Визитка

АЛЕКСЕЙ БОЙКО, программист, писатель технических текстов, переводчик

Предупрежден значит вооружен

Мгновенное оповещение системы мониторинга Zabbix на .NET

Zabbix — это система мониторинга сети и серверов с открытым исходном кодом. Центральный компонент — Zabbix-сервер — работает только на ОС типа UNIX. Но это не значит, что Zabbix не подходит для контроля компьютеров под управлением Windows

В данной статье мы рассмотрим одну из возможностей, которая позволяет разработчикам интегрировать свои приложения с Zabbix, на примере C#.NET.

Коротко о главном

Ядро всей системы – это Zabbix-сервер, который накапливает статистику и обрабатывает параметры мониторинга. Все данные о состоянии сети поступают к нему.

Далее будем использовать терминологию Zabbix.

Настройка мониторинга хорошо описана в документации [1]. Тут отметим только, что у «узла сети» (контролируемого компьютера) есть несколько «элементов данных» (контролируемых параметров). Изменения значений «элементов данных» могут вызывать «действия» (например, отправку письма системному администратору).

Каждый «элемент данных» характеризуется не только тем, какие значения он может принимать, но и тем, как он получает эти значения. Например, значение «элемента данных» может устанавливаться с помощью специальных программ, называемых Zabbix-агентами, или с помощью вызова Perl-скрипта (не обязательно Perl).

Таким образом «элемент данных» характеризуется:

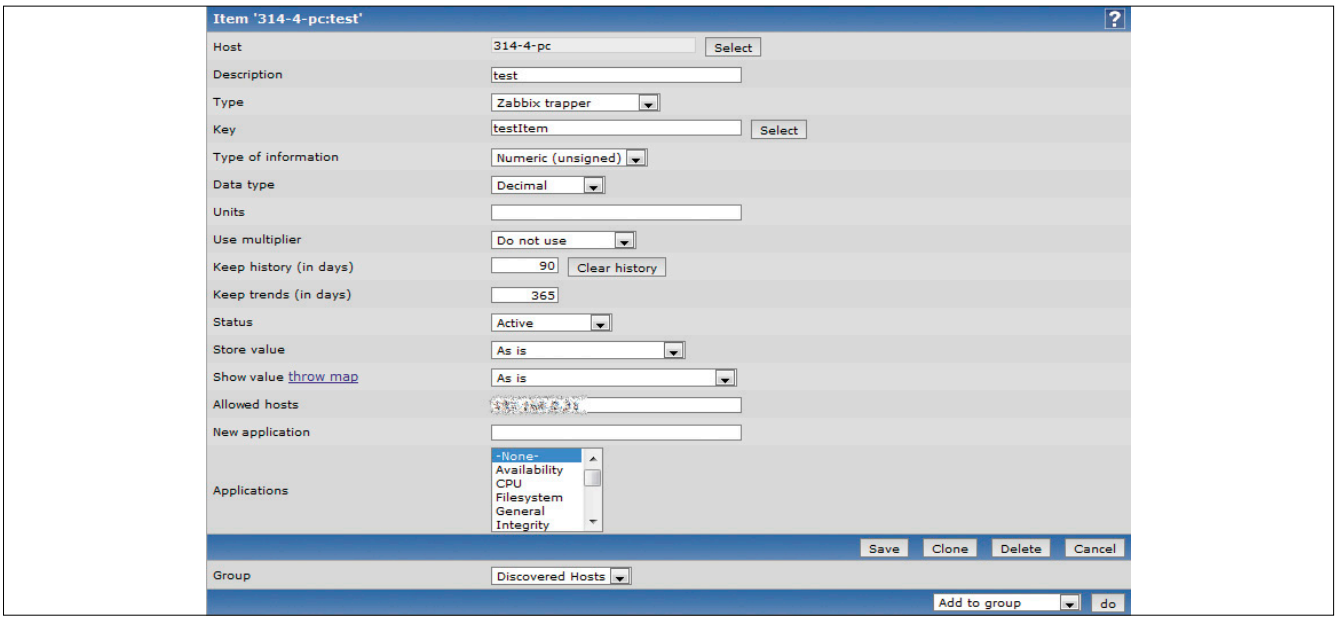
типом возвращаемой информации – «тип информации»;

типом источника информации – «тип элемента данных».

Мгновенное оповещение

Периодический замер контролируемых параметров подходит для многих задач мониторинга. Например, вполне логич-

Рисунок 1. «Элемент данных» типа Zabbix trapper



но периодически проверять доступность корпоративного сайта в Сети.

