2 Выражение триггера

Обзор

Используемые в триггерах выражения являются очень гибкими. Вы можете использовать их для создания сложных логических тестов, учитывая статистику по мониторингу.

Простое полезное выражение может выглядеть примерно так:

```
{<сервер>:<ключ>.<функция>(<параметр>)}<оператор><константа>
```

Хотя синтаксис абсолютно одинаков, с функциональной точки зрения имеется два типа выражений триггеров:

- выражение проблемы - определяет условия проблемы
- выражение восстановления (опционально) - определяет дополнительные условия решения проблемы

Когда задано только выражение проблемы, это выражение будет использоваться как для порога определения проблемы, так и для определения порога восстановления проблемы. Как только выражение проблемы будет вычислено значением ПРАВДА, произойдет проблема. Как только выражение проблемы будет вычислено значением ЛОЖЬ, проблема решится.

Когда определены выражение проблемы и дополнительное выражение восстановления, решение проблемы становится более сложным: не только выражение проблемы должно стать ЛОЖЬЮ, но также выражение восстановление должно стать ПРАВДОЙ. Такой подход полезен для избежания постоянного переключения состояния триггера в гистерезисе.

Функции

Функции триггеров позволяют ссылаться на собранные значения, текущее время и другие факторы.

Имеется полный список поддерживаемых функций.

Параметры функций

Большинство числовых функций принимают количество секунд в качестве параметра.

Вы можете использовать префикс **#**, чтобы указать что этот параметр должен иметь другой смысл:

ВЫЗОВ ФУНКЦИИ	СМЫСЛ
sum(600)	Сумма всех значений, которые не старше чем 600 секунд
sum(#5)	Сумма всех значений, которых не более чем 5 самых последних значений

Функция **last** использует другой смысл для значений, когда начинается с решетки - она дает выбрать n-ое предыдущее значение, так что с учетом значений 3, 7, 2, 6, 5 (от наиболее нового

до наиболее старого), при **last(#2)** вернется *7* и при **last(#5)** вернется *5*.

Несколько функций поддерживают дополнительный, второй параметр сдвиг_времени. Этот параметр позволят ссылаться на данные из периода времени в прошлом. Например, для **avg(1h,1d)** будет возвращено среднее значение за час днем ранее.

Вы можете использовать поддерживаемые суффиксы преобразований в выражениях триггеров, например, '5m' (минут) вместо '300' секунд или '1d' (день) вместо '86400' секунд. '1K' будет состоять из '1024' байт.

Числа со знаком '+' не поддерживаются.

Операторы

Следующие операторы поддерживаются для триггеров **(представлены по убыванию приоритета выполнения)**:

ПРИОРИТЕТ	ОПЕРАТОР	ОПРЕДЕЛЕНИЕ	Заметки по неизвестным значениям
1	**-**	Унарный минус	**-**Неизвестно → Неизвестно
2	**not**	Логическое НЕ	**not** Неизвестно → Неизвестно
3	*	Умножение	0 * Неизвестно → Неизвестно (да, Неизвестно, не 0 - чтобы не потерять Неизвестно в арифметических операциях) 1.2 * Неизвестно → Неизвестно
	/	Деление	Неизвестно / 0 → ошибка Неизвестно / 1.2 → Неизвестно 0.0 / Неизвестно → Неизвестно
4	**+**	Арифметический плюс	1.2 **+** Неизвестно → Неизвестно
	-	Арифметический минус	1.2 **-** Неизвестно → Неизвестно
5	**<**	Менее чем. Этот оператор может быть представлен в виде: A<B ⇔ (A<B-0.000001)	1.2 **<** Неизвестно → Неизвестно
	<=	Менее чем или равно.Этот оператор может быть представлен в виде: A<=B ⇔ (A≤B+0.000001)	Неизвестно **<=** Неизвестно → Неизвестно
	>	Более чем. Этот оператор может быть представлен в виде: A>B ⇔ (A>B+0.000001)	
	>=	Более чем или равно. Этот оператор может быть представлен в виде: A>=B ⇔ (A≥B-0.000001)	

ПРИОРИТЕТ	ОПЕРАТОР	ОПРЕДЕЛЕНИЕ	Заметки по неизвестным значениям
6	**=**	Равенство. Этот оператор может быть представлен в виде: A=B ⇔ (A≥B-0.000001) и (A≤B+0.000001)	
	<>	Не равно. Этот оператор может быть представлен в виде: A<>B ⇔ (A<B-0.000001) или (A>B+0.000001)	
7	**and**	Логическое И	0 **and** Неизвестно → 0 1 **and** Неизвестно → Неизвестно Неизвестно **and** Неизвестно → Неизвестно
8	**or**	Логическое ИЛИ	1 **or** Неизвестно → 1 0 **or** Неизвестно → Неизвестно Неизвестно **or** Неизвестно → Неизвестно

Операторы **not**, **and** and **or** регистрозависимы и должны быть в нижнем регистре. Они также должны быть окружены символами пробелов или круглыми скобками.

