4 Обнаружение JMX объектов

Обзор

Имеется возможноть обнаружения всех атрибутов JMX MBeans или MBean, а также можно указать шаблон для обнаружения этих объектов.

Для настройки правила обнаружения обязательно понимать разницу между Mbean и атрибутами Mbean. MBean является объектом, который может представлять собой устройство, приложение или любой другой ресурс, которым необходимо управлять. Например, имеется Mbean, который представляет собой веб-сервер. Его атрибутами являются количество подключений, количество потоков, время ожидания запросов, кэш файлов http, использование памяти и так далее. Выражая эту мысль человеческим языком, мы можем определить кофе машину как Mbean, у которого имеются следующие атрибуты для мониторинга: количество воды на каждую кружку, средний расход воды за определенный период времени, количество кофейных зерен требующихся на каждую кружку, кофейные зерна и время наполнения водой и так далее.

Ключ элемента данных

В настройках правила обнаружения выберите **JMX агент** в поле *Тип*.

Используемым ключом элемента данных, является

```
jmx.discovery[<режим обнаружения>,<имя объекта>]
```

где

- режим обнаружения - один из следующих: *attributes* (получение JMX MBean атрибутов, по умолчанию) или *beans* (получение JMX MBeans)
- имя объекта - шаблон имени объекта, который определяет получаемфые имена MBean (по умолчанию пусто, получение всех зарегистрированных компонентов).

Вы можете обратиться к ObjectName документации для получения опций по указанию шаблона имени объекта.

Если параметры не переданы, с JMX запрашиваются все MBean атрибуты.

Отсутствие заданных параметров в случае с JMX обнаружением или попытка получения всех атрибутов широкого диапазона, таких как `*:type=*,name=*`, может привести к потенциальным проблемам производительности.

Этот ключ поддерживается начиная с Zabbix Java gateway 3.4.

Примеры ключей элементов данных:

```
jmx.discovery  #Получение всех JMX MBean атрибутов
jmx.discovery[beans]  #Получение всех JMX MBeans
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]  #Получение всех
```

атрибутов сборщика мусора
`jmx.discovery[beans,"*:type=GarbageCollector,name=*"]` #Получение всех сборщиков мусора

Этот элемент данных возвращает JSON объект. Например, в обнаружении MBean атрибутов (переформатировано для наглядности):

```
{
   "data":[
      {
         "{#JMXVALUE}":"0",
         "{#JMXTYPE}":"java.lang.Long",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS
Scavenge,CollectionCount",
         "{#JMXATTR}":"CollectionCount"
      },
      {
         "{#JMXVALUE}":"0",
         "{#JMXTYPE}":"java.lang.Long",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS
Scavenge,CollectionTime",
         "{#JMXATTR}":"CollectionTime"
      },
      {
         "{#JMXVALUE}":"true",
         "{#JMXTYPE}":"java.lang.Boolean",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS
Scavenge,Valid",
         "{#JMXATTR}":"Valid"
      },
      {
         "{#JMXVALUE}":"PS Scavenge",
         "{#JMXTYPE}":"java.lang.String",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS
Scavenge,Name",
         "{#JMXATTR}":"Name"
      },
      {
         "{#JMXVALUE}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXTYPE}":"javax.management.ObjectName",
         "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
         "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS
Scavenge,ObjectName",
         "{#JMXATTR}":"ObjectName"
```

```
        }
    ]
}
```

В обнаружении MBeans (переформатировано для наглядности):

```
{
    "data":[
        {
            "{#JMXDOMAIN}":"java.lang",
            "{#JMXTYPE}":"GarbageCollector",
            "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
            "{#JMXNAME}":"PS Scavenge"
        }
    ]
}
```

Поддерживаемые макросы

Следующие макросы поддерживаются для использования в фильтре правила обнаружения и прототипах элементов данных, триггеров и графиков:

