



第3章 管理信息库

- 管理信息结构
- MIB树结构
- MIB-II



管理信息库概述

管理信息库（**Management Information Base, MIB**）是一个概念上的数据库，定义了一个网络中所有可能的被管理对象的集合的数据结构，指明了网络元素所维持的变量（即能够被管理进程查询和设置的信息）。

MIB的总体框架、数据类型的表示方法和命名方法是由**SMI**定义和说明的。



管理信息结构

1. 管理信息结构的概念

管理信息结构（Structure of Management Information, SMI）是简单网络管理协议(SNMP)的一部分。

SMI定义了一个ASN.1的子集，规定了SNMP使用到哪些ASN.1符号与元素，通过使用这些ASN.1子集的符号和元素描述SNMP。

SMI被划分为三个部分：模块定义，对象定义和陷阱定义。



什么是 ASN.1

- ASN.1(Abstract Syntax Notation One 抽象语法计法1):

一种**独立于机器的描述语言**，用于描述在网络上传递的消息，它定义了一组用来描述OSI网络上所传输的数据结构规则。

- 抽象语法:

一种高层的数据类型定义语言，允许使用者以独立于物理传输的方法定义协议标准中的数据类型。



为什么使用**ASN.1**

网络管理必须基于一个异构的网络环境，这样就必需有一个标准，使得S N M P消息可以在不同的系统和设备间传送并被它们所理解。



抽象语法标记1 (ASN.1)

- 是ISO和ITU-T的联合标准。
- 描述了一种对数据进行表示、编码、传输和解码的数据格式。它提供了一整套正规的格式用于描述对象的结构，而不管语言上如何执行及这些数据的具体指代。在任何需要以数字方式发送信息的地方，ASN.1 都可以发送各种形式的信息（音频、视频、数据等等）。它与特定的ASN.1编码规则一起通过使用独立于计算机架构和编程语言的方法来描述数据结构，为结构化数据特别是在网络环境的应用程序的交互提供了手段。



抽象语法标记1 (ASN.1)

ASN.1 可分为两个部分

- (1) 语法规则：从数据类型、内容顺序或结构等方面来描述消息的内容
 - (2) 编码规则：如何编码实际消息中的数据(基本编码规则BER)
- ASN.1中包含了一些预定义的通用类型，也规定了通过现有类型定义新类型的语法。



ASN.1语法

在ASN.1中为每个应用所需的所有数据结构类型进行定义。

ASN.1文本的书写规则(文本约定 Lexical Convention)：

- (1) 多个空格和空行等效于一个空格。
- (2) 用于表示值和字段的标识符、类型指针和模块名由大小写字母、数字和短线组成。
- (3) 标识符以小写字母开头。
- (4) 类型指针和模块名以大写字母开头。
- (5) ASN.1定义的内部类型全部用大写字母表示。
- (6) 关键字全部用大写字母表示。
- (7) 注释以一对短线 (--) 开始，以一对短线或行尾结束。



ASN.1的约定符号

符号	含义
::=	定义为，或赋值
	或、选择、列表选项
-	标记号
--	符号后跟随注释
{ }	列表的开始和结束
[]	标记的开始和结束
()	子类的开始和结束
..	范围



常用的ASN.1关键字

关键字	主要描述
BEGIN	ASN.1模块的开始
CHOICE	替代清单
DEFINITIONS	数据类型或管理对象的定义
END	ASN.1模块的结束
EXPORTS	可以输出数据类型到其他模块中
IDENTIFIER	非负数的排列
IMPORTS	在外部模块中定义的数据类型
INTEGER	任何整数，正整数或负整数或零
NULL	占位符
OBJECT	与IDENTIFIER一起使用，以便惟一识别一个对象
OCTET	无限的二进制数据8位字节
STRING	与OCTET一起使用，以便指明八位位组字符串
SEQUENCE	制作出排列好的列表
SEQUENCE OF	排列好的重复数据数组
SET	制成无序的列表
SET OF	重复数据的无序列表



ASN.1的数据类型

数据类型由标签的类型和值唯一确定：

- 通用标签：用关键字**UNIVERSAL**表示，带有这种标签的数据类型是由标准定义的，适合于任何应用。
- 应用标签：用关键字**APPLICATION**表示，是有某种具体应用定义的类型。
- 上下文专用标签：用关键字**CONTEXT_SPECIFIC**表示，这种标签在文本的一定范围（如1个结构）中适用。
- 私有标签：用关键字**PRIVATE**表示，这是用户定义的标签



