Визитка

ПАВЕЛ МАЛАХОВ, системный администратор, сторонник свободного ПО

Как читать MIB

Знание протокола SNMP нужно для управления и наблюдения за устройствами в Сети. С его помощью администратор может автоматизировать сбор статистики с ключевых узлов Сети. Рассмотрим на примерах, как понимать и использовать ключевое понятие в SNMP-протоколе — базу данных MIB

MIB – это структурированный текстовый файл или несколько файлов, которые содержат информацию обо всех объектах устройства. Объектом может быть какая-нибудь настройка или параметр системы. У каждого объекта есть свой набор полей, таких как тип данных, доступность (чтение, запись), статус (обязательный, необязательный), текстовое название (Object Name) и цифровой адрес (OID). Также объект может содержать другие объекты.

Есть стандартные MIB, определяемые различными RFC, и огромное множество MIB от производителей оборудования, которые дополняют стандартные и могут быть взяты с сайтов этих компаний. Эти дополнения необходимы, чтобы описать специфические для устройства параметры. Можно также составить и свои MIB, нигде их не регистрировать и успешно использовать.

SNMP-менеджер, используя OID, способен считывать или устанавливать значение объекта. Например, адрес объекта (OID), содержащий название системы: 1.3.6.1.2.1.1.5, а его имя (object name): sysName.

Так как все общение между SNMP-агентом устройства и SNMP-менеджером (системой наблюдения или администратором) происходит через OID, то понимать, что они описывают, очень даже полезно.

Имя объекта играет ту же роль в SNMP, что и DNS-имя в IP-сетях, – более наглядное описательное представление набора чисел. Строго говоря, в разных MIB оно может представлять разные OID, хотя те, что описаны в RFC, по идее должны быть уникальными для всех.

Общая информация

Для начала кратко опишем некоторые важные термины протокола SNMP (Simple Network Managment Protocol):

MIB (Managment Information Base) – база данных информации управления, хранящая информацию обо всех объектах (параметрах и настройках) устройства;

OID (Object IDentificator) – числовой идентификатор объекта в дереве MIB;

Object Name – имя объекта, уникальная константа для всего MIB, однозначно соответствующая определенному OID.

Как читать OID

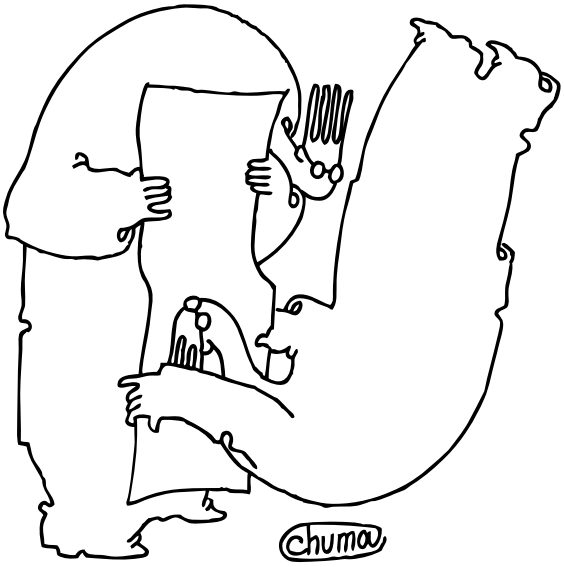
Вышеприведенный OID (1.3.6.1.2.1.1.5) для объекта sysName построен целиком на стандартном MIB и будет существовать скорее всего на всех устройствах. Он читается следующим образом:

1: iso – International Organization for Standardization (ISO);
3: identified-organization – схема определения организации согласно ISO/IEC 6523-2 [1];
6: dod – United States Department of Defense (DoD), эта организация изначально занималась стандартизацией протокола;
1: internet – Интернет;
2: mgmt – IETF Management;
1: mib-2 – база OID для спецификации MIB-2;
1: system – характеристики системы;
5: sysName – имя системы.