Макрос	Описание
Обнаружение MBean атрибутов	
{#JMXVALUE}	Значение атрибута.
{#JMXTYPE}	Тип атрибута.
{#JMXOBJ}	Имя объекта.
{#JMXDESC}	Имя объекта, включая имя атрибута.
{#JMXATTR}	Имя атрибуты.
Обнаружение MBeans	
{#JMXDOMAIN}	Домен MBean. (*зарезервированное имя в Zabbix*)
{#JMXOBJ}	Имя объекта. (*зарезервированное имя в Zabbix*)
{#JMX<ключ свойства>}	MBean свойства (такие как {#JMXTYPE}, {#JMXNAME}). Необходимо сделать важные замечания на которые следует обратить внимания при определении имени MBean атрибута, которое создано из имени MBean свойства по следующему алгоритму: * регистр имени атрибута сменился на верхний регистр; * регистр имени атрибута игнорируется (макрос LLD не генерируется) в случае, если он содержит неподдерживаемые символы. Поддерживаемые символы можно описать при помощи следующего регулярного выражения: “A-Z0-9_\.”; * если имя атрибута называется “*obj*” или “*domain*”, оно будет заменено значениями свойств Zabbix {#JMXOBJ} и {#JMXDOMAIN} (поддерживается начиная с Zabbix 3.4.3.).

Пожалуйста, обратите внимание на этот пример с jmx.discovery (с “beans” режимом). У MBean имеются следующие свойства:

```
name=test
тип=Type
attributes []=1,2,3
Name=NameOfTheTest
domAin=some
```

В результате JMX обнаружения сгенерируются следующие LLD макросы:

- {#JMXDOMAIN} - Zabbix внутренний, описывающий домен MBean
- {#JMXOBJ} - Zabbix внутренний, описывающий объект MBean
- {#JMXNAME} - создается из свойства “name”

Игнорируемые свойства:

- тип : это имя содержит неподдерживаемые символы (не-ASCII)
- attributes[] : это имя содержит неподдерживаемые символы (квадратные скобки не поддерживаются)
- Name : уже задан (name=test)
- domAin : является зарезервированным именем в Zabbix

Давайте рассмотрим еще два практических примера создания LLD правила с использованием Mbean. Для понимания разницы между LLD правилом, которое собирает Mbeans, и LLD правилом, которое собирает Mbean атрибуты, лучше взгляните на следующую таблицу:

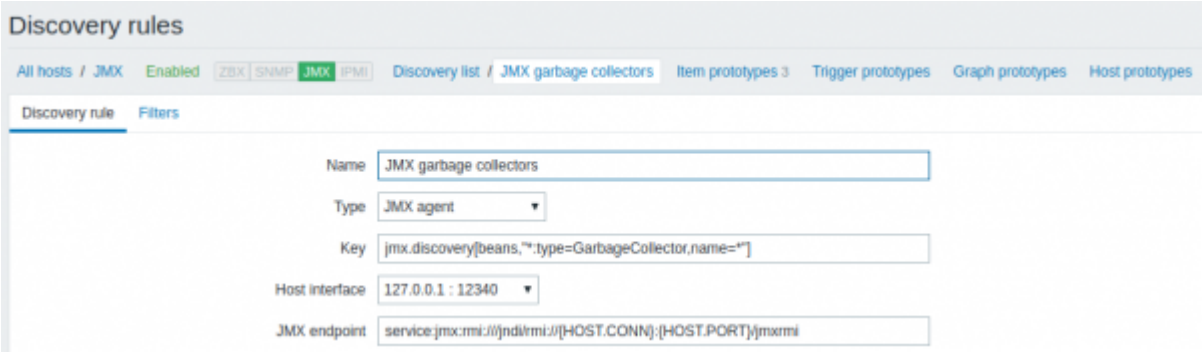
MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

LLD правило, собирающее Mbeans

Это правило вернет 3 объекта: колонки верхней строки: MBean1, MBean2, MBean3.

Для получения более подробной информации об объектах, пожалуйста, обратитесь к таблице поддерживаемых макросов, раздел *Обнаружение MBeans*.

