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. Trigger expressions
    a. No filters and complex aggregate functions will be allowed in trigger expressions
3. Calculated checks
    a. Calculated items must use new syntax
4. Aggregated checks
    a. Syntax of the existing aggregate functions must be changed to use new syntax
    b. Formula for aggregation must be moved from item key to Formula field similar to calculated items
        i. Therefore any valid string can be used as item key for aggregate item
        ii. Note that the item key must contain at least one LLD macro in order to make item key different from item prototype key
    c. Existing grpavg, grpmin, grpmax and grpsum functions must be changed to new functions
    d. Zabbix must not support aggregated items anymore
        i. Existing aggregated items must be converted into calculated items
5. Scope of use for LLD and user macros must remain the same or extended
6. BNF grammar must be created and documented
7. 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

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,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

Function	New	Parameters	Comments
abschange **change**	**abs(last(1)-last(2))** **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>)	**band**	item, period, mask	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	**last(1)<>last(2)**		Check if last and previous values differ
forecast (sec|#num,<time_shift>,time, <fit>,<mode>)	**forecast**	item, period,time, <fit>,<mode>	
fuzzytime (sec)	**fuzzytime**	item, period	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**	item, period	Return last value
logeventid (<pattern>)	**logeventid**	item, <pattern>	
logseverity	**logseverity**	item	

Function	New	Parameters	Comments
logsource (<pattern>)	**logsource**	item, <pattern>	
nodata (sec)	**nodata**	item, period	Return 1 if no elements, 0 - if there are elements
percentile (sec|#num, <time_shift>,percentage)	**percentile**	item, period, percentage	
prev	**last**(2)	-	prev() will no longer be supported
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**	item,period,time, <fit>,threshold, <fit>	
trendavg, trendcount, trenddelta, trendmax, trendmin, trendsum (period, period_shift)	**trendavg | trendcount | ~~trenddelta~~ |...**	item, period	**trenddelta** will not be supported anymore

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

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
7. Functions prev(), change(), abschange() and diff() will not be supported
8. No filtering by regexp for item keys for now
9. No filter by value using syntax like /host/key/?[value=123], use count() or find() instead
10. It will not be possible to get access to other item attributes, at least in scope of this development:
    - /host/key</attribute>: access item attribute
    - /host/key/clock: retrieve item timestamp
    - /host/clock/trends/min | max | avg | count | clock: retrieve trend data
11. No protection from heavy calculated checks that may significantly affect Zabbix performance will be implemented (say, aggregation for millions of items)