conversations,{tsf},{url},{ms},{processed},{matched},{results}
```

1. Source
2. Destination
3. Protocol
4. Packets
5. Bytes
6. Flows

7. Conversations
8. Filter to pass to *nm-flow-timeseries*
9. URL (useful for building web interfaces)
10. Milliseconds it took to process the query
11. Records processed
12. Records the query matched
13. Records included in the results

12.6 Examples

1. Retrieve the top 10 protocols over the last 1 hour sorted by bytes:

```
time last1h show pro,pkt,oct sort oct limit 10 meter 172.16.1.17
#,,Protocol,Packets,Bytes,,,,,22,158011,158011,77
,,tcp.http,3550502,2445296132,,,,
,,udp.unknown,3101492,613100963,,,,
,,tcp.unknown,1319277,556558287,,,,
,,tcp.mail,540574,432762436,,,,
,,tcp.https,1248692,348540628,,,,
,,tcp.pop-3,139209,119557565,,,,
,,tcp.netbios-ssn,173795,48857000,,,,
,,esp.unknown,161027,42902572,,,,
,,tcp.microsoft-ds,699756,42271680,,,,
,,udp.domain,442996,38515564,,,,
```

2. Retrieve the top 5 talkers over the last 10 minutes sorted by bytes:

```
time last10m show src,pro,pkt,oct sort oct limit 5 meter 172.16.1.17
#Source,,Protocol,Packets,Bytes,,,,,5,26598,26598,4804
157.187.62.203,,tcp.mail,69178,97197086,,,,
47.42.125.202,,udp.unknown,217449,68615167,,,,
124.61.150.202,,tcp.http,26486,32968775,,,,
118.28.194.173,,tcp.http,19610,29042465,,,,
35.46.125.202,,tcp.unknown,24522,18796489,,,,
```

3. Retrieve the top 5 listeners for last week in business hours sorted by bytes:

```
time "lastweek; sun to sat 8:00 to 17:00" show dst,pro,pkt,oct sort oct limit 5 meter 172.16.1.17
#,Destination,Protocol,Packets,Bytes,,,,,3860,8714743,8714743,481960
,2.94.18.203,tcp.mail,9415528,12912326241,,,,
,1.8.7.10,udp.unknown,44253976,8076854931,,,,
,51.184.62.203,tcp.unknown,4671831,6935860116,,,,
,25.174.55.203,tcp.http,4968006,6354019143,,,,
,228.174.55.203,tcp.unknown,3663764,3940752196,,,,
```

4. Retrieve all conversations with a source address of 157.187.62.203

```
time today src 157.187.62.203 show src,dst,pkt,oct meter 172.16.1.17
#Source,Destination,,Packets,Bytes,,,,,183,2664465,1217,70
157.187.62.203,1.146.107.175,,4540,198515,,,,
157.187.62.203,2.1.1.10,,98,8022,,,,
157.187.62.203,2.94.18.203,,1111526,1561651897,,,,
157.187.62.203,4.4.130.139,,1134,100331,,,,
157.187.62.203,4.27.104.112,,3,120,,,,
157.187.62.203,4.150.241.222,,1,40,,,,
157.187.62.203,4.212.113.140,,1,40,,,,
157.187.62.203,10.144.193.204,,5,675,,,,
157.187.62.203,11.144.193.204,,10,942,,,,
157.187.62.203,14.208.15.173,,3,120,,,,
157.187.62.203,15.144.193.204,,10,1786,,,,
157.187.62.203,16.19.179.24,,2,80,,,,
...
```

12.7 Web API

12.7.1 Overview

This is a Web API wrapper around the nm-flow-reporter program.
To activate the Netflow Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Netflow** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-ro** with an appropriate password.

12.7.2 Syntax

```
https://{server}/api-flow?password={pw};{option}={value};...
```

Option	Value	Default
meter	exporter IP	
time	time filter	Increments of 5 minutes
show	src,dst,pro,pkt,oct,flo,con	Multiple
sort	src|dst|pro|pkt|oct|flo|con	One only
proto	protocol.service	udp.snmp
limit	number	1 to ...
src|dst|any|both	ipaddr	Multiple

12.7.3 Examples

1. Retrieve TopN protocol data for packets/octets, sorted on octets.
   ```
   http://{server}/api-flow?password={pw};meter=10.0.0.254;time=last10m;show=pro,pkt,oct;sort=oct
   ```

13 Netflow Time-Series

The **nm-flow-timeseries** command is used to extract time-series values for:

- Packets
- Bytes
- Bits per second
- Flows

This data is typically used when creating time-series Netflow graphs.

13.1 Command Line Syntax

