net-snmp的MIBs扩展添加步骤

# 编写MIB文件

MIB文件是用 ASN.1 语法来描述的，所以为了精确定义MIB中各管理对象，用户不得不参考一些ASN.1语法的有关文档如RFC1155、RFC1212等等来定义设备自己的MIB。ASN.1是抽象句法表示法一 （Abstract SyntaxNotation One） 的简称，对于每个管理对象它都用文本来描述，一般文件的后缀名都用“.mib”,在net-snmp中后缀名是“.mib.txt”。

关于MIB文件示例，可以见编译安装后的net-snmp目录，一般是 /usr/local/net-snmp/share/snmp/mibs/。

ls/usr/local/net-snmp/share/snmp/mibs/

AGENTX-MIB.txt                       IPV6-TCP-MIB.txt                     SNMP-NOTIFICATION-MIB.txt

...

IPV6-MIB.txt                         SNMP-MPD-MIB.txt                     UDP-MIB.txt

这里我们建立一个mib文件，命名为TEST-GET-MIB.txt，放在/usr/local/net-snmp/share/snmp/mibs/目录下因为这个目录是snmpd的默认目录，只要把MIB库放入该目录就可以自动加载MIB库，否则需要修改snmpd.conf文件，自定义的MIB文件如下：

--开始

TEST-GET-MIB DEFINITIONS ::= BEGIN

--引入部分

IMPORTS

enterprises

FROM RFC1155-SMI

Integer32,OBJECT-TYPE

FROM SNMPv2-SMI

DisplayString

FROM SNMPv2-TC

TEXTUAL-CONVENTION

FROM SNMPv2-TC; --引用结束，用分号

--定义节点

--enterprises的OID是1.3.6.1.4

testGet OBJECT IDENTIFIER ::= { enterprises 77695 }

GetTime OBJECT IDENTIFIER ::= { testGet 1}

GetTime OBJECT-TYPE --对象名称

SYNTAX DisplayString --类型

MAX-ACCESS read-only --访问方式

STATUS current --状态

DESCRIPTION --描述

"get current time"

::= { testGet 1 } --父节点

--结束定义

END

写完后我们测一个MIB库有没有问题，在linux机器上用snmptranslate-Tp -IR TEST-GET-MIB::testGet显示结果如下：（这个测试不需要启动snmpd进程）

[root@localhostbin]# ./snmptranslate -Tp -IR TEST-GET-MIB::testGet

+--testGet(77695)

|

+-- -R-- String GetTime(1)

Textual Convention: DisplayString

Size: 0..255

OK,snmp自动发现了这个MIB库， 有了自定义的OID，接下来开始添加处理程序。

# 生成源代码

我们可以先来获取一下前面定义的 testGet 节点的值试试。 因为 enterprises 的OID是 1.3.6.1.4 ，而 testGet是 enterprises 的叶子(77695)，而 GetTime 又是 testGet 的叶子节点(1)。所以其OID为 1.3.6.1.4.77695.1 。

下面使用snmpget来测试一下（测试之前要先启动snmpd进程）

[root@localhostbin]# ./snmpget -c public -v 2c localhost 1.3.6.1.4.1.77695.1.0

SNMPv2-SMI::enterprises.77695.1= No Such Object available on this agent at this OID

结果是No Such Object available on this agent at this OID，我们需要用mib2c程序生成所需要的.c和.h文件。

执行env  MIBS="+/usr/local/net-snmp/share/snmp/mibs/TEST-GET-MIB.txt" ./mib2c  testGet，会引导你逐渐生成testGet.h和testGet.c,先选2再选1，过程如下：

[root@localhost bin]# env MIBS="+/usr/local/net-snmp/share/snmp/mibs/TEST-GET-MIB.txt" ./mib2c testGet

writing to -

mib2c has multiple configuration files depending on the type of

code you need to write. You must pick one depending on your need.

You requested mib2c to be run on the following part of the MIB tree:

OID: testGet

numeric translation: .1.3.6.1.4.1.77695

number of scalars within: 1

number of tables within: 0

number of notifications within: 0

First, do you want to generate code that is compatible with the

ucd-snmp 4.X line of code, or code for the newer Net-SNMP 5.X code

base (which provides a much greater choice of APIs to pick from):

1) ucd-snmp style code

2) Net-SNMP style code

Select your choice : 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GENERATING CODE FOR SCALAR OBJECTS:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It looks like you have some scalars in the mib you requested, so I

will now generate code for them if you wish. You have two choices

for scalar API styles currently. Pick between them, or choose not

to generate any code for the scalars:

