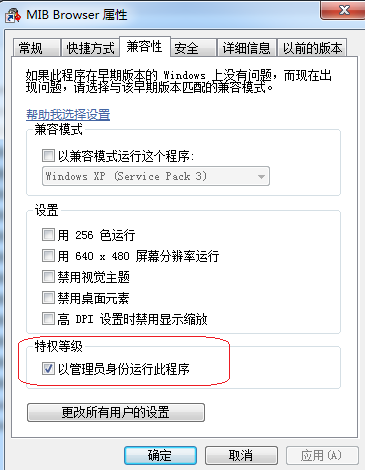
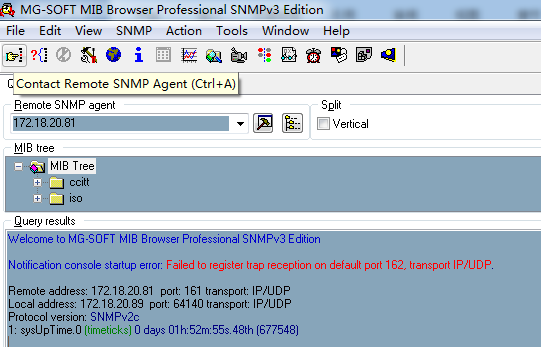
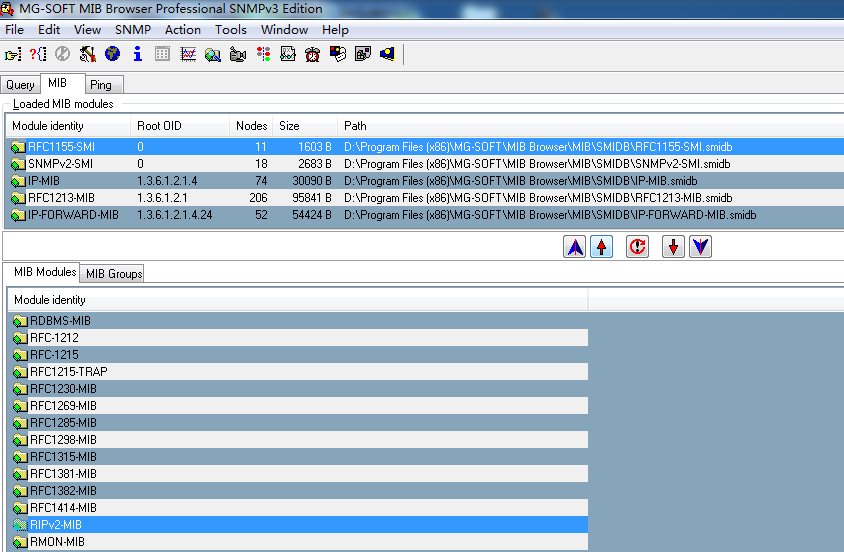
Mib browser的使用----冯松柏

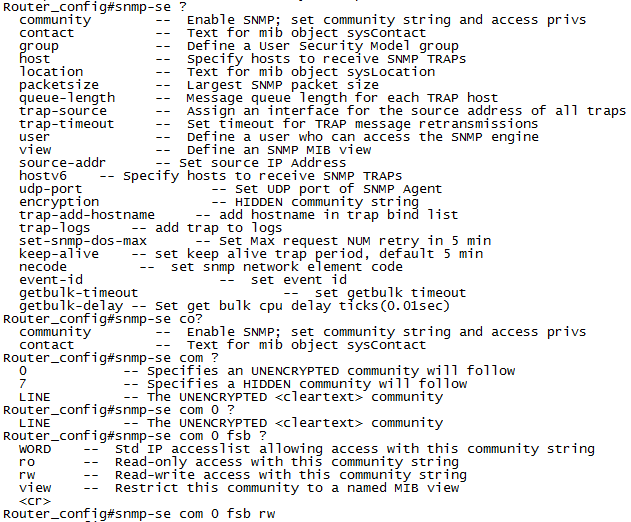
本PC地址为172.18.20.89，DUT地址为172.18.20.87,需要在DUT上启动SNMP server，在本机勾选mib browser以管理员账号运行。





添加mib文件，比如rip的





一个例子：

路由器上做为snmp agent，其配置如下

snmp-server community 0 fsb RW //在mibbrowser上，也需要配置对应的用户名，口令， 均为agent端的设置，fsb

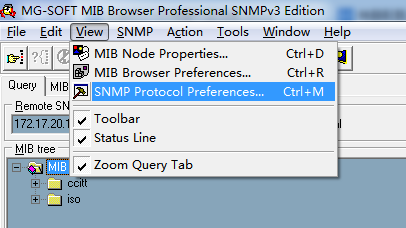
snmp-server host 172.18.20.89 fsb\_trap //必须指定mib browser所在的服务器地址。

