ACC: Unified syntax for trigger expressions, calculated and aggregated checks

v0.16

Summary

Currently Zabbix uses different syntax for triggers, calculated and aggregated items. It is confusing.

Also the existing syntax does not support certain trigger expressions limiting our abilities to create complex problem conditions.

It would be great to introduce more powerful universal syntax that will be suitable for everything: triggers, calculated and aggregated items.

Zabbix acceptance

Zabbix will support new syntax for trigger expressions, calculated and aggregated items.

1. General design principles for the new syntax
    a. As simple as possible for new users of Zabbix
    b. Must support expressions as function parameters like func3(10*func1(), func2(), 123)
    c. Easy upgrade from the older syntax reusing existing database schema with functions but without loss of existing functionality
    d. Existing trigger functions must be reviewed and simplified if possible
2. No filters and complex aggregate functions will be allowed in trigger expressions
3. BNF grammar must be created and documented
4. Documentation changes
    a. The changes must be well documented and release notes updated
    b. Much more examples should be given for trigger expressions and calculated items showing new capabilities

New syntax

Time periods

Two existing parameters sec|#num and time_shift will be merged into one, which will be defined as:

- N values
    - min(/host/key, 5), minimum of the last 5 values
- Time period
    - min(/host/key,5m), minimum for a period of 5 minutes (time suffix is given)
- Time period or N values with Time shift
    - min(/host/key, 5:now-1d), minimum of the last 5 values one day ago (time shift is given)
    - min(/host/key, 1h:now-1d), minimum for a period of 1h one day ago
    - min(/host/key, 1h:now/1h), minimum for the last hour

Note that value shift will not be supported:

- min(/host/key, 10, **20**): not supported

Item reference and filters

An item can be referenced using the following syntax:

- /host/key: single item
- //key: item of the current host (useful for definition of calculated items)

A set of items can be referenced this way:

- /*/key: set of items matching any host
- /host/key[abc,*]: items matching key wildcard key[abc,*]
- /*/key?[group="ABC" and tag="tagname:value"]: items having tag "tagname:value" matching any host of group "ABC"
- /*/key[a,*,c]?[(group="ABC" and tag="Tag1") or (group="DEF" and (tag="Tag2:" or tag="Tag3:value")]: a combination of various filters

It will be possible to get access to other item attributes:

- /host/key</attribute>: access item attribute
- /host/key/clock: retrieve item timestamp
- /host/clock/trends/min | max | avg | count | clock: retrieve trend data

Current syntax v.s. new syntax

	Current syntax	New syntax	
Trigger expressions	{host:key.func(params)}=0	func(/host/key/period, params)	
	{host:key.min(#3)}>0 and {host:key.max(1h)}>100	min(/host/key,3)>0 and max(/host/key/,1h)>100	Number N without time suffix is a reference to N values Time suffix is mandatory for time periods
	{host:key.last()}>0	last(/host/key)>0	If period is missing, last value is returned
	{host:key.prev()}>0	last(/host/key,1) or prev(/host/key)	
	{host:vfs.fs.size[/,pfree].last()}<10	last(/host/vfs.fs.size[/,pfree])<10	
	{host:vfs.fs.size["/var/log",pfree].last()}<10	last(/host/vfs.fs.size["/var/log",pfree])<10	
	{host:key.min(1h, 24h)}<10	min(/host/key, 1h:now-24h)<10	Time shift
	{host:key.trendavg(1M, now/M)} > 1.2*{host:key.trendavg(1M, now/M-1y)}	trendavg(/host/key/,1M:now/M) > 1.2*trendavg(/host/key/,1M:now/M-1y)	
	N/A: Aggregate function with parameters as expressions	min(min(/host/key,1h), min(/host/key,1h), 25)	
	N/A: Aggregate function with many parameters of various types, another example	min(min(/host/key,1h), avg(host/key,1h), 25)	
	N/A: Using expressions as parameters	min(min(/host/key,1h), avg(host/key,1h)*100, 25)	
	N/A: Calculate func() of multiple data sources or expressions	min(min(/host1/key1, 1h),min(/host2/key2,1h))	
	N/A: Absolute time periods	min(/host/key, 1d:now/d+1d) min(/host/key, 1d:now/d) or trendmin(/host/key/,1d:now/d) min(/host/key, 2d:now/d+1d) min(/host/key, 1w:now/w) or tremdmin(/host/key/1w:now/w)	today yesterday last 2 days previous week
Calculated items	func(key, 30m) or func(host:key, 30m)	func(//key, 30m) func(/host/key, 30m)	
	func1(host1:key1, 30m, param) + func1(host2:key2, 30m, param)	func1(/host1/key1, 30m, param) + func1(/host2/key2, 30m, param)	