1) If you're writing code for some generic scalars

(by hand use: "mib2c -c mib2c.scalar.conf testGet")

2) If you want to magically "tie" integer variables to integer

scalars

(by hand use: "mib2c -c mib2c.int\_watch.conf testGet")

3) Don't generate any code for the scalars

Select your choice: 1

using the mib2c.scalar.conf configuration file to generate your code.

writing to testGet.h

writing to testGet.c

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* NOTE WELL: The code generated by mib2c is only a template. \*YOU\* \*

\* must fill in the code before it'll work most of the time. In many \*

\* cases, spots that MUST be edited within the files are marked with \*

\* /\* XXX \*/ or /\* TODO \*/ comments. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

running indent on testGet.c

running indent on testGet.h

mib2c已经统计出我们的mib库包含1个scalar变量，0个table变量，0个通知变量，Scalar就是包含我们常用的整型，字符串，时间等等数据类型。table就是scalar的一种集合，有一个和多个列组成，类似于数据库中的表。它必须具有索引项，用来按一定顺序检索表项，当然我们只写了一个标量的OID，不是表结构也不是通知结构

生成的testGet.h如下

/\*

\* Note: this file originally auto-generated by mib2c using

\* $

\*/

#ifndef TESTGET\_H

#define TESTGET\_H

/\*

\* function declarations

\*/

void init\_testGet(void);

Netsnmp\_Node\_Handler handle\_GetTime;

#endif /\* TESTGET\_H \*/

生成的testGet.c文件如下：

/\*

\* Note: this file originally auto-generated by mib2c using

\* $

\*/

#include <net-snmp/net-snmp-config.h>

#include <net-snmp/net-snmp-includes.h>

#include <net-snmp/agent/net-snmp-agent-includes.h>

#include "testGet.h"

/\*\* Initializes the testGet module \*/

void

init\_testGet(void)

{

/\*

\* Note: this file originally auto-generated by mib2c using

\* $

\*/

#include <net-snmp/net-snmp-config.h>

#include <net-snmp/net-snmp-includes.h>

#include <net-snmp/agent/net-snmp-agent-includes.h>

#include "testGet.h"

/\*\* Initializes the testGet module \*/

void

init\_testGet(void)

{

const oid GetTime\_oid[] = { 1, 3, 6, 1, 4, 1, 77695, 1 };

DEBUGMSGTL(("testGet", "Initializing\n"));

netsnmp\_register\_scalar(netsnmp\_create\_handler\_registration

("GetTime", handle\_GetTime, GetTime\_oid,

OID\_LENGTH(GetTime\_oid), HANDLER\_CAN\_RONLY));

}

int

handle\_GetTime(netsnmp\_mib\_handler \*handler,

netsnmp\_handler\_registration \*reginfo,

netsnmp\_agent\_request\_info \*reqinfo,

netsnmp\_request\_info \*requests)

{

/\*

\* We are never called for a GETNEXT if it's registered as a

\* "instance", as it's "magically" handled for us.

\*/

/\*

\* a instance handler also only hands us one request at a time, so

\* we don't need to loop over a list of requests; we'll only get one.

\*/

switch (reqinfo->mode) {

case MODE\_GET:

snmp\_set\_var\_typed\_value(requests->requestvb, ASN\_OCTET\_STR,

/\*

\* XXX: a pointer to the scalar's data

\*/ ,

/\*

\* XXX: the length of the data in bytes

\*/ );

break;

default:

/\*

\* we should never get here, so this is a really bad error

\*/

snmp\_log(LOG\_ERR, "unknown mode (%d) in handle\_GetTime\n",

reqinfo->mode);

return SNMP\_ERR\_GENERR;

}

return SNMP\_ERR\_NOERROR;

}

以上的代码都是自动生成的，我们没有写一行代码，到了这一步，我们需要把testGet.c里面的XXX改成自己的值，也就两行，修改后testGet.c文件代码如下：

/\*

\* Note: this file originally auto-generated by mib2c using

\* $

\*/

#include <net-snmp/net-snmp-config.h>

#include <net-snmp/net-snmp-includes.h>

#include <net-snmp/agent/net-snmp-agent-includes.h>

#include "testGet.h"

/\*\* Initializes the testGet module \*/

void

init\_testGet(void)

{

const oid GetTime\_oid[] = { 1, 3, 6, 1, 4, 1, 77695, 1 };

DEBUGMSGTL(("testGet", "Initializing\n"));

netsnmp\_register\_scalar(netsnmp\_create\_handler\_registration

("GetTime", handle\_GetTime, GetTime\_oid,

OID\_LENGTH(GetTime\_oid), HANDLER\_CAN\_RONLY));

}

int

handle\_GetTime(netsnmp\_mib\_handler \*handler,

netsnmp\_handler\_registration \*reginfo,

netsnmp\_agent\_request\_info \*reqinfo,

netsnmp\_request\_info \*requests)

{

/\*

\* We are never called for a GETNEXT if it's registered as a

\* "instance", as it's "magically" handled for us.