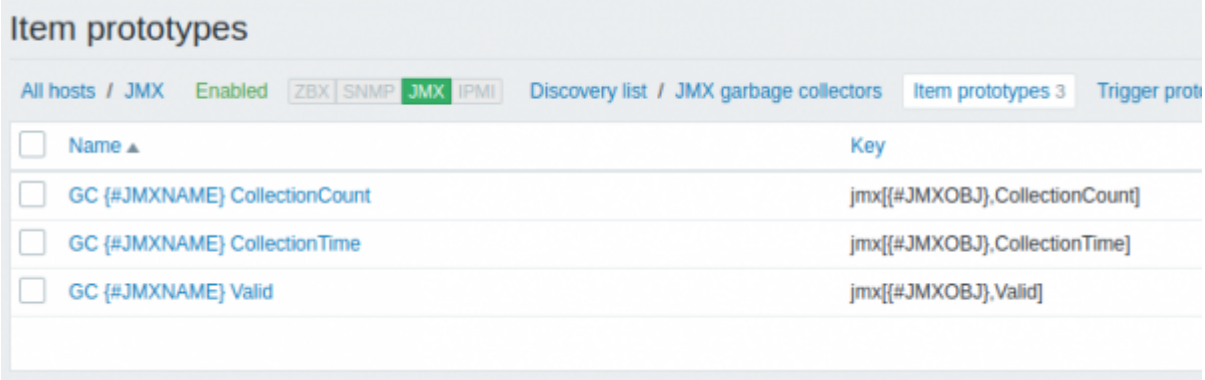
Настройки правила обнаружения, собирающего Mbeans (без атрибутов), выглядят следующим образом:



Используемый ключ:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

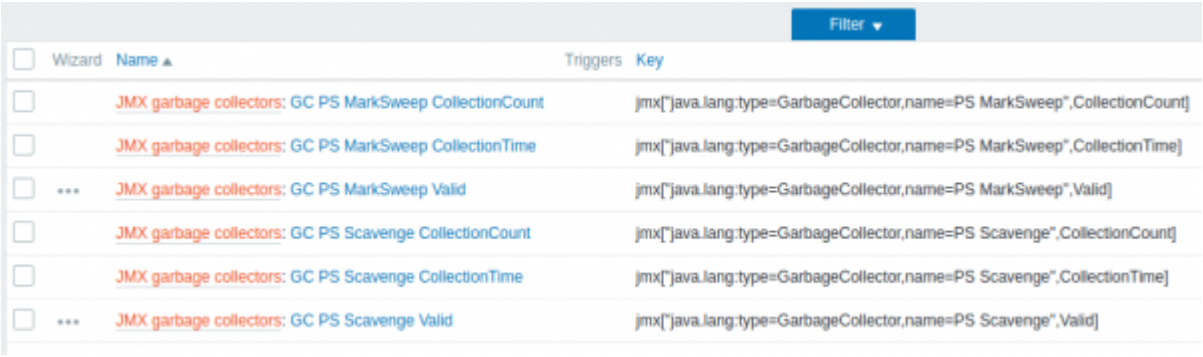
Обнаружатся все сборщики мусора без атрибутов. Так как сборщики мусора имеют одинаковый набор атрибутов, мы может использовать желаемые атрибуты в прототипах элементов данных следующим образом:



Используемые ключи:

```
jmx[{#JMXOBJ},CollectionCount]
jmx[{#JMXOBJ},CollectionTime]
jmx[{#JMXOBJ},Valid]
```

Результатом LLD правила обнаружения будет что-то близкое к этому (элементы данных обнаружились по двум сборщикам мусора):