	100*last("vfs.fs.size[/,free]/last("vfs.fs.size[/,total]")	100*last(//vfs.fs.size[/,free]) / last(//vfs.fs.size[/,total])	//key/ is a reference to /current host/key/
	avg("Zabbix Server:zabbix[wcache,values]",600)	avg(/Zabbix Server/zabbix[wcache,values], 10m)	
	last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]")	last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes])	
	N/A: Calculate aggregate function using filter	sum(/host/vfs.fs.size[*,free], 10m)	
	N/A: Complex filters for data sources	sum(/hostA/vfs.fs.size[*,pfree]?[tag="Service:" or tag="Importance:High") and (group="Production" or group="Preproduction"], 5m)	Filtering by host groups and items tags will be supported
	N/A: Absolute time periods for trend functions	trendavg(/host/key/clock,1M:now/M) > 1.2*trendavg(/host/key,1M:now/M-1y)	
Aggregated items	grpmin["Servers",qps,avg,5m]	min(avg_foreach(/*/qps?[group="Servers"], 5m))	In func_foreach() foreach prefix means that the function will be executed for each array element. Array will be returned as a result.
	grpsum["MySQL Servers","vfs.fs.size[/,total]",last]	sum(last_foreach(/*/qps?[group="MySQL Servers"]))	
	grpavg[["Servers A","Servers B"],system.cpu.load,last]	avg(last_foreach(/*/qps?[group="Servers A" or group="Servers B"]))	

Conversion of existing trigger function

Existing functions will be transformed to a new syntax according to this table.

Function	New	Parameters	Comments
abschange **change**	**last(1)-last(2)** **abs(last(1)-last(2))**	-	The amount of absolute difference between last and previous values. The amount of difference between last and previous values.
avg, min, max, sum (sec|#num,<time_shift>)	**avg, min, max , sum**	item, period OR expr1, ...,exprN	avg(/host/key, 5m) OR min(min(/host/key),2*10,100*sum(3,4,5))
band (<sec|#num>,mask,<time_shift>)	TBD		Value of "bitwise AND" of an item value and mask.
count (sec|#num,<pattern>,<operator>,<time_shift>)	**count**	item, period, <operator>, <pattern>	Number of matching values
date, dayofmonth, dayofweek, now, time	**date** **dayofmonth** **dayofweek** **now** **time**	-	Current date in YYYYMMDD format Day of month in range of 1 to 31 Day of week in range of 1 to 7 (Mon - 1, Sun - 7) Number of seconds since the Epoch Current time in HHMMSS format