OID специфичного объекта для конкретного устройства, дополненный своими MIB, будет значительно длиннее. Вот пример OID датчика температуры у первого вентилятора в Intel Modular Server: 1.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16.1. Первые семь параметров – из стандартных MIB, остальные 10 – из MIB Intel.

Четыре первых мы уже расшифровали выше, остальные поясняются следующим образом:

4: private – частные проекты;
1: enterprise – частные организации;
343: intel – этот номер закреплен за компанией Intel;
2: products – продукты;
19: modularsystems – серверы линейки Modular System;
1: multiFlexServer – тип сервера Multi-Flex Server;
2: components – компоненты;
10: chassis – контейнер для информации об аппаратном блоке;
206: fans – вентиляторы;
1: fanFruTable – таблица вентиляторов;
1: fanFruEntry – информация о вентиляторе;
16: fanFruInletTemperature – температура возле вентилятора;
1: ? – датчик возле первого вентилятора.



Зная, как читать MIB и OID, администратору будет легче настраивать и использовать системы мониторинга системы

Вся описательная информация находится как раз в текстовых файлах MIB, поэтому давайте разберемся, как их читать.

Как читать MIB

Как уже говорилось выше, при работе с удаленной системой по протоколу SNMP все запросы происходят через OID, отражающий положение объекта в дереве объектов MIB.

Например, все OID системы можно получить, просканировав устройство, например, командой snmpwalk:

```
$ snmpwalk -c public -v2c 10.0.0.1
```

где:

-c public – обращение к сообществу (comminity) public, на многих устройствах оно существует по умолчанию и в режиме только для чтения;

v2c – использовать вторую версию протокола SNMP;

10.0.0.1 – IP-адрес устройства.

К сожалению, иногда данная команда не успевает вытащить все переменные, так как на некоторых устройствах их достаточно много, и защита от DOS-атак срабатывает раньше, блокируя доступ на некоторое время. Поэтому данные иногда удобнее получать частично, лишь для определенной ветки:

```
$ snmpwalk -c public -v2c 10.0.0.1  ↵
        1.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16
```

Однако полученные цифровые значения часто не раскрывают своего предназначения, поэтому возникает обратная задача: узнать, какой OID у интересующего нас объекта, чтобы потом по нему делать запросы. Для этого придется изучать MIB устройства.

Например, для того чтобы узнать температуру в корпусе Intel Modular Server, сделаем сквозной поиск слова «temperature» по всем файлам MIB, поставляемым вместе с этим сервером.

В результате находим объект fanFruInletTemperature. Смотрим его описание. Вот нужный нам фрагмент:

```
--
-- fans tree
--

    fans OBJECT-IDENTITY
        STATUS current
        DESCRIPTION
            "Container for Fan specific information as
            well as all components logically contained
            within."
      ::= { chassis 206 }


    fanFruTable OBJECT-TYPE
        SYNTAX SEQUENCE OF FanFruEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Each row describes a Fan FRU in the chassis
            Current, this should include the 3 Fan FRUs
            (the two system
            Fan FRUs and the I/O Fan FRU)
            All system FRUs rows will be present"
      ::= { fans 1 }


    fanFruEntry OBJECT-TYPE
        SYNTAX FanFruEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            ".."
        INDEX { fanFruIndex }
      ::= { fanFruTable 1 }


    FanFruEntry ::=
        SEQUENCE {

            fanFruIndex             Index,
            fanFruPresence          Presence,
            fanFruVendor            DisplayString,
            fanFruMfgDate           DisplayString,
            fanFruDeviceName        DisplayString,
            fanFruPart              IdromBinary16,
            fanFruSerialNo          IdromBinary16,
            fanFruMaximumPower      Power,
            fanFruNominalPower      Power,
            fanFruAssetTag          IdromBinary16,
            fanFruPowerLed          PowerLedStates,
            fanFruFaultLed          FaultLedStates,
            fanFruOpCodeVersio      DisplayString,
```

```
            fanFruBootBlockVersion DisplayString,
            fanFruNumOfFans        INT32withException,
            fanFruInletTemperature INT32withException

        }
```