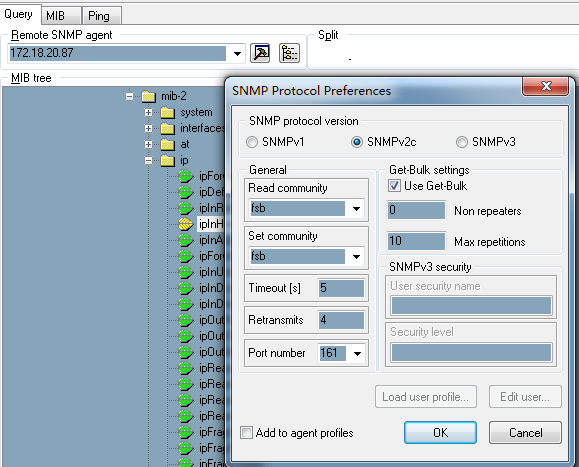
snmp-server trap-source GigaEthernet0/0

snmp-server trap-add-hostname

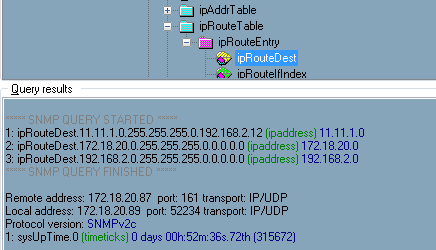
snmp-server trap-logs

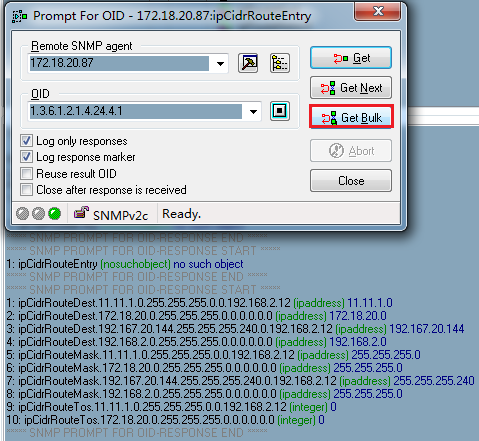
!



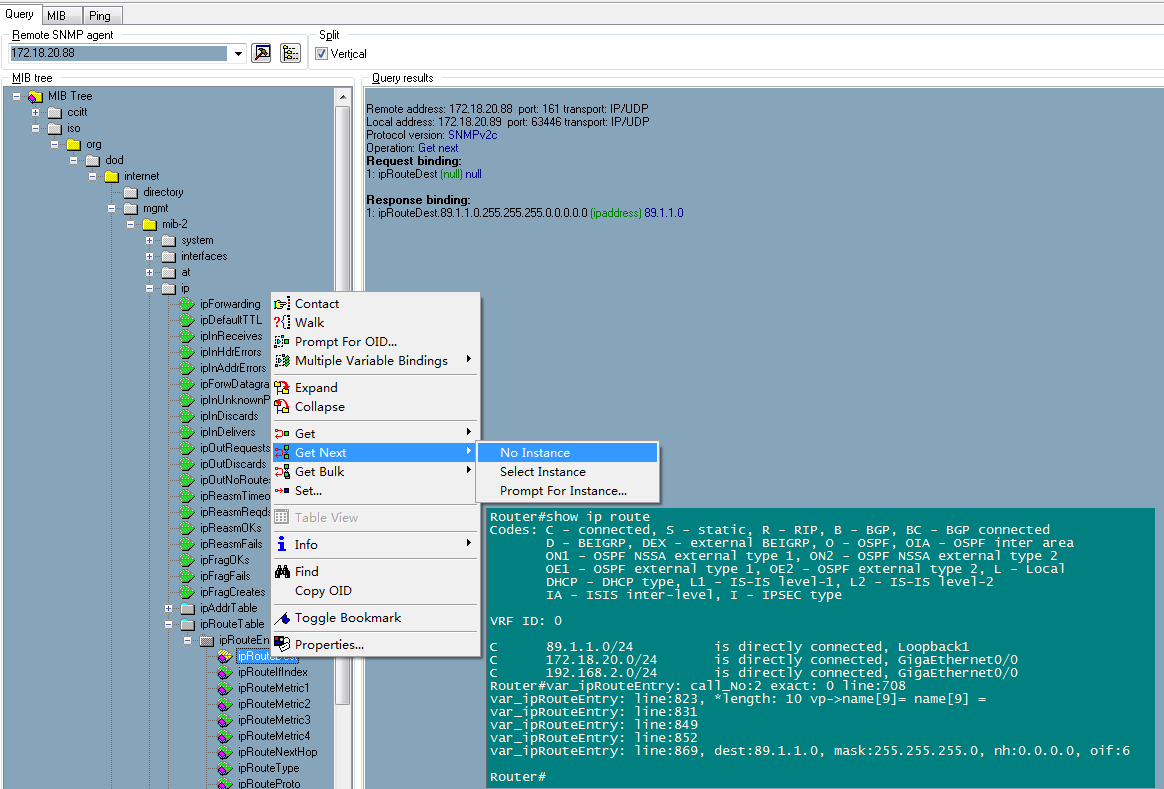


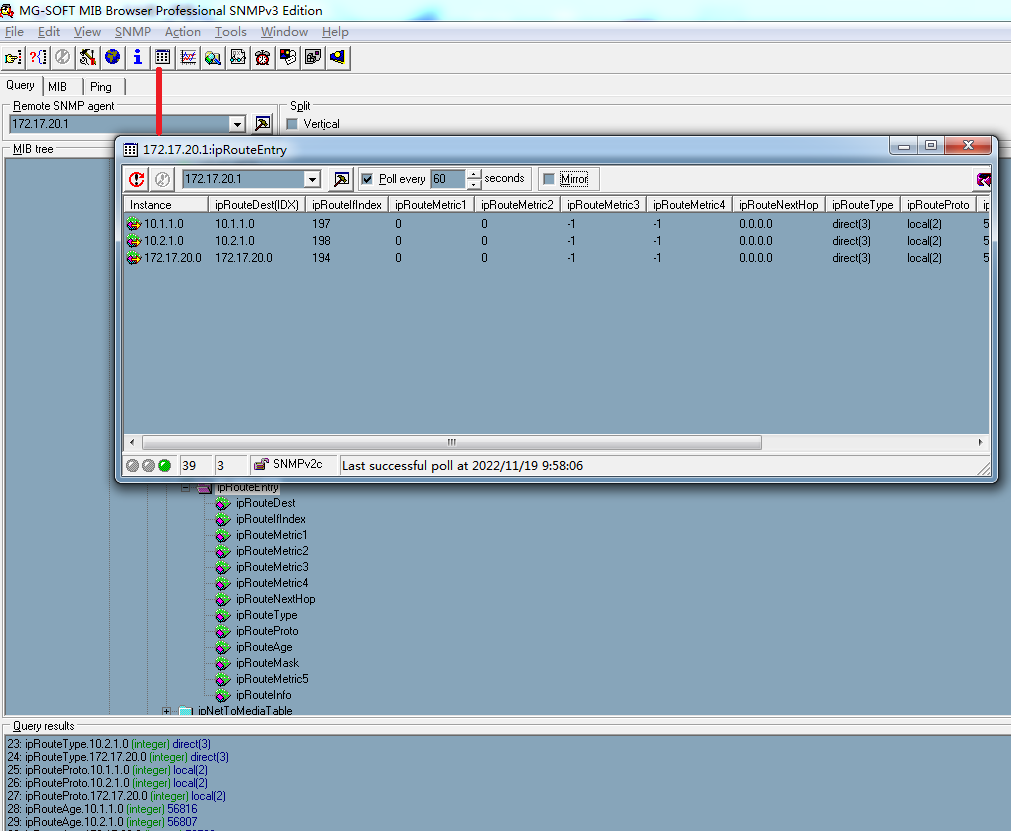
以上设置完毕以后，可以正常启动并register到mib server，并读取需要的信息。