delta (sec|#num,<time_shift>)	**max**-**min**		Difference between the maximum and minimum values within the defined evaluation period ('max()' minus 'min()').
diff	**prev**<><**last**		Check if last and previous values differ
forecast (sec|#num,<time_shift>,time,<fit>, <mode>)	**forecast**	period,time, <fit>, <mode>	
fuzzytime (sec)	TBD		Check how much an item value (as timestamp) differs from the Zabbix server time.
iregexp (<pattern>,<sec|#num>)	**find**	item, period, <operator>, <pattern>	Check if there is at least one value that matches regular expression.
last (<sec|#num>,<time_shift>)	**last**	period	Return last value
logeventid (<pattern>)	TBD		
logseverity	TBD		
logsource (<pattern>)	TBD		
nodata (sec)	**nodata**	period	Return 1 if no elements, 0 - if there are elements
percentile (sec|#num,<time_shift>, percentage)	**percentile**	period, percentage	
prev	**last**(2)	-	
regexp (<pattern>,<sec|#num>)	**find**	item, period, <operator>, <pattern>	Check if there is at least one value that matches regular expression.
str (<pattern>,<sec|#num>)	**find**	item, period, <operator>, <pattern>	Check if there is at least one value that contains this string.
strlen (<sec|#num>,<time_shift>)	**length**(last)	expr	Length of values in characters (not bytes).
timeleft (sec|#num,<time_shift>,threshold, <fit>)	**timeleft**	period,time,<fit>, threshold, <fit>	
trendavg, trendcount, trenddelta, trendmax, trendmin, trendsum (period, period_shift)	**trendavg | trendcount | ...**	item, period	

New trigger functions

The following new trigger functions wl section contains list of alternative syntaxes that have been declined for various resons.

New function	Parameters	Comments
abs	expr	Return absolute value of the expression. For example: abs(last(/host/key,1)-last(/host/key,2))
find	item, period, <operator>, <pattern>	Check if there is at least one value that matches condition

Declined alternative syntaxes

This section contains list of alternative syntaxes that have been declined for various resons.

	Current syntax	Alternative syntax #1	Alternative syntax #2	Alternative syntax #3	
Trigger expressions	{host:key.func(params)}=0	func(/host/key/period, params)	(/host/key/period).func(params)	{host:key.func(params)}=0	/host/key/period is a datasource returning array of values
	{host:key.min(#3)}>0 and {host:key.max(1h)}>100	min(/host/key/3)>0 and max(/host/key/1h)>100	(/host/key/3).min>0 and (/host/key/1h).max>100	{host:key.min(#3)}>0 and {host:key.max(1h)}>100	Number N without time suffix is a reference to N values, .i.e/ host
	{host:key.last()}>0	last(/host/key)>0	(/host/key).last>0	{host:key.last()}>0	If period is missing, last value is returned
	{host:key.prev()}>0	last(/host/key/"1,1")	(/host/key/"1,1").last	{host:key.prev()}>0	In other words, /host/key/period,shift
	{host:vfs.fs.size[/,pfree].last()}<10	last(/host/vfs.fs.size[/,pfree])<10	(/host/vfs.fs.size[/,pfree]).last<10	{host:vfs.fs.size[/,pfree].last()}<10	Quoting
	{host:vfs.fs.size["/var/log",pfree].last()}<10	last(/host/vfs.fs.size["/var/log",pfree])<10	(/host/vfs.fs.size["/var/log",pfree]).last<10	{host:vfs.fs.size["/var/log",pfree].last()}<10	Quoting
	{host:key.min(1h, 24h)}<10	min(/host/key/1h, 24h)<10	(/host/key/1h, 24h).min<10	{host:key.min(1h, 24h)}<10	Time shift
	{host:key.trendavg(1M, now/M)} > 1.2*{host:key.trendavg(1M, now/M-1y)}	avg(@trends.avg/host/key/"1M,now/M") > 1.2*avg(@trends.avg/host/key/"1M, now/M-1y")	(@trends.avg/host/key/"1M, now/M").avg > 1.2*(@trends.avg/host/key/"1M, now/M-1y").avg	{host:key.trendavg(1M, now/M)} > 1.2*{host:key.trendavg(1M, now/M-1y)}	@trends.avg is a reference to trends.value_avg
	N/A: Aggregate function with many parameters of various types	min(/host/key/1h, /host/key/1h, 25)	(/host/key/1h, /host/key/1h, 25).min	min({host:key.min(1h)}, {host:key.min(1h)}, 25)	
	N/A: Aggregate function with many parameters of various types, another example	min(/host/key/1h, avg(host/key/1h), 25)	(/host/key/1h, (host/key/1h).avg, 25).min	min({host:key.min(1h)}, {host:key.avg(1h)}, 25)	
	N/A: Using expressions as parameters	min(/host/key/1h, avg(host/key/1h)*100, 25)	(/host/key/1h, (host/key/1h).avg*100, 25).min	min({host:key.min(1h)}, {host:key.avg(1h)}*100, 25)	
	N/A: Calculate func() of multiple data sources or expressions	min(/host1/key1,/host2/key2)	(/host1/key1,/host2/key2).min	min({host1:key1.min(1h)}, {host2:key2.min(1h)})	
	N/A: Absolute time periods	min(/host/key/"1d,now/d+1d") min(/host/key/"1d,now/d") or min(@trends.min/host/key/"1d,now/d") min(/host/key/"2d,now/d+1d") min(/host/key/"1w,now/w") or min(@trends.min/host/key/"1w,now/w")	(/host/key/"1d,now/d+1d").min (/host/key/"1d,now/d").min or (@trends.min/host/key/"1d,now/d").min (/host/key/"2d,now/d+1d").min (/host/key/"1w,now/w").min or (@trends.min/host/key/"1w,now/w").min	{host:key.min(1d,now/d+1d)}>0 {host:key.min(2d,now/d+1d)}>0	today yesterday last 2 days previous week
Calculated items	func(key, 30m) or func(host:key, 30m)	func(//key/30m) func(/host/key/30m)	(//key/30m).func (/host/key/30m).func	{key.func(30m)}>0 {host:key.func(30m)}>0	