```
nm-flow-timeseries [options] time {timefilter} meter {meter}
   interval [minutes]
   src|dst|any|both {addr}
   proto {filter}
```

interval must be in 5 minute increments (eg. 5, 10, 30, 60, etc).

The number of CSV time-series values is calculated by the time span divided by the interval. For example, a time filter of *today* and interval of 60 minutes will produce 24 time-series values.

13.2 Output Format

Each query results in four lines of CSV output that may contain the following fields:

1. source address
2. destination address
3. protocol
4. type (ie. Pkts, Bytes, Bits or Flows)
5. start time (seconds since epoch)
6. array of time-series values in the requested interval

13.3 Examples

1. `time last1h src 157.187.62.203 interval 10 meter 172.16.1.17`

```
157.187.62.203,,,,,Pkts,1436243100,600,69970,1236,44698,45,38252,33,21
157.187.62.203,,,,,Bytes,1436243100,600,98279249,134122,62765460,4296,53683296,3352,2009
157.187.62.203,,,,,Bits,1436243100,600,786233992,1072976,502123680,34368,429466368,26816,16072
157.187.62.203,,,,,Flows,1436243100,600,32,41,30,45,38,31,21
```

2. `time last1h src 10.1.8.62 proto udp.snmp interval 5 meter 10.4.2.22`

```
10.1.8.62,,,,udp.snmp,Pkts,1436244300,300,126,114,125,128,495,125,128,106,131,128,121,0,52
10.1.8.62,,,,udp.snmp,Bytes,1436244300,300,76605,68846,75615,77655,120197,75738,77745,64000,79708,...
10.1.8.62,,,,udp.snmp,Bits,1436244300,300,612840,550768,604920,621240,961576,605904,621960,512000,...
10.1.8.62,,,,udp.snmp,Flows,1436244300,300,49,51,50,50,140,50,50,48,52,50,22,0,16
```

13.4 Web API

13.4.1 Overview

This is a Web API wrapper around the nm-flow-timeseries program.
To activate the Netflow Time-series Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Netflow Time-series** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-ro** with an appropriate password.

13.4.2 Syntax

`http://{server}/api-flow-timeseries?password={pw};{option}={value};...`

Option	Value	Default
meter	exporter IP	
time	time filter	Increments of 5 minutes
interval	increments of 5 min	5, 10, 15, 20, ... 60, ... 120, etc
proto	protocol.service	udp.snmp
src|dst|any|both	ipaddr	Multiple

13.4.3 Examples

1. Retrieve 5 minute time-series values for all flow records on exporter 10.0.0.254
   `http://{server}/api-flow-timeseries?password={pw};meter=10.0.0.254;time=last4h;interval=5`

14 Switch Port Mapper

14.1 Overview

Switch Port Mapper data is stored in CSV format and can be accessed via:

- Command line
- System Log Viewer
- Web API

14.2 Raw File Formats

The following table describes the format of the Switch Port Mapper CSV data files, which are located on the AKIPS server in */home/akips/data/spm*

Filename	Description	Output Format (CSV)
arp	ARP Tables	epoch,ipaddress,mac
ip2mac	IP to MAC mapping	epoch,mac,ipaddress
mac2ip	MAC to IP mapping	epoch,mac,ipaddress
mac2sp	MAC to Switch Port mapping	epoch,mac,switch,port
mac2vspa	MAC to Vendor, Switch, Port, IP Address	mac,vendor,switch,port,ipaddress
sp2mac	Switch Port to MAC	switch,port,mac
sp2vlan	Switch Port to VLAN	epoch,switch,port,vlan
vlan2sp	VLAN to Switch Port	epoch,vlan,switch,port
vlans	List of all VLANs	vlan

14.3 Web API

The Web API is used to retrieve the raw switch port mapper files.
To activate the Switch Port Mapper Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Switch Port Mapper** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-ro** with an appropriate password.

14.4 Syntax

```
http://{server}/api-spm?password={pw};filename={file}
```

14.5 Examples

```
http://{server}/api-spm?password={pw};filename=mac2vspa
```

15 Unused Interfaces

15.1 Overview

To activate the Unused Interfaces Web API:

1. Go to **Admin -> API -> API Settings** and turn on the **Unused Interfaces** option.
2. Go to the **Admin -> User/Profile -> User Settings** and add a user name of **api-ro** with an appropriate password.

15.2 Syntax