通用数据类型

标签	类型	值集合
UNIVERSAL 0	Reserved for use by the encoding rules	保留给编码规则使用
UNIVERSAL 1	Boolean	TRUE, FALSE
UNIVERSAL 2	Integer	正数、负数和零
UNIVERSAL 3	Bitstring	0个或多个比特组成的序列
UNIVERSAL 4	Octetstring	0个或多个字节组成的序列
UNIVERSAL 5	Null	空类型
UNIVERSAL 6	Object identifier	对象标识符
UNIVERSAL 7	Object descriptor	对象描述符



通用数据类型

标签	类型	值集合
UNIVERSAL 8	External type and Instance-of type	外部文件定义的类型
UNIVERSAL 9	Real	所有实数
UNIVERSAL 10	Enumerated	枚举类型(整数值 的表，每个整数 有一个名字)
UNIVERSAL 11	Embedded-pdv	嵌入的 pdv 类型
UNIVERSAL 12	UTF8String	UTF8 字符串类型
UNIVERSAL 13	Relative object identifier	相关对象标识符 类型
UNIVERSAL 14- 15	保留	保留给本建议的 以后版本和国际 标准使用



通用数据类型

标签	类型	值集合
UNIVERSAL 16	Sequence Sequence-of	序列和类型序列
UNIVERSAL 17	Set , Set-of	集合和类型的集合
UNIVERSAL 18	Numeric String	数字0~9 和空格
UNIVERSAL 19	Printable String	可打印字符串，包括所有大小写字母、数字、标点符号和空格
UNIVERSAL 20	Teletex String	由原CCITT T.61 建议定义的新字符集
... ..		



数据类型

根据组成结构又可以分为4大类。

- 简单类型：由单一成分构成的原子类型。包括BOOLEAN、INTEGER、BIT STRING、OCTET STRING、NULL、OBJECT IDENTIFIER、OBJECT DESCRIPTOR、REAL、ENUMERATED、CHARACTER STRING等
- 构造类型：由两种以上成分构成的构造类型，包括SEQUENCE、SEQUENCE OF、SET、SET OF。
- 标签类型：由已知的类型定义的新类型。
- 其他类型：包括CHOICE和ANY两种类型。



ASN.1 类型定义

- 语法: `<type name> ::= <type>`

- 示例:

```
Counter ::= INTEGER
```

```
IpAddress ::= OCTET STRING
```

```
Months ::= ENUMERATED {  
    january (1), february (2), march (3),  
    april (4), may (5), june (6),  
    july (7), august (8), september (9),  
    october (10), november (11), december (12)  
}
```




1. 简单类型

(1) INTEGER, 整数类型, ASN.1中没有有限制整数的位数, 可以是任意大小的整数。

例2.1 Number : : =INTEGER

(2) BOOLEAN, 布尔类型, 取值为TRUE (真) 或 FALSE (假)。

例2.2 Married : : =BOOLEAN

(3) REAL, 实数类型, ASN.1中对实数的精度没有限制, REAL可以表示所有的实数。实数也可以表示为科学计数法: $M \times B^E$, 其中尾数M和指数E可以取任何正或负整数值, 基数B可以取2或10。



1. 简单类型

(4) **ENUMERATED**, 枚举类型, 实际上是一组个数有限的整数值。可以给每个整型值赋予不同的意义。

例 2-3 `Week : : =ENUMERATED { Monday (1), Tuesday (2), Wednesday (3), Thursday (4), Friday (5), Saturday (6), Sunday (7) }`



1. 简单类型

(5) **BIT STRING**, 位串类型, 由0个或多个比特组成的有序位串。位串的值可以由对应的二进制或十六进制串表示。

例如, 10100010B或A2H都是位串类型的有效数值。

(6) **OCTET STRING**, 八位位组串, 由0个或多个8位位组组成的有序串。和位串类型一样, 八位位组串也可以用对应的二进制或十六进制串表示。每个字符可从0 ~ 255取值