要从mib browser对设备侧写操作，有set操作，以及新增操作，而新增操作需要设备侧对行状态的支持，首先创建一个空的行，然后才能做下一步的新增各字段。





和你遇见了同样的问题，最后搞定了，操作如下：  
1、网管监控端服务器需要禁用windows操作系统的SNMP TRAP服务  
并确保MG-SOFT的SNMP TRAP服务处于启动状态  
2、关闭操作系统的防火墙，或者在高级选项里添加入站规则，允许UDP 162端口消息进入即可。

发布于：2014.02.21 08:57

[sinat\_28022939](https://my.csdn.net/sinat_28022939" \t "https://ask.csdn.net/questions/_blanlk) 我抓包显示，数据确实发到windows计算机上的，而且windows7是启用了trap服务，也允许了162端口进入消息，但是mib browser就是显示不了SNMP trap消息，怎么解决呢？

[sinat\_28022939](https://my.csdn.net/sinat_28022939" \t "https://ask.csdn.net/questions/_blanlk) 请问怎么确保MG-SOFT 的SNMP trap服务处于启动状态啊？ 还有网管监控服务器怎么禁用windows 操作系统的SNMP TRAP服务呢？

[回复](https://ask.csdn.net/questions/javascript:void(0);)

主要是保障MIB Browser的162端口是处于开启的状态，MIB Browser Preferences中的“Trap Ringer”中的“port”的状态应该是Ready既可。  
若不是的话，则应该在DOS中看端口162被哪个程序占用了，将那个程序kill既可；  
至于防火墙，高级选项里添加入站规则，允许UDP 162端口消息进入即可

[Win7下SNMP服务的配置](http://blog.chinaunix.net/uid-24058189-id-2105677.html)

分类： WINDOWS

2011-08-07 23:42:17

SNMP服务的安装在控制面板中勾选Windows的简单服务管理即可，如下图

在Windows服务组件中将会新增二个服务，分别是“SNMP Service”和“SNMP Trap”，不过在我尝试用Mib Browser查询相关信息时出现了"Cantot connect to remote SNMP agent"的错误，很明显SNMP服务已经被开启，但是请求之后仍无法找到SNMP Agent。

在进行一番摸索之后，很自然地想到是不是SNMP的Community的设置在Win7中有了特别处理。右键"SNMP Service"->属性->安全选项卡，果然其中社区一栏为空。添加Community即可。

win7下面使用MG-SOFT MIB Browser不能接收snmptrap

1、网管监控端服务器需要禁用windows操作系统的SNMP TRAP服务  
并确保MG-SOFT的SNMP TRAP服务处于启动状态  
2、关闭操作系统的防火墙，或者在高级选项里添加入站规则，允许UDP 162端口消息进入即可。

# mib浏览器中的trap-receiver 不能绑定162端口

2017年10月20日 10:27:45 [BABYWANG1128](https://me.csdn.net/BABYWANG1128" \t "https://blog.csdn.net/BABYWANG1128/article/details/_blank) 阅读数：858

打开mib浏览器中的trap-receiver时弹出err提示框，提示你不能绑定162端口。

此时需要打开电脑的终端输入：net stop snmptrap

这事就不会再提示不能绑定的错误提示了；

PS：接收trap数据之前需要先get/set一下数据，我也不知道原因，猜测是建立连接之类的吧，如果有人知道详细的原理还请给说一下~

目 录

**[1  MIB简介](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_vrp_mib_overview_0001)**

[1.1  MIB概述](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0002)

[1.1.1  MIB介绍](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0003)

[1.1.2  MIB分类](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0004)

[1.2  MIB的相关概念](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0008)

[1.2.1  MIB的存储结构](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0005)

[1.2.2  MIB文件介绍](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0006)

[1.3  MIB访问流程](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0009)

1  MIB简介

[1.1 MIB概述](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0002)  
介绍了MIB的概念、作用以及MIB的分类。

[1.2 MIB的相关概念](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0008)  
介绍了MIB树、MIB对象及节点和MIB文件的相关概念。

[1.3 MIB访问流程](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "dc_ar_mib_operation_0009)  
简要介绍了访问MIB的整个操作流程。

1.1  MIB概述

介绍了MIB的概念、作用以及MIB的分类。

1.1.1  MIB介绍

随着网络的规模越来越庞大，网络中的设备种类繁多，如何对越来越复杂的网络进行有效的管理，从而提供高质量的网络服务已成为网络管理所面临的最大挑战。

在上述情景下，网络管理员对设备的管理效率被时间、场地、设备数量等因素制约。通过命令行方式下发配置、维护命令可以一定程度的缓解这些问题，但有些时候会给实施者带来很多繁杂、冗余的操作，甚至在一些情况下，命令行操作并不能实现特定的功能。