```
http://{server}/api-unused-interfaces?password={pw};{option}={value};...
```

Option	Default	Description
time	last30d	
device	*	Device Regex
group		Device Group

15.3 CSV Output Format

Field	Value
1	Device Name
2	Interface Name
3	Interface Speed
4	Used | Free
5	Current IF-MIB.ifOperStatus
6	Last change time (epoch seconds)
7	IF-MIB.ifAlias
8	VLAN Name

15.4 Examples

1. Retrieve data for device swt251
   `http://{server}/api-unused-interfaces?password={pw};device=swt251`
2. Retrieve data for all devices in the *cisco* device group
   `http://{server}/api-unused-interfaces?password={pw};group=cisco`

16 TCP Socket API

16.1 Overview

The TCP socket daemon is useful for direct remote access to the following programs:

- nm-db
- nm-msg-reporter

NOTE: There is no authentication performed by the TCP socket daemon, therefore it should only be used in a secure network environment

16.2 Syntax

```
{program} {socket number} [{Restrict IP Address}]
```

16.3 Examples

1. Start a nm-db process listening on TCP port 3000
   ```
   nm-db 3000
   ```
2. Start a nm-db process listening on TCP port 3001 and only allow connections from 10.0.0.50
   ```
   nm-db 3001 10.0.0.50
   ```
3. Start a nm-msg-reporter process listening on TCP port 3002 and only allow connections from 10.0.0.51
   ```
   nm-msg-reporter 3002 10.0.0.51
   ```

17 Perl Module: Common

This module can be found at */usr/local/akips/pm/Akips/Common.pm*. It is worth looking at the available definitions and externally visible functions.

17.1 Useful array definitions

```
@weekday_names_short = (
   "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"
);

@weekday_names_long = (
   "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"
);

@month_names_short = (
   "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
);

@month_names_long = (
   "January", "February", "March", "April", "May", "June",
   "July", "August", "September", "October", "November", "December"
);

@days_of_month = (
   "1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th", "9th", "10th",
   "11th", "12th", "13th", "14th", "15th", "16th", "17th", "18th", "19th", "20th",
   "21st", "22nd", "23rd", "24th", "25th", "26th", "27th", "28th", "29th", "30th", "31st"
);

@days_in_month = ( 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 );


@metric_prefix = (
   'p', 'n', 'u', 'm', '', 'K', 'M', 'G', 'T', 'P', 'E'
);

@binary_prefix = (
   '', 'K', 'M', 'G', 'T', 'P', 'E'
);
```

17.2 BEGIN and END

By including the *Akips::Common* module, the following is aways executed:

BEGIN:

- Process title is set to a meaningful value
- Changes the default WARN and DIE functions to PRINT_LINE and EXIT_FATAL

END:

- Closes all lock files
- Closes the error logging socket
- Logs an exit message if *process_init()* was called with *PROC_LOG => 1*

17.3 PRINT_LINE

The *PRINT_LINE* function is useful when debugging. It prints the following on a single line to STDERR:

- Date/Time
- Process name
- Function name
- Filename
- Line number
- Error message

17.4 process_init ()

process_init() is typically called at the begining of a Perl script to setup various useful things.

These include:

- Changes to the *akips* Unix user and group
- Sets process title to a meaningful value
- Sets up signal handling for SIGINT, SIGTERM, SIGPIPE, etc
- Blocks signals
- Creates and checks a lock file
- Opens the AKIPS error logging socket
- Logs start/end messages

The options to *process_init()* include:

- title => {program title}
- lockfile => {path to lock file}
- PROC_LOG => 1
  Log start/end messages to the AKIPS log files.
- PROC_NOSETUID => 1
  Don't call setuid() to switch to the *akips* user and group ID

17.5 process_signals ()

Unblocks and then blocks signals. This allows any pending signals to be processed.

17.6 process_lock ()

```
process_lock ( [lock filename], [lock flags] )
```

Under the hood, process_lock() uses the standard Perl flock() function. The default lock flags are LOCK_NB and LOCK_EX (i.e. a non-blocking exclusive lock). To get a blocking exclusive lock, just pass in the LOCK_EX flag.

17.7 errlog ()

AKIPS uses its own error logging functionality.