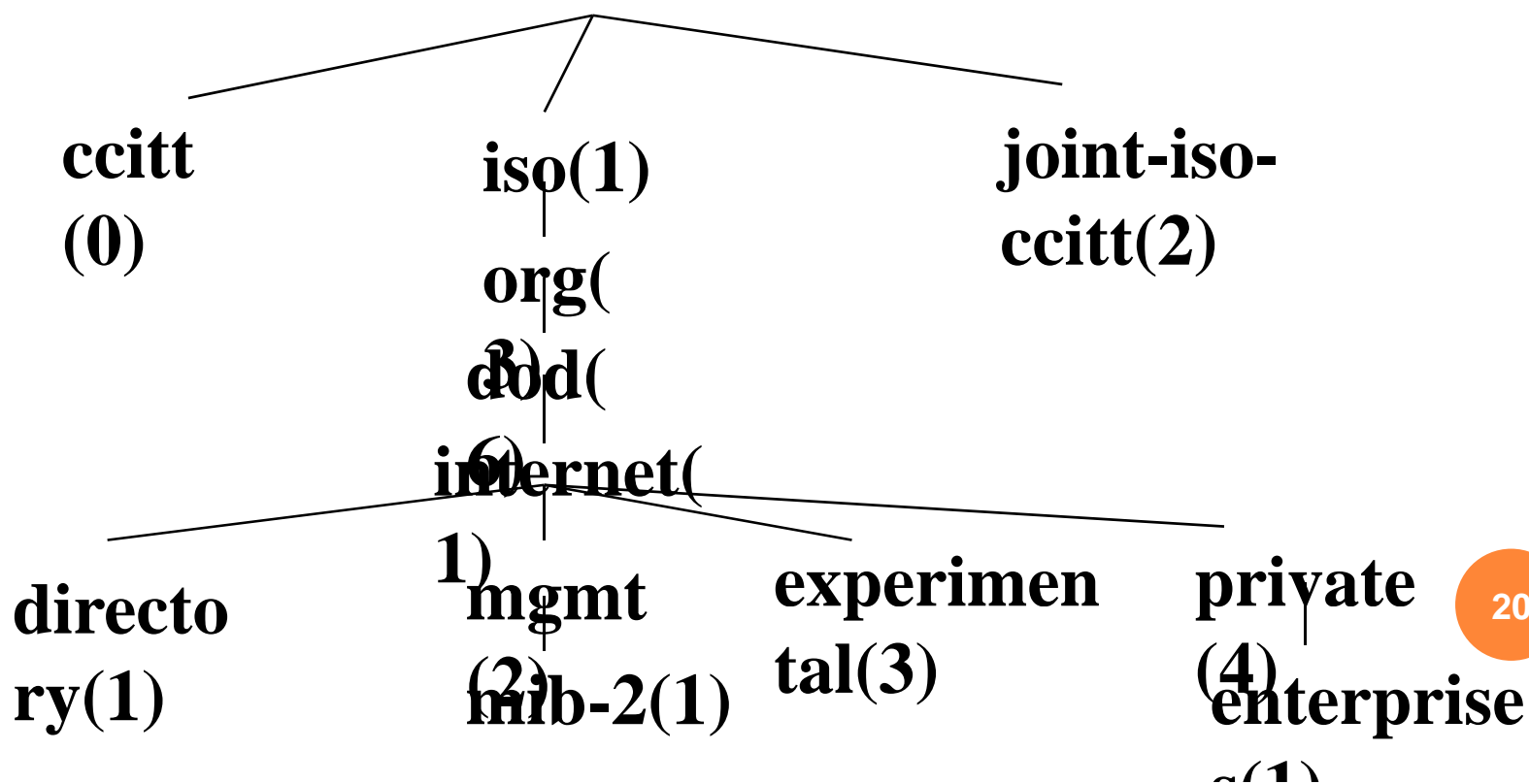
例: `MacAddress::=OCTET STRING(SIZE(6))`



1. 简单类型

(7) OBJECT IDENTIFIER, 对象标识符, 从对象树派生出的一系列点分数字串的形式, 用来标识对象。

internet OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }





1. 简单类型

(8) NULL, 空值类型, 它仅包含一个值—NULL, 主要用于位置的填充。如果某个时刻无法得知数据的准确值, 简单的方法就是将这一数据定义为NULL类型。数据值为NULL时, 表示该值还不知道。还可以用NULL表示序列中可能缺省的某个元素。

(9) CHARACTER STRING, 字符串类型。



2. 构造类型

(1) SEQUENCE, 序列类型, 是包含0个或多个组成元素的有序列表。列表的不同元素可以分属于不同的数据类型。每个元素由元素名和元素类型组成, 元素类型可以是简单类型, 也可以是定义的其他构造类型。与程序设计语言中的结构类似。



例子

- 类型定义

```
UserAccount ::= SEQUENCE {  
    username PrintableString,  
    password PrintableString,  
    accountNr INTEGER  
}
```