为了解决上述问题，基于简单网络管理协议SNMP（Simple Network Management Protocol）的网络管理在TCP/IP网络中被广泛应用。

基于SNMP的网络管理体系结构中包含4个主要组成部分：

网络管理站NMS（Network Management Station）

NMS通常是一个独立的设备，运行网络管理应用程序。网络管理应用程序至少能够提供一个人机交互界面，网络管理员通过它完成绝大多数网络管理工作。

SNMP代理器（Agent）

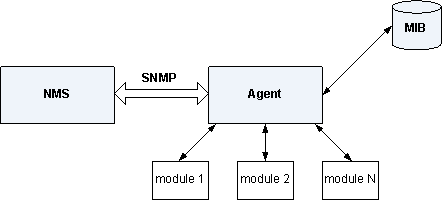
Agent是驻留在被管理设备的一个软件模块，主要负责接收和处理来自NMS的请求报文，并形成响应报文，返回给NMS，在一些紧急情况下，它会主动发送Trap报文，通知NMS。

SNMP协议

SNMP协议属于TCP/IP网络的应用层协议，用于在NMS和被管理设备间交互管理信息。

管理信息库MIB（Management Information Base）

MIB是一个被管理对象的集合，是NMS同Agent进行沟通的桥梁，可以使网管软件和设备进行标准对接。每一个Agent都维护这样一个MIB库，NMS可以对MIB库中对象的值进行读取或设置。

图1-1  基于SNMP网络管理的示意图   


从[图1-1](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "fig_dc_vrp_mib_overview_000301)可以了解网络管理中涉及到的几个主要组成部分的相互关系，它们之间的通信方式描述如下：

NMS通过SNMP协议与设备的Agent通信，完成对MIB的读取和修改操作，从而实现对网络设备的监控与管理。

SNMP是NMS与Agent之间通信的载体，通过其协议数据单元PDU（Protocol Data Unit）完成信息交换。SNMP并不负责数据的实际传输，数据交换的任务是通过UDP等传输层协议来完成的。

Agent是设备上的代理进程，主要工作包括与NMS通信，对设备中的MIB库进行维护，以管理和监控设备中的各个模块。

MIB保存设备中各个模块的信息。通过对MIB信息的读写操作来完成对设备的监控和维护。

MIB主要用途是通过SNMP查询、修改代理MIB中相应对象的值，实现对系统资源的监视和控制。

1.1.2  MIB分类

MIB可以分为公有MIB和私有MIB两种。

公有MIB：一般由RFC定义，主要用来对各种公有协议进行结构化设计和接口标准化处理。例如：OSPF-MIB（RFC1850）/BGP4-MIB（RFC1657）都是典型的公有MIB。大多数的设备制造商都需要按照RFC的定义来提供SNMP接口。

私有MIB：是公有MIB的必要补充，当公司自行开发私有协议或者特有功能时，可以利用私有MIB来完善SNMP接口的管理功能，同时对第三方网管软件管理存在私有协议或特有功能的设备提供支持。

公有MIB和私有MIB相互独立，不会存在公有MIB节点和私有MIB节点混在同一个MIB的情况。

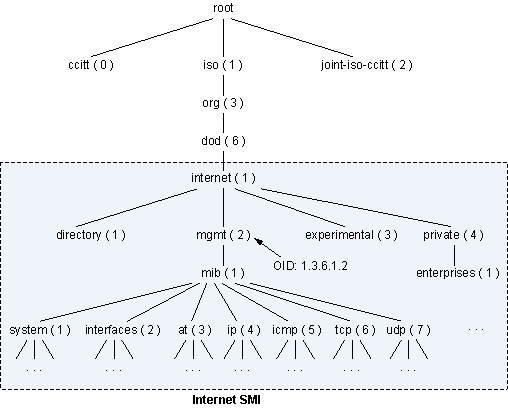
1.2  MIB的相关概念

介绍了MIB树、MIB对象及节点和MIB文件的相关概念。

1.2.1  MIB的存储结构

MIB树结构

MIB以树状结构进行存储，树的叶子节点表示管理对象，它可以通过从根节点开始的一条唯一路径来识别，这也就是对象标识符OID（Object Identifier）。

图1-2  MIB树结构示意图   


OID是由一些系列非负整数组成，用于唯一标识管理对象在MIB树中的位置。由SMI来保证OID不会冲突。

MIB文件一旦发布，OID就和被定义的对象绑定，不能修改。MIB节点不能被删除，只能将它的状态置为“obsolete”，表明该节点已经被废除。

在[图1-2](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "fig_dc_ar_mib_operation_000501)的树形结构中，mgmt对象可以标识为：{ iso(1) org(3) dod(6) internet(1) mgmt(2) }，简单标记为：1.3.6.1.2，这种标识就叫做OID。

NMS通过OID引用Agent中的对象。

MIB对象及节点

如[图1-3](http://support.huawei.com/enterprise/docinforeader!loadDocument1.action?contentId=DOC1000097258&partNo=10042" \l "fig_dc_vrp_mib_overview_0012)所示，以IF-MIB为例。MIB树中有表对象、行对象与列对象。表对象由行对象构成，行对象由一系列的列对象构成。