Кроме периодических проверок, вам может понадобится мгновенное оповещение о случившемся событии. Представьте, что во время оформления заказа на кассе происходит сбой. Вы хотите, чтобы управляющий как можно быстрее разобрался с ситуацией – для этого его нужно оповестить как можно раньше. Кроме того, периодические проверки могут и вовсе не выявить ошибки.

Отметим, что оба способа мониторинга не конкурируют между собой – нельзя сказать, что один лучше другого. У каждого своя область применения. Событийную модель будет трудно реализовать для контроля работоспособности сетевой карты на удаленном компьютере. А периодические проверки не смогут отловить «плавающие», редко возникающие сбои.

Zabbix поддерживает оба варианта мониторинга. В данной статье мы рассмотрим только событийную модель.

Zabbix_sender

Мгновенно оповестить Zabbix-сервер о событии, произошедшем на удаленном компьютере, можно с помощью утилиты Zabbix_sender. Утилита устанавливает значения только «элементов данных» типа Zabbix trapper.

Для экспериментов создайте «элемент данных» типа Zabbix trapper, значения которого могут быть числами («тип информации» Numeric), как показано на рис. 1.

Обратите внимание на поле Allowed hosts – здесь нужно указать IP-адрес компьютера, с которого Zabbix-сервер будет получать сообщения (на нем установлена утилита Zabbix_sender).

Теперь установить значение «элемента данных» можно с помощью вызова утилиты Zabbix_sender:

Листинг 1. Вызов утилиты Zabbix_sender

```
zabbix_sender -z ***.***.***.*** -p 10051 -s 314-4-pc ↵
       -k testItem -o 2
```

где:

- **-z** – IP-адрес Zabbix-сервера;
- **-p** – порт, который слушает Zabbix-сервер (по умолчанию 10051);
- **-s** – название «узла сети», заданное в Zabbix (в английском интерфейсе – host);
- **-k** – название «элемента данных» (поле key);
- **-o** – присваиваемое значение.

На официальном сайте проекта [1] можно скачать Zabbix_sender только для UNIX-систем, однако есть версия и для Windows [2].

Протокол взаимодействия

Zabbix_sender передает сообщения Zabbix-серверу по TCP на порт 10051 (порт по умолчанию). Формат сообщений представлен на рис. 2.

Рисунок 2. Формат сообщений Zabbix

Каждое сообщение начинается с заголовка «ZBXD\x01». Далее следует 8 байт, указывающих, какой объем данных передает сообщение (число в 64-битном формате).

Обратите внимание, что размеры всех частей сообщения четко определены. Данные могут быть произвольного объема, но их размер указан в сообщении. При чтении из TCP сокета важно знать объем получаемой информации (при написании кода мы в этом убедимся).

Данные передаются в виде закодированной JSON-строки. Вызов Zabbix_sender, показанный в листинге 1, передает такие данные (см. листинг 2):

Листинг 2. Формат данных Zabbix на примере сообщения от Zabbix_sender

```
{
    "request":"sender data",
    "data":
    [
        { "host":"314-4-pc", "key":"testItem", "value":2 }
    ]
}
```

Из листинга 2 видно, что в одном запросе можно передать значения для нескольких «элементов данных».

Описанный формат используется не только для отправки сообщений Zabbix-серверу. При формировании ответа Zabbix-сервер следует тем же правилам.

Реализация протокола на .NET

Для мгновенного оповещения Zabbix-сервер о событии из программ на .NET нам нужно, по сути, написать свой Zabbix_sender. Только оформим мы его не в виде утилиты, а в виде библиотеки dll.

План действий:

- Сформировать JSON с данными.
- Посчитать размер данных.
- Сформировать сообщение.
- Отправить сообщение по TCP.
- Прочитать ответ.

Формирование JSON

Листинг 3. Формирование JSON. Тип параметра value может быть любым

```
private void _SetItem(string hostName, string ItemName, ↵
        object value)
{
    JavaScriptSerializer serializer = ↵
        new JavaScriptSerializer();
    string json = serializer.Serialize(
        new
        {
            request = "sender data",
            data = new[]
                {
                    new
                    {
                        host = hostName,
                        key = ItemName,
                        value = value
                    }
                }
        });
    ...
```

Такой прием формирования JSON-строки (см. листинг 3) кажется мне интересным.

Здесь мы:

- не возимся со строками;
- используем анонимные объекты;
- значение «элемента данных» теоретически может быть любым (например, строкой или числом), поэтому мы указываем object в качестве типа параметра value.

Мы сможем использовать один код для передачи Zabbix-серверу значений разных типов («тип информации» в терминологии Zabbix). JavaScriptSerializer сам разберется, как представить объект переданный в качестве параметра value.

Формирование сообщения

Сообщение состоит из трех частей: заголовка, размера данных и самих данных. Тут ничего интересного – для полноты описания приведем код (см. листинг 4).