```
errlog ({LOG_LEVEL}, {log message});
```

LOG_LEVELS:

- $ERR_FATAL
- $ERR_ERROR
- $ERR_WARNING
- $ERR_INFO
- $ERR_USER
- $ERR_DEBUG
- $ERR_CGI

Example:

```
my $error_msg = "it broke";
errlog ($ERR_FATAL, "something went wrong - %s", $error_msg);
```

All error messages can be viewed in the *Admin -> System -> System Log Viewer*.

17.8 trim ()

trim strips leading and trailing whitespace from a string.

Example:

```
my $str = "  some text  ";
$str = trim ($str);
```

17.9 valid_regex ()

Validates a regular expression syntax string.
Returns 0 on failure and 1 on success.

Example:

```
my $regex_str = "some regex string";
if (validate (regex_str) {
   do something
}
```

17.10 get_localtime (epoch)

This is a convenient wrapper around the standard localtime() function.
Optional passed in *epoch*, otherwise current epoch time is used.
get_localtime() returns a hash reference with the following populated.

```
$hash{sec}
$hash{min}
$hash{hour}
$hash{mday}   - day number of the month
$hash{mon}    - month 0 - 11
$hash{year}
$hash{wday}   - weekday 0 - 6
$hash{yday}   - day number of the year
$hash{isdst}  - is daylight savings time
```

Example:

```
my $h_ref;
$h_ref = get_localtime ();
printf STDOUT "%s %s %s",
       $days_of_month[$h->{mday} - 1],
       $month_names_long[$h->{mon}],
       $h->{year};
```

17.11 mail ()

The *mail()* function takes a hash reference which contains:

- to
- subject
- body

Example:

```
my @body = (
          "Hello,",
          "Greetings from AKIPS!",
          "Sincerely",
          "Bob"
);

@files = ("/path/to/graph1.png", "/path/to/report1.pdf", "/path/to/report2.pdf");

mail ({
        to      => 'admin@example.com',
        subject => 'Greetings',
        body    => \@body,
        attach  => \@files,  #(optional)
    });
```

17.12 syslog ()

The *syslog()* function takes a hash reference which contains:

- ipaddr (Destination IP address - syslog collector)
- priority (optional, default: notice)
- facility (optional, default: user)

- message

priority must be one of:

- emergency
- alert
- critical
- error
- warning
- notice
- info
- debug

facility must be one of:

- auth
- authpriv
- console
- cron
- daemon
- ftp
- kern
- lpr
- mail
- news
- ntp
- security
- syslog
- user
- uucp
- local0
- local1
- local2
- local3
- local4
- local5
- local6
- local7

Example:

```
syslog ({
          ipaddr   => "10.50.1.100",
          priority => "error",
          facility => "local3",
          message  => "Quick brown fox has jumped",
       });
```

17.13 http_send() and http_result()

The *http_xxxx* functions are a wrapper around the **nm-http** command line tool, which is an HTTP client similar to Curl.
http_send() performs a http/https request and returns NO output.
http_result() performs a http/https request and returns the response data.

Both functions take a hash reference which contains the following:

Name	Value	Default	Comment
url	http://www.example.com	none	http or https, port number optional
method	GET or POST	none	Case insensitive
content_type	text/html	application/x-www-form-urlencoded	Whatever the remote system is expecting
headers	see *Example 3*	none	Send custom HTTP headers
secure	0 or 1	1	0 = allow self signed certificate
data	post data		

17.13.1 Example 1

```
my $args;

$args{url}          = "http://www.example.com/",
$args{method}       = "get",
$args{content_type} = "text/html",

http_send (\%args);
http_close ();
```

17.13.2 Example 2

```
http_send ({
   url          => "http://www.example.com:8080/",
   method       => "GET",
   content_type => "text/html",
});
```

17.13.3 Example 3

```
my @result = http_result ({
                url          => "https://www.example.com/script?key=val",
                method       => "GET",
                secure       => 0,
                content_type => "text/html",
                headers      => [ "Authorization: MyAPIKey 01234-abcde", ],
             });

for my $line (@result) {
   printf "%s\n", $line;
}
```

18 Perl Module: ADB

The command line tool for the AKIPS database is **nm-db**. The ADB module simplifies interacting with the *nm-db* program by automatically opening and closing a bidirectional pipe when the module is loaded and at script exit.

The following functions are provided:

- adb_flush ();
- adb_send ();
- adb_result ();

18.1 adb_flush ()

Use *adb_flush()* to flush all buffered output to *nm-db* This is required after calling *adb_send()* one or more times.

18.2 adb_send ()

The *adb_send()* function writes the specified command to *nm-db*.