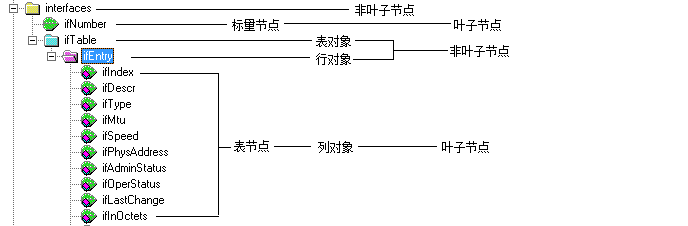
MIB树的节点可以分为以下两种。

叶子节点

叶子节点即在完整的MIB树中不存在子节点的节点。叶子节点又分为标量节点和表节点，访问标量节点时，在该节点OID后加上.0，访问表节点时，在该节点OID后加上.表的索引值。例如访问标量节点sysDescr（1.3.6.1.2.1.1.1）时加上.0即1.3.6.1.2.1.1.1.0，访问表节点ifDescr（1.3.6.1.2.1.2.2.1.2）时加上.表的索引值3即1.3.6.1.2.1.2.2.1.2.3。

非叶子节点

非叶子节点用来表明该节点的子节点的相关性，不能通过SNMP协议直接访问。

图1-3  MIB节点及对象   


IMG_259 说明：

在具体的MIB参考中，经常将行对象“Entry”描述为前缀。例如，在ifTable中描述为“该表的OID前缀为1.3.6.1.2.1.2.2”对应ifEntry。

1.2.2  MIB文件介绍

定义

MIB文件是指使用抽象语法标记（Abstract Syntax Notation One）ASN.1来描述MIB中各管理对象的文本文件，它通过ASN.1语法的有关文档如RFC1155、RFC1212等精确定义MIB中各管理对象。

目的

在网管侧使用网管软件导入MIB文件和设备进行标准对接后，通过对MIB节点的操作，实现网络管理。

MIB文件特点

MIB文件以DEFINITIONS ::= BEGIN开始，最后一行以END结束。DEFINITIONS ::= BEGIN前面是库名称，库名称就像是一个系统中的对象名，是唯一的。

可以用记事本、写字板等一些编辑器来打开或编写MIB文件，一般文件的后缀名都用“.mib”。

1.3  MIB访问流程

简要介绍了访问MIB的整个操作流程。

MIB访问流程

访问MIB节点前，需要满足：

完成网管侧和设备侧的所有参数配置。

设备侧的参数配置请参见[3 通过MIB管理前配置](http://support.huawei.com/enterprise/javascript:window.parent.showDocSection('DOC1000097258','10062','dc_ar_mib_operation_0060');)；网管侧的配置请参考网管的操作手册。

在MIB Browser中编译MIB文件并且加载到MIB文件列表中。

IMG_260 说明：

本手册中以MG-SOFT MIB Browser作为网管软件进行操作演示。若使用其他的网管软件，请参考具体软件的使用说明。

MG-SOFT MIB Browser是一个开源软件，用户可到http://www.mg-soft.com/上获取相关文档。

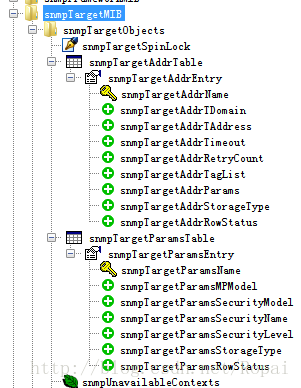
当满足上述条件后，用户可以在MIB Browser中对已经导入的MIB节点进行操作。操作时，Agent通过UDP端口161接收来自NMS的Request报文，经过基本处理过程后返回Response报文，NMS得到响应报文后，经过同样的处理，最终显示对应操作的返回值。当被管理设备状态发生变化如链路失效、负载超过门限等，Agent也可以主动发出Trap报文及时上报异常事件。

# SNMP MIB中的含read-create节点的表的实现

2013年10月16日 16:55:38 [非著名码农](https://me.csdn.net/Ropai" \t "https://blog.csdn.net/ropai/article/details/_blank) 阅读数：8057

     做过snmp/mib开发的知道，常见的节点类型一般只有no-accessible,read-only,read-write三种访问类型。snmp V2中引入了一种新的访问类型:read-create。

     最近在一个产品的snmp管理接口开发中，需要实现snmpTargetMIB(.1.3.6.1.6.3.12)，其下面两个表snmpTargetAddrEntry和snmpTargetParamsTable的节点都是read-create类型。



    read没什么好说的，关键是这个create。Create是指这个表的行可以通过“snmp途径”来操作（增加，删除）。这些行操作是通过表中的“RowStatus”节点来进行的，比如上图中的snmpTargetAddrRowStatus和snmpTargetParamsRowStatus。mib中这两个节点的类型定义如下：

****ROWSTATUS INTEGER {active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6) }****

****1.表中行的创建/删除操作****

       如果把rowstauts节点值设成（SnmpSet）4,5则表示在表中建立一个新的行（索引值同rowstatus节点）。

        而1,2,3则是用来标记这一行描述的对象的状态的（前提是已经建立）。即建立一行后，应根据对象的=状态来更新rowstatus节点值到1/2/3。

        如果把RowStatus的值设成6，则表示要删除这行。

        可能有点绕，总结一下，有管理软件（发起snmp命令的）和被管理方（MIB所在处），表的行的节点建立&删除指令由管理软件来发送，由被管理方来执行。

        举个例子：

             比如表中已经有两条记录：

             snmpTargetAdrrRowStatus(1) --->Active

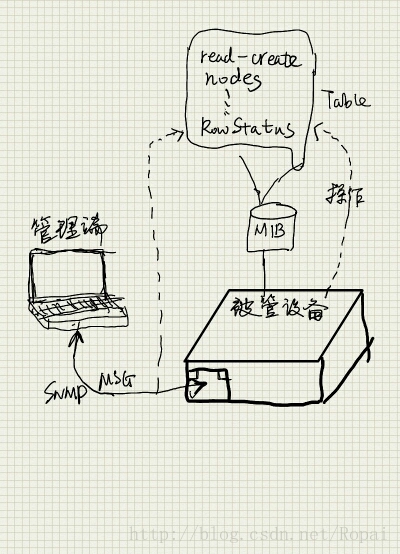
             snmpTargetAddrRowStatus(2) ---->notReady