Все операторы, кроме унарных **-** и **not**, имеют ассоциативность слева на право. Унарные **-** и **not** не ассоциативны (имеется в виду необходимо использовать **-(-1)** и **not (not 1)** вместо **--1** и **not not 1**).

Результат вычисления:

- Операторы **<**, **<=**, **>**, **>=**, **=**, **<>** должны давать '1' в выражении триггера, если указанное соотношение правдиво и '0', если оно ложно. Если по крайней мере один операнд Неизвестен, то и результат будет Неизвестно;
- **and** по известным операндам должно давать '1', если оба из этих операндов сравнения не равны '0'; в противном случае, будет давать '0'; для неизвестных операндов **and** даст '0' только, если один из операндов сравнения равен '0'; в противном случае, он даст 'Неизвестно';
- **or** по известным операндам должно давать '1', если какой-либо из этих операндов сравнения не равен '0'; в противном случае, будет давать '0'; для неизвестных операндов **or** даст '1' только, если один из операндов сравнения не равен '0'; в противном случае, он даст 'Неизвестно';
- Результат логического операнда отрицания **not** для известного операнда равен '0', если значение этого операнда сравнения не равно '0'; '1', если значение его операнда сравнения равно '0'. Для неизвестных операндов **not** даст 'Неизвестно'.

Кэширование значений

Значения, которые требуются для вычисления триггеров, кэшируются Zabbix сервером. По этой причине такое вычисление триггеров на некоторое время приводит к более высокой загрузке базы данных после перезапуска сервера. Кэш значений не очищается, когда значения истории элементов данных удаляются (либо вручную, либо при помощи автоматической

очистки истории), поэтому сервер будет использовать кэшированные значения пока они не станут старше, чем периоды времени, которые заданы в функциях триггеров, либо пока сервер не будет перезапущен.

Примеры триггеров

Пример 1

Высокая загрузка процессора на www.zabbix.com.

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5
```

'www.zabbix.com:system.cpu.load[all,avg1]' представляет короткое имя наблюдаемого параметра. Эта строка указывает, что сервер - 'www.zabbix.com' и наблюдаемый ключ - 'system.cpu.load[all,avg1]'. Используя функцию 'last()', мы ссылаемся на самое последнее значение. И наконец '>5' означает, что триггер перейдет в состояние ПРОБЛЕМА всякий раз, когда самое новое измерение загрузки процессора на сервере www.zabbix.com будет превышать 5.

Пример 2

www.zabbix.com перегружен

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or
{www.zabbix.com:system.cpu.load[all,avg1].min(10m)}>2
```

Это выражение будет истинным, когда либо текущая загрузка процессора станет более 5, либо загрузка процессора больше значения 2 за последние 10 минут.

Пример 3

/etc/passwd был изменен

Используется функция diff:

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff()}=1
```

Это выражение будет истинным, когда предыдущее значение контрольной суммы файла /etc/passwd отличается от самого нового значения.

Аналогичные выражения могут быть полезны для мониторинга изменений в важных файлах, таких как /etc/passwd, /etc/inetd.conf, /kernel и других.

Пример 4

Кто-то скачивает большой файл из Интернет

Используется функция min:

```
{www.zabbix.com:net.if.in[eth0,bytes].min(5m)}>100K
```

Это выражение будет истинным, когда количество полученных байт на eth0 превышает 100 КБ за последних 5 минут.

Пример 5

Оба узла кластера SMTP серверов недоступны

Примечание, в выражении используются два разных узла сети:

```
{smtp1.zabbix.com:net.tcp.service[smtp].last()}=0 and
{smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
```

Это выражение будет истинным, когда оба SMTP сервера недоступны на обоих smtp1.zabbix.com и smtp2.zabbix.com.

Пример 6

Zabbix агент нуждается в обновлении

Используется функция str():

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
```

Это выражение будет истинным, когда версия Zabbix агента содержит в себе 'beta8' (возможно 1.0beta8).

Пример 7

Сервер недоступен

```
{zabbix.zabbix.com:icmpping.count(30m,0)}>5
```

Это выражение будет истинным, если узел сети “zabbix.zabbix.com“ недоступен более 5 раз за последние 30 минут.

Пример 8

Нет данных за последние 3 минуты

Используется функцию nodata():

```
{zabbix.zabbix.com:tick.nodata(3m)}=1
```

Для того, чтобы этот триггер заработал, элемент данных 'tick' должен быть задан как элемент данных типа Zabbix траппер. Узел сети должен периодически отправлять данные этому элементу данных, используя zabbix_sender. Если не было получено данных за последние 180 секунд, значением триггера станет ПРОБЛЕМА.

Обратие внимание, что 'nodata' можно использовать с любым типом элементов данных.