	func1(host1:key1, 30m, param) + func1(host2:key2, 30m, param)	func1(/host1/key1/30m, param) + func1(/host2/key2/30m, param)	(/host1/key1/30m).func1(param) + (/host2/key2/30m).func1(param)		
	100*last("vfs.fs.size[/,free]/last("vfs.fs.size[/,total]")	100*last(//vfs.fs.size[/,free]) / last(//vfs.fs.size[/,total])	100*(//vfs.fs.size[/,free]).last / (//vfs.fs.size[/,total]).last		//key/ is a reference to /current host/key/
	avg("Zabbix Server:zabbix[wcache,values]",600)	avg(/Zabbix Server/zabbix[wcache,values]/10m)	(/Zabbix Server/zabbix[wcache,values]/10m).avg		
	last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]")	last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes])	(//net.if.in[eth0,bytes]).last+(//net.if.out[eth0,bytes]).last		
	N/A: Calculate aggregate function using filter	sum(/host/vfs.fs.size[*,free]/10m)	(/host/vfs.fs.size[*,free]/10m).sum	{host:vfs.fs.size[*,free].sum(10m)}	
	N/A: Complex filters for data sources	sum(/hostA/vfs.fs.size[*,pfree]?(tag="Service:" or tag="Importance:High") and (group="Production" or group="Preproduction")/5m)	(/hostA/vfs.fs.size[*,pfree]?(tag="Service:" or tag="Importance:High") and (group="Production" or group="Preproduction")/5m).sum	{host:vfs.fs.size[*,pfree]?(tag="Service:" or tag="Importance:High") and (group="Production" or group="Preproduction").sum(5m)}	Filtering by host groups and items tags will be supported
	N/A: Absolute time periods for trend functions	avg(@trends.avg/host/key/"1M,now/M") > 1.2*avg(@trends.avg/host/key/"1M,now/M-1y")	(@trends.avg/host/key/"1M,now/M").avg > 1.2*(@trends.avg/host/key/"1M,now/M-1y").avg		
Aggregated items	grpmin["Servers",qps,avg,5m]	min(avg_foreach(/*/qps?group="Servers"/5m))	(/*/qps?group="Servers"/5m).avg_foreach.min	min({*:qps[*,aaa]?[group="Servers" or group="Workstations"].avg_foreach(5m)})	In func_foreach() foreach prefix means that the function will be executed for each array element. Array will be returned as a result.
	grpsum["MySQL Servers","vfs.fs.size[/,total]",last]	sum(avg_foreach(/*/qps?group="MySQL Servers"))	(/*/qps?group="MySQL Servers").avg_foreach.sum	min({*:qps?group="MySQL Servers"@last()})	