             要增加一个索引为3的记录，管理方发起一条 snmpSet snmpTargetAddrRowStatus(3) --->createAndGo，被管理方收到这条消息后，在表中建立一条索引为3的记录，包括snmpTargetAddrRowStatus(3）节点。

****2.表中行的节点值的维护/更新****

       表行中所有的节点的值，除了rowStatus的4,5,6只能由管理软件来写（snmp-set），其他节点的值和RowStatus的2,3,4可以由管理软件或被管理方来更新。这点mib定义是未做约束的，我觉得这样也是合理的，可以根据需要灵活处理。

         比如要添加一个告警目标（IP地址），这个地址通过管理软件来设置（snmp-set snmpTargetAddrTAddress)，也可以在被管理方上来设置（比如通过Console接口，配置文件等）。  而这个目标是否可以连通则由被管理方来检测（根据结果设置rowStatus的值到1/2/3）。其实每个节点在创建后被管理方都应给它个默认值。



     关于read-create节点表的软件实现，提供点思路吧。

     管理方软件：

             其实就是snmp-get 和snmp-set的命令组合，和read-write节点操作一样。“create”是通过snmp-set RowStatus节点来实现的。

     被管理方软件：

             对于RowStatus节点，定义好对RowStatus节点操作的回调函数，回调函数中根据写入的值来创建/删除表中的记录行。对于其他节点的实现则和read-write节点一样。

SNMP介绍, OID及MIB库

2018年04月12日 16:11:01 路痴的旅行 阅读数：3791

1 SNMP概览

1.1 SNMP的基本知识介绍

简单网络管理协议（SNMP－Simple Network Management Protocol)是一个与网络设备交互的简单方法。该规范是由IETF在1990年五月发布的RFC 1157中定义的。SNMP通常被认为相当难懂，并且过于复杂，其可用的API似乎在本来非常简单的东西外面封装了大量的东西。现在关于SNMP的书籍又往往只是把它更加复杂化了，而没有解释清楚。

SNMP对于任何程序设计人员来说是特别易于理解的。总体的简化能够很好地把这个系统简化。一个网络设备以守护进程的方式运行SNMP代理，该守护进程能够响应来自网络的各种请求信息。该SNMP代理提供大量的对象标识符（OID－Object Identifiers）。一个OID是一个唯一的键值对。该代理存放这些值并让它们可用。一个SNMP管理器（客户）可以向代理查询键值对中的特定信息。从程序员的角度看，这和导入大量的全局变量没有多少区别。SNMP的OID是可读或可写的。尽管向一个SNMP设备写入信息的情况非常少，但它是各种管理应用程序用来控制设备的方法（例如针对交换机的可管理GUI）。SNMP中有一个基本的认证框架，能够让管理员发送公共名来对OID读取或写入的认证。绝大多数的设备使用不安全的公共名 "public" 。 SNMP协议通过UDP端口161和162进行通信的。

注意，我还没有提到MIB！MIB的重要性被大大地夸大了。刚开始时，MIB显得非常复杂，但是它们其实非常简单。OID是数字的和全局的键值对。一个OID看起来和一个IPv6的地址很象，并且不同的厂商有不同的前缀等信息。OID都非常长，使得人们难以记住，或者对他非常感冒。因此，人们就设计了一种将数字OID翻译为人们可读的格式。这种翻译映射被保存在一个被称为 “管理信息基础"（Management Infomation Base) 或MIB的、可传递的无格式文本文件里。使用SNMP或者向SNMP设备查询，你不需要使用MIB，但是，如果没有MIB，你就得猜测你正在查看的数据是什么。某些情况下，不使用MIB也非常简单，例如查看主机名、磁盘使用率数字，或者端口状态信息。其他情况下，就非常困难了，这个时候使用MIB就非常有帮助。对于准备编写的应用程序来说，为了让用户避免妥当安装MIB带来的麻烦，而严格使用数字OID很常见。安装一个MIB的动作，只是将他放置到你的SNMP客户端应用软件能够搜索到并进行上述翻译映射工作的某个位置而已。

SNMP可以按照两种方式来使用: 轮询和陷阱。轮询就是说你编写一个应用程序能够设置一个发送给一个SNMP代理查看某些值的SNMP GET请求。这种方法非常有用，因为如果该设备响应了请求，你就得到了你需要的信息，如果该设备没有响应请求，你就能够知道存在某些问题。轮询是网络监控的一种主动形式。另一方面，SNMP陷阱能够被用来进行被动形式的网络监控。SNMP陷阱是通过配置SNMP设备的代理，让他在某些动作发生时联系另一个SNMP代理来实现的

可以配置为在某些事件发生时发送SNMP陷阱。例如，你可以配置Cisco的IOS在某个独立事件（例如链路断开）发生时，或者在任何定义的陷阱事件发生时，发送SNMP陷阱。（IOS：snmp服务器开启了链路断开的snmp陷阱）。当陷阱事件发生时，设备中的snmp代理会发送该陷阱到一个预先配置好的通常成为陷阱主机的目标上。陷阱主机会运行有自己的SNMP代理，该代理能够接受并处理传入的陷阱。这些陷阱的处理由陷阱处理器来完成。陷阱处理器可以用任何语言编写，并且可以通过STDIN（标准输入）传入的来自发送陷阱的信息。该处理器之后可以根据陷阱进行任何想作的事情，例如发送邮件或者你想要的任何事情。