Пример 9

Активность CPU в ночное время

Используется функция time():

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2 and
{zabbix:system.cpu.load[all,avg1].time()}>000000 and
{zabbix:system.cpu.load[all,avg1].time()}<060000
```

Триггер может изменить свое состояние в истинное только в ночное время (00:00-06:00).

Пример 10

Проверка синхронизации времени на клиенте со временем на Zabbix сервере

Используется функция fuzzytime():

```
{MySQL_DB:system.localtime.fuzzytime(10)}=0
```

Триггер изменит состояние на проблему тогда, когда локальное время на сервере MySQL_DB и Zabbix сервере различаются более чем на 10 секунд. Обратите внимание, что 'system.localtime' необходимо настроить пассивной проверкой.

Пример 11

Сравнение средней загрузки сегодня со средним значением загрузки за это же время вчера (использование второго параметра сдвиг_времени).

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,1d)}>2
```

Триггер изменит свое состояние на проблему, если средняя загрузка за последний час будет в два раза больше чем за аналогичный период времени вчера.

Пример 12

Использование значение другого элемента данных в качестве порогового значения триггера:

```
{Template PfSense:hrStorageFree[{#SNMPVALUE}].last()}<{Template
PfSense:hrStorageSize[{#SNMPVALUE}].last()}*0.1
```

Триггер изменит свое состояние на проблему, если свободное пространство на диски упадет ниже 10 процентов.

Пример 13

Использование результата вычисления для получения количества триггеров больше порога:

```
({server1:system.cpu.load[all,avg1].last()}>5) +
({server2:system.cpu.load[all,avg1].last()}>5) +
({server3:system.cpu.load[all,avg1].last()}>5)>=2
```

Триггер изменит свое состояние на проблему, если по крайней мере два триггера из выражения будут больше 5.

Гистерезис

Порой требуется интервал между состояниями проблемы и восстановлением, отличный от простого порога. Например, если мы хотим задать триггер, который докладывает о проблеме, когда температуры в серверной комнате становится больше 20°C, и мы хотим, чтобы он оставался в состоянии проблемы пока температура не опустится ниже 15°C, простого триггера с порогом в 20°C будет недостаточно.

На самом деле, нам необходимо задать выражение триггера сначала для события проблемы (температура выше 20°C). Затем нам необходимо задать дополнительное условие (температура ниже 15°C). Это можно сделать определив дополнительный параметр *Выражение восстановления* при настройке триггера.

В этом случае восстановление проблемы будет происходить в два этапа:

- Первым этапом, выражение проблемы (температура выше 20°C) должна быть вычислена значением ЛОЖЬ
- Вторым этапом, выражение восстановления (температура ниже 15°C) должна быть вычислена значением ПРАВДА

Выражение восстановление будет вычисляться только, когда сначала событие о проблеме будет решено.

Наличие только выражения восстановления со значением ПРАВДА не решает проблему, если выражение проблемы всё ещё вычисляется значением ПРАВДА!

Пример 1

Температура в серверной комнате слишком высокая.

Выражение проблемы:

```
{server:temp.last()}>20
```

Выражение восстановления:

```
{server:temp.last()}<=15
```

Пример 2

Очень мало свободного места на диске

Выражение проблемы: если меньше 10ГБ за последние 5 минут

```
{server:vfs.fs.size[/,free].max(5m)}<10G
```

Выражение восстановления: если больше 40ГБ за последние 10 минут

```
{server:vfs.fs.size[/,free].min(10m)}>40G
```

Выражения с неподдерживаемыми элементами данных и неизвестными значениями

Версии до Zabbix 3.2 очень строго относились к неподдерживаемым элементам данных в выражениях триггеров. Любой неподдерживаемый элемент данных в выражении незамедлительно менял значение триггера на Неизвестно.

Начиная с Zabbix 3.2 существует более гибкий подход к неподдерживаемым элементам данных, допуская неизвестные значения при вычислении выражений:

- У некоторых функций их значения не зависят от того поддерживается ли элемент данных или нет. Такие функции теперь вычисляются даже, если ссылаются на неподдерживаемые элементы данных. Смотрите список в разделе функции и неподдерживаемые элементы данных.
- Логические выражения с ИЛИ и И могут быть вычислены для известных значений в двух случаях независимо от неизвестных операндов:
  - “1 `or` Неподдерживаемый_элемент_данных1.некая_функция() `or` Неподдерживаемый_элемент_данных2. некая_функция() `or` ...” может быть вычислена как '1' (Правда),
  - “0 `and` Неподдерживаемый_элемент_данных1. некая_функция() `and` Неподдерживаемый_элемент_данных2. некая_функция() `and` ...” может быть вычислена как '0' (Ложь).
    Zabbix пытается вычислить логические выражения принимая неподдерживаемые элементы данных как Неизвестные значения. В двух случаях, упомянутых выше,

будет приниматься известное значение; в остальных случаях значением триггера будет Неизвестно.

- Если вычисление триггера по поддерживаемому элементу данных приведет к ошибке, значением функции будет Неизвестно и оно будет частью дальнейшего вычисления выражения.

Обратите внимание на то, что неизвестные значения могут “исчезать” только в логических выражениях описанных выше. В арифметических выражениях неизвестные Неизвестному результату (за исключением деления на 0).

Если выражение триггера с несколькими неподдерживаемыми элементами данных вычисляется как Неизвестное, сообщение об ошибке в веб-интерфейсе ссылается на последний вычисленный неподдерживаемый элемент данных.