Листинг 4. Формирование массива байт сообщения

```
byte[] header = Encoding.ASCII.GetBytes("ZBXD\x01");
byte[] length = BitConverter.GetBytes((long)json.Length);
byte[] data = Encoding.ASCII.GetBytes(json);
byte[] all = new byte[header.Length + length.Length + ↵
        data.Length];
System.Buffer.BlockCopy(header, 0, all, 0, header.Length);
System.Buffer.BlockCopy(length, 0, all, header.Length, ↵
        length.Length);
System.Buffer.BlockCopy(data, 0, all, header.Length + ↵
        length.Length, data.Length);
```

Работа с сокетами TCP

С отправкой сообщения никаких хитростей нет, нужно только указать IP-адрес и порт Zabbix-сервера (см. листинг 5).

Листинг 5. Отправка данных с помощью сокета

```
using (var client = new Socket(AddressFamily.InterNetwork, ↵
        SocketType.Stream, ProtocolType.Tcp))
{
    client.Connect(_Ip, _TrapperPort);
    client.Send(all);
    // чтение ответа Zabbix-сервер
    ...
}
```

А вот при чтении из сокета нужно учитывать, что информация может быть разбита на несколько кусков. Нет никакой гарантии, что единичный вызов socket.Receive() вернет все посылаемые сервером данные. Вот почему важно знать размер получаемых данных. Мы на это обращали внимание в разделе «Протокол взаимодействия».

В листинге 6 приведен код, который я нашел в одном блоге, выполняющий чтение из сокета заданного количества байт. Если получение информации не укладывается в отведенный промежуток времени, генерируется исключение.

Листинг 6. Чтение информации из сокета. Размер получаемой информации нужно знать заранее

```
private static void _Receive(Socket socket, byte[] buffer, ↵
        int offset, int size, int timeout)
{
    int startTickCount = Environment.TickCount;
    int received = 0;
    do
    {
        if (Environment.TickCount > startTickCount + timeout)
            throw new Exception("Timeout.");
        try
        {
            received += socket.Receive(buffer, ↵
                offset + received, size - received, ↵
                SocketFlags.None);
        }
        catch (SocketException ex)
        {
            if (ex.SocketErrorCode == ↵
                SocketError.WouldBlock || ↵
                ex.SocketErrorCode == ↵
                SocketError.IOPending || ↵
                ex.SocketErrorCode == ↵
                SocketError.NoBufferSpaceAvailable)
            {
                // Socket buffer is probably empty,
                // wait and try again
                Thread.Sleep(30);
            }
            else
                throw ex;  // Any serious error occurr
        }
    } while (received < size);
}
```

С помощью метода _Receive() мы можем надежно получить ответ Zabbix-сервера (см. листинг 7).

Листинг 7. Надежное получение данных от Zabbix-сервера

```
// Прочитаем заголовок
byte[] buffer = new byte[5];
_Receive(client, buffer, 0, buffer.Length, ↵
        _TcpReceiveTimeout);
if ("ZBXD\x01" != Encoding.ASCII.GetString(buffer, 0, ↵
        buffer.Length))
    throw new Exception("Invalid response");
// Прочитаем длину сообщения
buffer = new byte[8];
_Receive(client, buffer, 0, buffer.Length, ↵
        _TcpReceiveTimeout);
int dataLength = BitConverter.ToInt32(buffer, 0);
if (dataLength == 0)
    throw new Exception("Invalid response");
// Прочитаем сообщение
buffer = new byte[dataLength];
_Receive(client, buffer, 0, buffer.Length, ↵
        _TcpReceiveTimeout);
string result = Encoding.ASCII.GetString(buffer, 0, ↵
        buffer.Length);
```

***

В данной статье мы:

- сделали краткое введение в основы и терминологию системы мониторинга Zabbix;
- подробно разобрали протокол взаимодействия Zabbix-сервера и удаленных клиентов;
- на конкретном примере рассмотрели принципы надежной работы с TCP-сокетами;
- и в качестве небольшого дополнения рассмотрели прием формирования JSON без ручной обработки строк.

В заключение нужно отметить еще один важный момент. К сожалению, Zabbix не предоставляет возможности создавать свои «типы информации». Мы можем присваивать «элементам данных» значения только простых типов: таких, как число и текст. EOF

1. Описание Zabbix, хорошая документация, wiki, живой форум – http://www.zabbix.com.
2. Zabbix_sender для Windows – http://www.suiviperf.com/zabbix.
3. David B. Makofske, Michael J. Donahoo, Kenneth L. Calvert. TCP/IP Sockets in C#. Practical Guide for Programmers.