SNMP被广泛应用在NMS网络管理系统中（Network Management System）。知名的NMS包括BMC的Patrol、CA的Unicenter、Sun Mangegement控制台、IBM的Tivoli Netview、以及全球著名的HP Openview。NMS的目标是提供一个监控和管理所有开启SNMP功能的设备的单一入口。通过配置你的设备代理来接受写访问，你可以从一个应用程序中处理你的网络环境。如果你的整个环境围拢NMS解决方案架构你的环境，你就能无限制地控制、查看你的整个网络。尽管Net-SNMP提供了可用来构建你自己的NMS网管系统的所有工具，我们不会再进一步讨论关于NMS的话题。不过请记住，如果你认为你的SNMP设备厂商没有提供SNMP代理方面的详细信息，很可能是因为他们希望你购买他们的NMS网络管理系统，或者购买能够在另一个NMS平台上使用的插件。

1.2 SNMP的三大版本

SNMP的常用版本有三个：SNMPv1、SNMPv2、SNMPv3

SNMPv1是为基于公共管理的初始标准。SNMPv2是SNMPv1框架下衍生出来的，但是没有定义信息，其后修订为SNMPv2c,一个带有于SNMPv1类似信息格式的给予公共管理的版本。SNMPv2添加了几个新的数据类型（Counter32、Counter64、Gauge32、UInteger32、NsapAdress 以及BIT STRING),以及对OID表和OID值的设置的增强。SNMPv3是一个带有新的信息格式、ACL、安全功能和远处SNMP参数配置的、扩展了SNMPv2框架的版本。

SNMP是基于几个其他规范的，包括定义给予SNMP的数据类型的ASN.1 BER(Abstract Syntax Notation 1 Basic Encoding Rules), 以及详细描述有SNMP MIB使用的语法的管理信息结构（SMI）。如果你选择修改MIB的某些东西，你需要学习SMIv2和ASN.1语法，不过其他情况下你只需要在兴趣时看看他，而不必学习他。

现在，SNMPv1和SNMPv2被广泛应用，但是由于这些协议的不安全特性，通常只使用只读访问。通常，除非你确实需要附加安全特性，否则你不需要过多地关注SNMPv3。SNMPv3是具有安全性的通信协议。

Net-SNMP中关于同步和异步应用程序的文档，常常会把没有经验的编码新手给吓唬住。别担心，那只是指无法坐等响应的应用程序。如果你的应用程序需要以非阻塞方式处理SNMP数据流，就使用一步接口（例如GUI、线程、forking等）。否则，只需要使用同步接口就可以了。

2 MIB和OID

OID(对象标识符），是SNMP代理提供的具有唯一标识的键值。MIB（管理信息基）提供数字化OID到可读文本的映射。

2.1 OID

OID的编写规则和习惯

SNMP OID是用一种按照层次化格式组织的、树状结构中的唯一地址来表示的，它与DNS层次相似。与其他格式的寻址方式类型，OID以两种格式加以应用：全名和先对名（有时称为“相关”）

完全验证格式从root根开始，并且向外移到某个设备的独立的质上。例如一个完整验证的地址为:

该OID可用人们可读的方式重写为：

所有完全验证OID都有 .iso.org.dod.internet.private 开始，数字表达为: .1.3.6.4. 。几乎所有的OID都会跟上企业(.1)和由IANA（互联网编号分配中心分配的）唯一的厂商标号。例如OID 789表示Network Appliance格式的厂商编号( NetApp ）。厂商编号后面的是基于厂商实现的功能，并且各不相同。请注意，在iso.前面的 . ，与DNS中的后点相似，正确验证的OID是有一个表示根的前缀 . 开始的。

OID的相对格式，从企业值开始，略过所有的隐含地址。因此，我们可以用相对地址 enterprises.netapp.netappl.raid.diskSUmmary.diskSpaceCount.0 来表示上述的OID，或者用数字格式 .1.789.6.4.8.0 .

写OID的常用格式是用MIB名称和在MIB中定义的唯一键值。例如，我们可以用简写的格式重写上述OID:

NETWORK-APPLIANCE-MIB::diskSpareCount.0

MIB中OID的书写格式规则为：:MIB Name::唯一键值.instance.

某些唯一键值，可用多个实例表示，这样所有的OID都以实例值结尾。这就是为什么你得注意到大多数OID都是以一个 .0 结尾的。

2.2 MIB介绍

MIB的内部结构刚开始时会让人感觉有些奇怪和不好理解，不过它的结构非常好，你可以在不懂的情况下一个一个看进去。MIB的结构来源于IETF RFC1155和2578定义的管理信息结构。如果你想要修改或编写自己的MIB，在动手前理解SMI非常有帮助。

为了更好地理解他们是怎样工作的，我们先来看看MIB的头:

-- PowerNet-MIB {iso org(3) dod(6) internet(1) private(4)

-- enterprises(1) apc(318) }

PowerNet-MIB DEFINITIONS ::= BEGIN

IMPORTS

enterprises, IpAddress, Gauge, TimeTicks FROM RFC1155-SMI

DisplayString FROM RFC1213-MIB

OBJECT-TYPE FROM RFC-1212

TRAP-TYPE FROM RFC-1215;

apc OBJECT IDENTIFIER ::= { enterprises 318 }

products OBJECT IDENTIFIER ::= { apc 1 }

apcmgmt OBJECT IDENTIFIER ::= { apc 2 }

Comments can be inserted into a MIB by prepending them with two dashes. In the header the declaration BEGIN starts off the MIB. Imports can be used to pull information from other MIBs, typically those mandated by the MIB-II standard.