- 赋值

```
myAccount UserAccount ::= {  
    username "tly",  
    password "guesswhat",  
    accountNr 2345  
}
```



AirlineFlight ::=SEQUENCE {

实例

airplane1

**AirlineFlight ::= { airline
"china", flight "C3416",
seats {320 , 280, 40},
airport { original
"Qingdao", stop[0]
"TaiYuan",
destination "WuLuMuQi"},
crewsizes 10 }**

**或 airplane1 ::= {"china",
"C3416", {320 , 280, 40},
{ original "Qingdao",
stop[0] "TaiYuan",
destination "WuLuMuQi"},
10 }**

airline IA5STRING,

flight IA5STRING,

seats SEQUENCE {

maximum INTEGER,

occupied INTEGER,

vacant INTEGER },

airport SEQUENCE {

origin IA5STRING,

stop[0] IA5STRING OPTIONAL,

stop[1] IA5STRING OPTIONAL,

destination IA5STRING },

crewsizes ENUMERATED {

six (6),

eight (8),

ten (10)},

cancel BOOLEAN DEFAULT FALSE }



2. 构造类型

(2) SEQUENCE OF, 单纯序列（数组）类型，即序列中的各项都属于同一类型。用于表格，与程序设计语言中的数组类似。

例 Seats ::= SEQUENCE OF INTEGER

例 AirlineFlights ::= SEQUENCE OF AirlineFlight



2. 构造类型

(3) SET , 集合类型

- 类似于**SEQUENCE**, 但不考虑分量顺序
- 类型定义

```
UserAccount ::= SET {  
    username [0] PrintableString,  
    password [1] PrintableString,  
    accountNr [2] INTEGER }
```

- 赋值

```
myAccount UserAccount ::= {  
    accountNr 2345,  
    username "tly",  
    password "guesswhat" }
```



2. 构造类型

(4) SET OF, 单纯集合类型, 类似于SEQUENCE OF

例3-12 Seats : : =SET OF INTEGER

vipseats Seats : : = {10, 20, 30}



3. 标签类型

标签类型由一个标签类（class）和一个标签号（class number）组成.

加标签后的类型实质上是一个新的类型，它和原来的类型在结构上是一样的，但是是不同的类型。举例如下。

例 Number : : =[UNIVERSAL 2]INTEGER
 valA Number: : =200603



4. 其他类型

CHOICE和ANY是两个没有标签的类型，因为它们的值是未定的，而且类型也是未定的。当这种类型的变量被赋值时，它们的类型和标签才能确定。

(1) CHOICE，选择类型，包含一个可供选择的数据类型列表。CHOICE类型的每个值都是其中某一数据类型的值。数据可能在不同情况下取不同的类型，这些可能的类型能够在事先都知道。

例 Prize : : = CHOICE{ car IA5STRING, cash
INTEGER, nothing BOOLEAN }



4. 其他类型

(2) ANY, 和选择类型具有确定的数据类型选择范围不同, 若在定义数据时不能确定数据的类型, 可以使用ANY类型。

例 Book: : =SEQUENCE{ author IA5STRING,
reference ANY }

Book的正确实例:

{author "Martin" , reference IA5STRING "ISBN007895"}

{author "Martin", reference INTEGER 1998}



子类型

通过对某些类型加以限制，可以定义子类型

- ◆ 单个值（Single Value）
- ◆ 包含子类型（Contained Subtype）
- ◆ 值区间（Value Range）
- ◆ 允许字符（Permitted Alphabet）
- ◆ 限制大小（Size Constraint）
- ◆ 内部子类型（Inner Subtyping）



ASN.1子类型定义

- 语法:

`<subtype name> ::= <type> (<constraint>)`

- 示例:

`Counter ::= INTEGER (0..65536)`

`IpAddress ::= OCTET STRING (SIZE(4))`

`Spring ::= Months (march | april | may)`

`Summer ::= Months (june | july | august)`

`SmallPrime ::= INTEGER (2 | 3 | 5 | 7 | 11)`

`ExportKey ::= BIT STRING (SIZE(40))`



ASN.1 子类型

1. 单个值 (Single Value)

列出子类型可取的各个值。例如，

例 `TestResult: : =INTEGER (0|1|2)`

2. 值区间 (Value Range)

这种方法只能用于整数和实数，指出子类型可取的区间。

例 `EmployeeNumber: : =INTEGER (1000..20000)`



ASN.1 子类型

3. 允许字符 (Permitted Alphabet)