```
adb_send ("add device group CoreRouters");
adb_send ("add device group CoreSwitches");
adb_flush ();
```

18.3 adb_result ()

adb_result() writes the command to *nm-db* and returns the results in either a scalar or array. No need to call adb_flush().

Example:

```
#! /usr/local/bin/perl

use warnings;
use strict;

use Akips::ADB;
use Akips::Common;

sub main
{
   my $val;
   my @arr;

   #
   # Retrieve a single value
   #
   $val = adb_result ("get swt250 sys SNMPv2-MIB.sysLocation");
   printf STDOUT "Got %s\n", $val;

   #
   # Retrieve lots of results and load into an array
   #
   @arr = adb_result ("mget device *");
   for my $line (@arr) {
      printf ("%s\n", $line);
   }

   return;
}

main ();

exit 0;
```

19 Perl Module: Discover

In version 17.x, the discover functions have been simplified to reduce the complexity of doing custom site integrations scripting.

19.1 discover_scan()

discover_scan() performs the first stage of device discovery, and creates the necessary intermediate files for the *discover_config()* stage.

19.1.1 Examples

Scan three IP ranges using the existing discover SNMP credentials:

```
discover_scan (undef, "10.0.0.0/16", "10.3.0.0/24", "10.5.0.0/24");
```

Scan one IP address using the specified SNMPv2 credentials:

```
discover_scan ("version 2 community foobar", "10.0.0.1");
```

Scan multiple IP ranges using the specified SNMPv2 credentials:

```
discover_scan ("version 2 community foobar", "10.0.0.0/24", "10.1.0.0/24");
```

Scan one IP address using the specified SNMPv3 credentials:

```
discover_scan ("version 3 sha passwd1 aes256 passwd1", "10.2.0.1"");
```

19.1.2 Output Files

discover_scan() creates the following output files in the *data/discover/* directory:

ping.scan
List of IPv4 and IPv6 addresses that responded to a ping scan.

- IP Address

snmp.scan
Output of a snmp walk of SNMPv2-MIB.system for each device found in the ping scan stage.

devices

- IPv4 Address
- IPv6 Address
- SNMPv2-MIB.sysName
- SNMP credentials
- SNMPv2-MIB.sysObjectID
- SNMPv2-MIB.sysDescr

19.2 discover_config

discover_config() does SNMP walks of each device discovered by the *discover_scan()*, then processes the walk out and creates the necessary device configuration.

19.2.1 Syntax

```
discover_scan ( {SNMP Parameters}, {IP Address Range}, ... );
```

19.2.2 Example

```
discover_scan ("version 2 community foobar", "10.0.0.1");
discover_config ();
```

19.3 discover_device_rewalk

19.3.1 Syntax

```
discover_device_rewalk (\%hash_ref);
```

Name	Value	Default	Comment
ipaddr	IPv or IPv6 address	none	IP Address to rewalk

19.3.2 Examples

Rewalk a single device. Returns 0 on failure, or nonzero on success.

```
discover_device_rewalk ({
   ipaddr      => "10.1.2.3",                  # IP address to rewalk
   device      => "atlanta-ro",                # Device name
   snmp_param  => "version 2 community public", # Same format as snmp.cfg
   sysObjectID => "...",                       # SNMPv2-MIB.sysObjectID (optional)
   sysDescr    => "...",                       # SNMPv2-MIB.sysDescr    (optional)
});
```

20 Perl Scripting Guide

- AKIPS is mostly written in C and Perl. All the high performance and low level programs (eg. databases, poller, reporters, etc) are written in C, and Perl is generally used for everything else.
- Perl is NOT part of the FreeBSD base system. It is an additional package which gets installed in /usr/local.
- AKIPS currently uses Perl version 5.22.
- We highly recommend the *O'Reilly Perl Best Practices* book.
- Every AKIPS Perl script is validated using Perl-Critic - Critique Perl source code for best-practices. Perl-Critic is typically used in conjunction with the *Perl Best Practices* book.

20.1 Basic Structure

```
#! /usr/local/bin/perl

use warnings;
use strict;
use feature qw(switch);
no warnings "experimental::smartmatch";

# AKIPS specific Modules
use Akips::ADB;
use Akips::Common;

sub funct1
{
   # some code...
   return;
}

sub main
{
   process_init ();

   # code to do stuff...
   funct1 ();

   # Log a message
   errlog ($ERR_DEBUG, "doing some stuff");

   return;
}

main ();

exit 0;
```