可以用行开头为 -- 的方法在MIB中加入注释。

在头部用 BEGIN`声明来开始MIB的定义。

`Imports 可用来从其他MIB中提取信息,通常用它来提取MIB-II规范要求的内容。

MIB放置从enterprise值开始的OID地址的结构。在此，enterprise值是318, 对应 "apc" (相对地址为 .1.318）。 通常之后会定义几个类别。注意在花括号间定义的两个值,其父地址后面跟一个它自己的地址。因此产品标识符有apc标识符表示，其父为enterprise标识符，以此类推。类别和自类别的类型通常跟在MIB头的后面，并且把有用的键值分割为子组。通过分段，各种值分别被列出，这样可用的值更容易浏览。

MIB的真正好处在于对象类型的描述。以下是一个整形键值的例子:

upsBasicOutputStatus OBJECT-TYPE

SYNTAX INTEGER {

unknown(1),

onLine(2),

onBattery(3),

onSmartBoost(4),

timedSleeping(5),

softwareBypass(6),

off(7),

rebooting(8),

switchedBypass(9),

hardwareFailureBypass(10),

sleepingUntilPowerReturn(11),

onSmartTrim(12)

}

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The current state of the UPS. If the UPS is unable

to determine the state of the UPS this variable is set

to unknown(1)."

::= { upsBasicOutput 1 }

因此，在这里真正需要注意的是，MIB其实只是提供给我们一张我们想从某个SNMP代理中获得的可用OID的各种值的地图。一个MIB描述了在哪里找某个值、以及返回结果是什么。我们可以不用MIB与设备进行交互，只不过在理获得'Up'的返回值，要比‘1’的返回值要简单的多。通过利用Net-SNMP命令行工具，你可以决定你希望返回结果的输出样式（这种方式下使用“Up"这样的格式更好），或者你用脚本调用工具时（这种方式下使用”1“的格式就更好）。

2.3 OID数据类型

SMI定义的OID返回值的数据类型。

SMI定义了一定数量的OID返回的数据类型。这些数据类型包括：

Integer 整型

Signed 32bit Integer (values between -2147483648 and 2147483647). 有符号32位整数（值范围: -2147483648 - +2147483648）

Integer32

Same as Integer. 与Integer相同。

UInteger32

Unsigned 32bit Integer (values between 0 and 4294967295). 无符号32位整数（值范围：0－4294967295）.

Octet String

Arbitrary binary or textual data, typically limited to 255 characters in length. 任意二进制或文本数据，通常长度限制在255个字符内。

Object Identifier

An OID. 一个OID。

Bit String

Represents an enumeration of named bits. This is an unsigned datatype. 表示取名的位的枚举。这是一个无符号的数据类型。

IpAddress

An IP address. 一个IP地址。

Counter32

Represents a non-negative integer which monotonically increases until it reaches a maximum value of 32bits-1 (4294967295 dec), when it wraps around and starts increasing again from zero. 表示一个非负的整数（可递增到32位最大值－1），然后恢复并从0开始递增。

Counter64

Same as Counter32 but has a maximum value of 64bits-1. 与Counter32相同，最大值为64位的最大值－1。

Gauge32

Represents an unsigned integer, which may increase or decrease, but shall never exceed a maximum value. 表示无符号整数，可增加或减少，但是不超过最大值。

TimeTicks

Represents an unsigned integer which represents the time, modulo 2?32 (4294967296 dec), in hundredths of a second between two epochs. 表示代表数据的一个无符号整数，2^32取模（4294967296），两个值之间为百分之一秒。

Opaque

Provided solely for backward-compatibility, its no longer used. 提供向下兼容，不再使用的数据类型

NsapAddress

Represents an OSI address as a variable-length OCTET STRING. 表示一个用变长八进制字符窗表示的OSI地址。

Net-SNMP tools will report the datatype when returning an OID unless you otherwise disregard it. As an example of that you’ll see:

Net-SNMP工具在返回一个OID时会包括其数据类型，除非你不想要他。以下是一个例子：

SNMPv2-MIB::sysContact.0 = STRING: Ben Rockwood

IF-MIB::ifPhysAddress.1 = STRING: 0:c0:b7:63:ca:4c

SNMPv2-MIB::sysUpTime.0 = Timeticks: (47372422) 5 days, 11:35:24.22

IF-MIB::ifAdminStatus.1 = INTEGER: up(1)

SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.318.1.3.7

RFC1213-MIB::atPhysAddress.1.1.10.10.1.1 = Hex-STRING: 00 50 73 28 47 A0

RFC1213-MIB::atNetAddress.1.1.10.10.1.1 = Network Address: 0A:0A:01:01

IF-MIB::ifSpeed.1 = Gauge32: 10000000

SNMPv2-MIB::snmpInPkts.0 = Counter32: 316

SNMPv2-MIB::snmpOutPkts.0 = Counter32: 314

这是一个相当典型的由Net-SNMP工具返回的数据类型表单。注意有些值已经由Net-SNMP自动解释了，比如sysUpTime和ifAdminStatus。这些值返回时使用到MIB，Net-SNMP会在MIBzhong找到返回值，并且为我们提供该值的文本表示。

3 实例

(1) MIB 结构

TEST-TRAP-MIB DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, TimeTicks FROM SNMPv2-SMI

DisplayString, FROM SNMPv2-TC

enterprises

FROM RFC1155-SMI;

test OBJECT IDENTIFIER ::= { enterprises 1000 }

TestTraps OBJECT IDENTIFIER ::= { test 1 }

cpuRatioHigh NOTIFICATION-TYPE

OBJECTS {SystemTrapDescription}

STATUS current

DESCRIPTION

"."

::= { TestTraps 1 }

TestDescription OBJECT IDENTIFIER ::= { TestTraps 2 }

TestTrapDescription OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..256))

MAX-ACCESS read-only

STATUS current

DESCRIPTION " "

::= { TestDescription 1 }

END

(2) 对应OID:

const oid cpuRatioHigh\_oid[] = { 1,3,6,1,4,1,1000,1,1 };

const oid TestTrapDescription\_oid[] = { 1,3,6,1,4,1,1000,1,2,1, 0 };

某些唯一键值，可用多个实例表示，这样所有的OID都以实例值结尾。这就是为什么你得注意到大多数OID都是以一个 .0 结尾的。