2 Пользовательские макросы

Обзор

В дополнение к макросам поддерживающимся из коробки в Zabbix также для большей гибкости поддерживаются пользовательские макросы.

Пользовательские макросы можно можно определить на глобальном, уровне шаблона и уровне узла сети. Такие макросы имеют специальный синтаксис:

```
{$МАКРОС}
```

Zabbix раскрывает макросы в соответствии со следующей очередностью:

1. макрос, назначенный узлу сети (проверяется в первую очередь)
2. макрос, назначенный на первый уровень шаблонов у узла сети (т.е. шаблоны присоединенные напрямую к узлу сети), которые отсортированы по ID шаблона
3. макрос, назначенный на второй уровень шаблонов у узла сети, которые отсортированы по ID шаблона
4. макрос назначенный на третий уровень шаблонов у узла сети, которые отсортированы по ID шаблона и так далее
5. глобальные макросы (проверяются в последнюю очередь)

Иными словами, если макрос не существует непосредственно у узла сети, тогда Zabbix попытается найти его в шаблонах, присоединенных к этому узлу сети, с учетом увеличивающейся глубины шаблонов. Если макрос все еще не найден, то будет использован макрос глобального уровня, если он существует конечно.

В случае, если Zabbix не удалось найти макрос, макрос не будет раскрыт.

Пользовательские макросы можно использовать в:

- именах элементов данных
- в параметрах ключей элементов данных
- в интервалах обновления и гибких интервалах элементов данных
- именах и описаниях триггеров
- в параметрах и константах выражений триггеров (смотрите примеры)
- многих других местах (смотрите Макросы поддерживаемые по назначению)

Общие случаи использования глобальных макросов и макросов уровня узлов сети

- используйте глобальные макросов в нескольких местах; затем измените значения макроса и изменения конфигурации применятся во всех местах за “одно нажатие”
- получение преимуществ шаблонов со специфичными для узлов сети атрибутами: пароли, номера портов, имена файлов, регулярные выражения, и т.д.

Настройка

Для добавления пользовательских макросов, перейдите в соответствующие разделы в веб-интерфейсе:

- для глобальных макросов, посетите *Администрирование → Общие → Макросы*
- для макросов на уровне узлов сети и шаблонов, откройте свойства узла сети или шаблона и найдите вкладку *Макросы*

Если пользовательский макрос используется в элементах данных или триггерах шаблона, то предлагается добавить этот макрос к шаблону, даже если он задан на глобальном уровне. Таким образом при экспорте шаблона в XML и импорте его в другую систему элементы данных и триггеры продолжат работать, как это и ожидалось.

В именах макросов допускаются следующие символы: **A-Z** , **0-9** , **_** , **.**

Примеры

Пример 1

Использование макроса в ключе элемента данных “Состояние демона SSH”:

```
net.tcp.service[ssh,{$SSH_PORT}]
```

Этот элемент данных можно присоединить к нескольким узлам сети, при условии, что значение **{$SSH_PORT}** указано на этих узлах сети.

Пример 2

Использование макроса уровня узла сети в триггере “Загрузка CPU слишком высокая”:

```
{ca_001:system.cpu.load[,avg1].last()}>{$MAX_CPULOAD}
```

Такой триггер можно создать у шаблона, он не будет изменен у отдельных узлов сети.

Если вы хотите использовать количество значений в качестве параметра функции (например, **max(#3)**), добавьте символ решетки # непосредственно в значение макроса, например вот так: SOME_PERIOD ⇒ #3

Пример 3

Использование двух макросов в триггере “Загрузка CPU слишком высокая”:

```
{ca_001:system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$MAX_CPULOAD}
```

Заметьте, что макрос можно использовать в качестве параметра в функциях триггеров, в этом примере в функции **min()**.

Пользовательские макросы будут раскрыты в триггерах, если макросы используются в параметрах или как константы. Они НЕ будут раскрыты, если используются как ссылка на

функции, имена узлов сети, ключи элементов данных или операторы.

Пример 4

Синхронизация условия недоступности агентов с интервалом обновления элемента данных:

- задайте макрос {$INTERVAL} и используйте его в интервале обновления элемента данных;
- используйте {$INTERVAL} параметром в триггере на недоступность агента:

```
{ca_001:agent.ping.nodata({$INTERVAL})}=1
```

Пример 5

Централизация конфигурации рабочего времени:

- создайте глобальный макрос {$WORKING_HOURS} равный `1-5,09:00-18:00`;
- используйте его в *Администрирование → Общие → Рабочее время*;
- используйте его в *Пользователь → Оповещения → Когда активно*;
- используйте его для настройки более частого опроса элементов данных в течении рабочего времени:

- используйте его в условии действия *Период времени*;
- отредактируйте рабочее время в *Администрирование → Общие → Макросы*, если требуется.

Контекст пользовательских макросов

В пользовательских макросах можно использовать опциональный контекст, позволяющий переопределять значение по умолчанию на значение с учётом конкретной ситуации.

Пользовательские макросы с контекстом имеют схожий синтаксис:

```
{$МАКРОС:контекст}
```

Контекстом макроса является простое текстовое значение. Общим случаем использования контекстов в макросах будет использование значения низкоуровневого макроса в качестве контекста пользовательских макросов. Например, можно определить прототип триггера для обнаружения примонтированных файловых систем с использованием разных порогов малого свободного места в зависимости от точек монтирования или типов файловых систем.

В контекстах макроса поддерживаются только макросы низкоуровневого обнаружения. Любые

другие макросы игнорируются и обрабатываются как текстовые значения.

Технически, контекст макроса задается с использованием правил, похожих на параметры ключей элементов данных, за исключением того, что контекст макроса при наличии , символа не обрабатывается как несколько параметров:

- Контекст макроса необходимо заключать в "" кавычки, если контекст содержит } символ или начинается с "" символа. Кавычки внутри заключенном в кавычки контексте необходимо экранировать при помощи \ символа. Сам символ \ не экранируется, что означает, что невозможно задать заключенный в кавычки контекст оканчивающийся на \ символ - макрос {$MACRO:""a:\b\c\""} ошибочный
- Пробелы в начале контекста игнорируются, пробелы в конце не игнорируются. Например, {$MACRO:A} тоже самое что и {$MACRO: A}, но не {$MACRO:A }.
- Все пробелы до кавычек в начале и после кавычек игнорируются, но все пробелы внутри кавычек не игнорируются. Макросы {$MACRO:""A""}, {$MACRO: ""A""}, {$MACRO:""A " } и {$MACRO: ""A" } одинаковы, но макрос {$MACRO:""A""} и {$MACRO:"" A ""} не одинаковы.

Следующие макросы одинаковы, так как имеют один и тот же контекст: {$MACRO:A}, {$MACRO: A} и {$MACRO:""A""}. Такое поведение отлично от ключей элементов данных, где key[a], key[a] и key[""a""] одинаковы семантически, но различны для критерия уникальности.

Когда макросы с контекстами обрабатываются, Zabbix ищет макрос со своим контекстом вычисляется. Если макрос с этим контекстом не задан на узле сети или присоединенных шаблонах, и не задан как глобальный макрос с контекстом, тогда выполняется поиск макроса без контекста.

Смотрите пример использования контекста макроса в прототипе триггера о свободном месте на диске и примите во внимание пункт об ограничениях.

From:
https://www.zabbix.com/documentation/4.2/ - **Zabbix Documentation 4.2**

Permanent link:
https://www.zabbix.com/documentation/4.2/ru/manual/config/macros/usermacros

Last update: **2018/10/01 09:42**