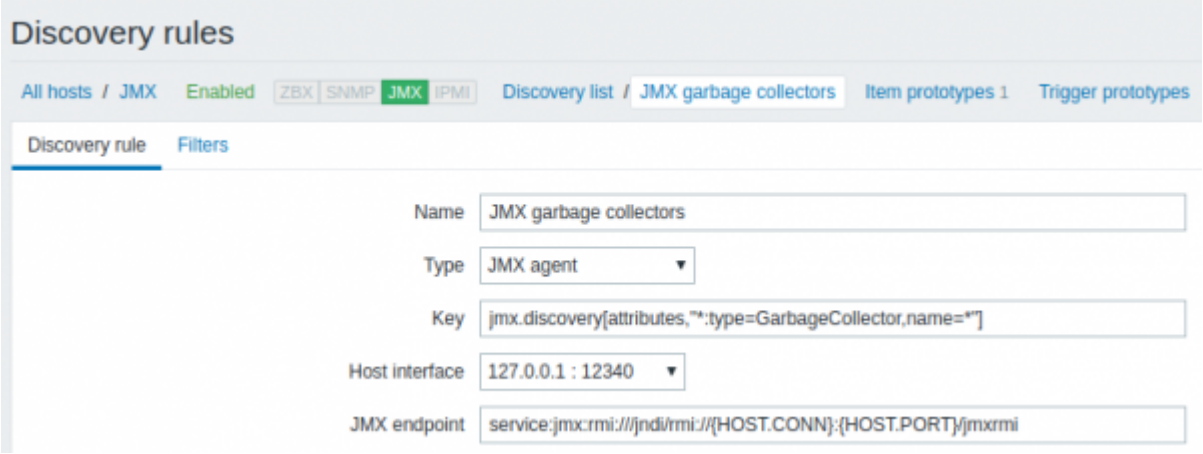


LLD правило, собирающее атрибуты Mbean

Это правило вернет 9 объектов со следующими полями: MBean1Attribute1, MBean2Attribute1, Mbean3Attribute1,MBean1Attribute2,MBean2Attribute2, Mbean3Attribute2, MBean1Attribute3, MBean2Attribute3, Mbean3Attribute3.

Для получения более подробной информации об объектах, пожалуйста, обратитесь к таблице поддерживаемых макросов, раздел *Обнаружение MBean атрибутов*.

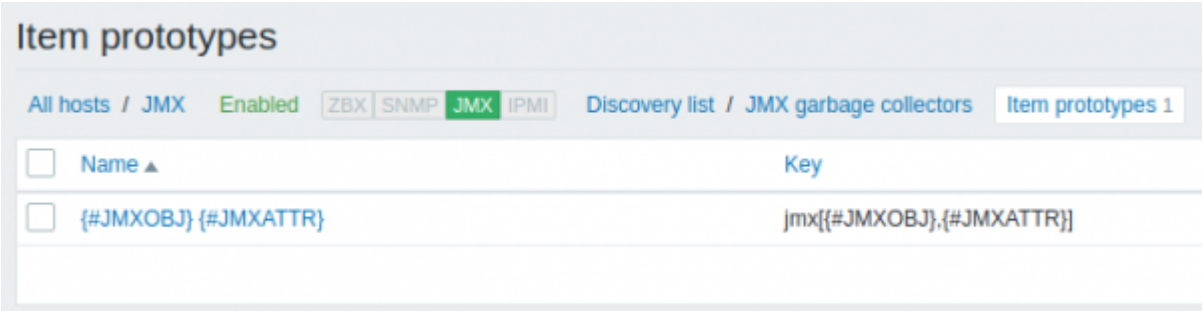
Настройки правила обнаружения, собирающего Mbean атрибуты, выглядят следующим образом:



Используемый ключ:

```
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]
```

Обнаружатся все сборщики мусора с одним элементом атрибута.



В этом конкретном случае элемент данных создастся из прототипа по каждому MBean атрибуту. Основным недостаткой подобной конфигурации является то, что невозможно создать триггер из прототипов триггеров, так как имеется только один прототип элементов данных по всем атрибутам. Таким образом, подобную установку можно использовать для сбора данных, но не рекомендуется использовать для автоматического мониторинга.

From:
https://www.zabbix.com/documentation/current/ - **Zabbix Documentation 5.0**

Permanent link:
https://www.zabbix.com/documentation/current/ru/manual/discovery/low_level_discovery/jmx

Last update: **2019/10/07 06:35**