\*/

/\*

\* a instance handler also only hands us one request at a time, so

\* we don't need to loop over a list of requests; we'll only get one.

\*/

time\_t t;

switch (reqinfo->mode) {

case MODE\_GET:

time(&t);

char szTime[100];

snprintf(szTime,100,"%s",ctime(&t));

snmp\_set\_var\_typed\_value(requests->requestvb, ASN\_OCTET\_STR,

/\*

\* XXX: a pointer to the scalar's data

\*/ szTime,

/\*

\* XXX: the length of the data in bytes

\*/ strlen(szTime));

break;

default:

/\*

\* we should never get here, so this is a really bad error

\*/

snmp\_log(LOG\_ERR, "unknown mode (%d) in handle\_GetTime\n",

reqinfo->mode);

return SNMP\_ERR\_GENERR;

}

return SNMP\_ERR\_NOERROR;

}

# 配置编译

把testGet.c和testGet.h复制到/net-snmp-5.7.3/agent/mibgroup/路径下

设置编译参数（红色部分即为加上我们自己的mib模块）

[root@localhostnet-snmp-5.7.3]# ./configure  --prefix=/usr/local/net-snmp   --with-mib-modules="testGet"

查看文件net-snmp-5.7.3/agent/mibgroup/mib\_module\_inits.h，发现已经添加到初始化中。

/\*This file is automatically generated by configure.  Do not modify by hand. \*/

  if (should\_init("testGet"))init\_testGet();

编译并安装

[root@localhostnet-snmp-5.7.3]# make

[root@localhostnet-snmp-5.7.3]# make install

# 测试新加的MIB

启动snmpd服务

[root@localhostsbin]# ./snmpd -f -Le

Turning on AgentXmaster support.

NET-SNMP version5.7.3

No Shmem line in/proc/meminfo

我们再调用snmpget来测试结果：

[root@localhostbin]# ./snmpget  -v2c -c public localhost  1.3.6.1.4.1.77695.1.0

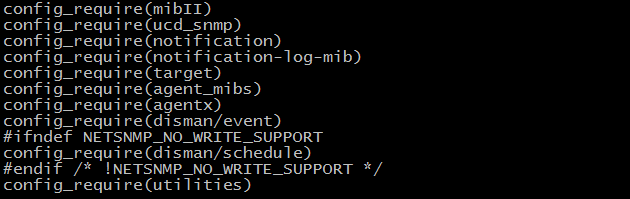
SNMPv2-SMI::enterprises.77695.1.0= STRING: "Fri Nov  3 14:02:13 2017

显示出了当前的时间，说明我们添加自定义MIB get操作成功！

# Configure执行流程

## 默认编译模块

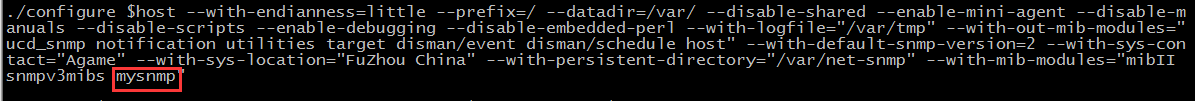
默认编译agent/mibgroup/default\_modules.h下的模块



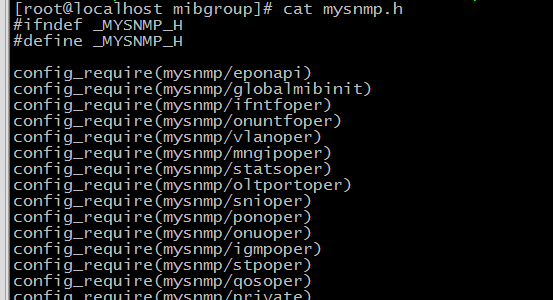
## 加入模块选项

加入编译模块选项 --with-mib-modules="list"

以mysnmp模块为例



先找到agent/mibgroup/mysnmp.h文件



然后将初始化接口写入 agent/mibgroup/mib\_module\_inits.h 中（必须有相同名字的.c 或 .h 文件）

接口名字统一为 init\_模块名