	grpavg [["Servers A"," Servers B"],system.cpu.load, last]	avg(last_foreach(/*/qps?group="Servers A" or group="Servers B"))	(/*/qps?group="Servers A" or group="Servers B")).last_foreach.avg		
Why declined?		**min(/host/key/1h, /host/key/1h, 25) cannot be mapped to functionids**	**(/host/key/1h, /host/key/1h, 25).min cannot be mapped to functionids**	**min({host:key.min(1h)}, {host:key.min(1h)}, 25) cannot be mapped to functionids**	

Nonfunctional requirements

1. The new syntax must not worsen existing level of performance

Use cases

1. I want to calculate minimum of two different items in trigger expression
2. I want simpler calculated checks, I do not want to be confused by existence and different syntax of aggregate and calculated checks
3. I want same syntax for triggers, calculated and aggregated items. Why confuse users with different syntax?

Decisions made

1. No dotted notation like last().avg()
2. No filters and complex aggregates will be allowed in trigger expressions
3. Period and timeshift will be merged into single parameter period<:timeshift> and "now" is mandatory
    a. 2h
    b. 1d:now-1h
    c. 1M:now/M-30d/d-1h
4. No syntax #N will be supported anymore, time suffixes are mandatory now
5. No support of advanced syntax like last(/host/key/clock), last(/host/key/log/eventid), last(/host/key/logeventid), avg(/host/key/trends/avg,1M,now/M). Can be implemented later if needed.
6. No support of expressions in function parameters like in count(/host/key, 1h, eq, last(/host/key2))
    a. It cannot be converted to functionids

Open questions

1. count(/host/key, 1d, eq, 123) v.s. count(/host/key/?[value=123], 1d) vs count(/host/key, 1d, "value=123") vs /host/key, 1d, "value=123", "value*value")
        i. count(/host/key/?[value=123], 1d) is not good because it maps hardly to functionids
        ii. count(/host/key/log/?[eventid=123])
2. Shall we:
    a. replace prev() with last(2)
3. Shall we support map() type of operations to modify existing dataset, kind of preprocessing:
    a. I want to calculate sum of value*value for a period of time like sum(/host/key?[value=value*value], 1h) or (filtering and preprocessing) sum(/host/key, 1h, "value>0", "1/value""]
4. Filtering by regexp for item keys
5. Ability to skip key like in /host/?[tag="Service:ABC"] or /host/*?[tag="Service:ABC"]

Changes log

- **0.11**
    - Minor changes
- **0.12**
    - Simplified quoting for item keys

- **0.13**
  - Syntax of filter changed: /host/key/[filter]/period to /host/key?filter/period
  - Simple functions will not require []: min(1,2,3) v.s. count([1,2,3],eq,1)
  - New syntax for trends functions: trendavg(/host/key/1M/[now/M]) => avg(@trends.avg/host/key/1M/[now/M])
  - New syntax for aggregate functions: sum(grpavg(/*/qps?group="MySQL Servers")) => sum(avg_foreach(/*/qps?group="MySQL Servers"))
  - New syntax to reference current host in calculated items: ./key/period => //key/period
- **0.14**
  - Massive changes in the new syntax and new document structure
- **0.15**
  - More details for conversion of the existing trigger functions
- **0.16**
  - New trigger functions abs() and find() described
  - Some other minor changes