Строка в описании объекта fans «::= { chassis 206 }» говорит о том, что описанный объект будет расширять объект (являться веткой в дереве объектов) chassis, имея в нем индекс 206, а следующий объект fanFruTable, в свою очередь, будет расширять объект fans, представляя в нем ветку с индексом 1, также fanFruEntry будет первой веткой у объекта fanFruTable. В параметрах fanFruEntry и содержится интересующий нас fanFruInletTemperature.

Запоминаем адрес ветки: 206.1.1 начиная от объекта chassis. Теперь ищем далее в файле описание объекта fanFruInletTemperature:

```
fanFruInletTemperature OBJECT-TYPE
        SYNTAX INT32withException
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "FRU Inlet Temperature in Degrees Celsius"
        ::= { fanFruEntry 16 }
```

Видим, что его индекс равен 16. Таким образом, 206.1.1.16 содержит список всех температурных датчиков возле вентиляторов системы, и соответственно 206.1.1.16.1 – номер первого из них. Сколько их всего, узнаем позже.

Теперь выясним адрес родительского объекта chassis, в котором находятся только что найденные нужные нам параметры. Проведем сквозной поиск строки chassis OBJECT-IDENTITY по всем MIB-файлам приводит нас к другому MIB:

```
chassis OBJECT-IDENTITY
        STATUS current
        DESCRIPTION
            "chassis for the Multi-Flex Server product.
            Container for chassis specific information
            as well as all components logically contained
            within."
        ::= { components 10 }
```

где мы узнаем, что он содержится в объекте components.

Далее сквозной поиск строки components OBJECT-IDENTITY (нужно учесть, что пробелов между словами может быть разное количество) дает строчку:

```
components    OBJECT IDENTIFIER ::= { multiFlexServer 2 }
```

Аналогично находим и остальное:

```
--
-- Path to our product
--

 intel           OBJECT IDENTIFIER ::= { enterprises 343 }
 products        OBJECT IDENTIFIER ::= { intel 2 }
 modularsystems  OBJECT IDENTIFIER ::= { products 19 }
 multiFlexServer OBJECT IDENTIFIER ::= { modularsystems 1 }
```

Записывая все полученные ID объектов, получаем полный OID для температурных датчиков: 1.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16.

Теперь можно узнать их значения, заодно выяснив и их количество:

```
$ snmpwalk -c public -v2c   10.0.0.1   ↵
        1.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16

iso.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16.1 = INTEGER: 27
iso.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16.2 = INTEGER: 26
iso.3.6.1.4.1.343.2.19.1.2.10.206.1.1.16.3 = INTEGER: 19
```

По приведенному несложному алгоритму можно прочитать любой MIB. Конечно, предварительно его надо получить, а это, к сожалению, не всегда возможно.

Для облегчения работы с MIB-файлами существует множество программ как платных, так и бесплатных, в том числе и on-line [2, 3]. Любой поисковик на запрос «MIB browser» выдаст много полезных ссылок. Я пользуюсь iReasoning MIB Browser, но не потому, что он лучше других, а просто я попробовал его первый, и он мне вполне понравился.

***

Теперь, зная, как читать MIB и OID, администратору будет легче использовать и донастраивать системы мониторинга здоровья системы, такие как Zabbix, MRTG, PRTG, Cacti и т.п. EOF

1. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=25774.
2. http://www.oid-info.com/get – поиск по OID с детальным пояснением всех узлов. Есть возможность расширить список своими MIB. Можно посмотреть дерево объектов.
3. http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en – поиск по OID и Object Name. Кроме стандартных, есть специфичные для Cisco MIB.