允许字符只能用于字符串类型，限制字符集的取值范围。

例: `House Size ::= IA5STRING (FROM ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9") SIZE (5))`

4. 限制大小 (Size Constrained)

可以限制5种类型 (BIT STRING, OCTET STRING, CHARACTER STRING, SEQUENCE OF, SET OF) 的规模大小。

例:

`WorkstationNumber ::= OCTET STRING (SIZE (32))`



ASN.1 子类型

5. 包含子类型 (Contained Subtype)

从已有的子类型定义新的子类型，新子类型包含原子类型的全部可能的值。用关键字**INCLUDES**。例如，

Months: : =ENUMERATED{January (1), February (2),
March (3) , April (4) , May (5) , June (6) , July
(6) , August (8) , September (9) , October (10) ,
November (11) , December (12) }

First-quarter: : =Months (January, February, March)

Second-quarter: : =Months (April, May, June)

First-half: : =Months (**INCLUDES** First-quarter ,
INCLUDES Second-quarter)



应用类型

RFC1155定义了一些应用类型:

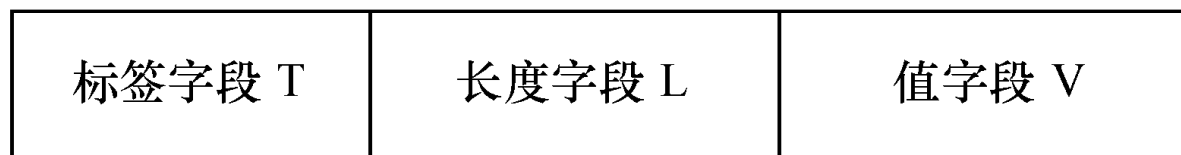
1. NetworkAddress::=CHOICE{internet IpAddress}
2. IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING(SIZE(4))
3. Counter::=[APPLICATION 1] IMPLICIT INTEGER(0..4294967295)
4. Gauge::=[APPLICATION 2] IMPLICIT INTEGER(0..4294967295)
5. TimeTicks::=[APPLICATION 3] IMPLICIT INTEGER(0..4294967295), 以0.01秒为单位
6. Opaque ::= [APPLICATION 4] OCTET STRING



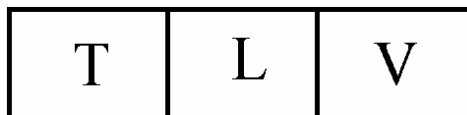
ASN.1基本编码规则

描述了如何将ASN.1类型的值编码成字节串(string of octets)的方法。

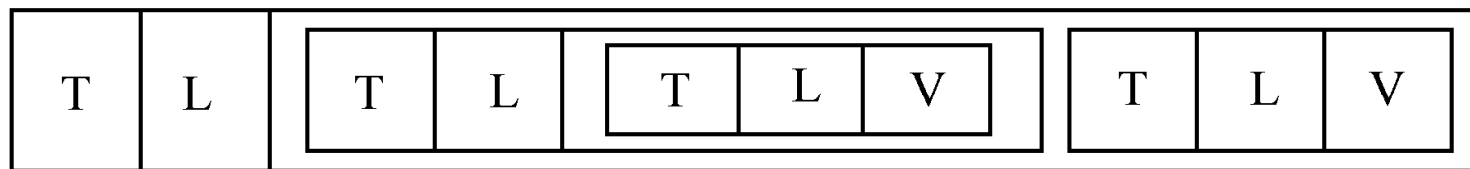
BER传输语法的格式是TLV三元组<标签Tag, 长度 Length, 值 Value>。TLV每个域都是一系列八位位组, 对于构造结构, 其中V还可以是TLV三元组。



简单数据



构造数据



BER编码的结构



ASN.1类型标记

UNIVERSAL CLASS TAGS

UNIVERSAL 1	BOOLEAN	UNIVERSAL 2	INTEGER
UNIVERSAL 3	BIT STRING	UNIVERSAL 4	OCTET STRING
UNIVERSAL 9	REAL	UNIVERSAL 10	ENUMERATED
UNIVERSAL 6	OBJECT IDENTIFIER		
UNIVERSAL 7	ObjectDescriptor		
UNIVERSAL 26	VisibleString	...	
UNIVERSAL 5	NULL		
UNIVERSAL 23	UTCTime		
UNIVERSAL 24	GeneralizedTime		
UNIVERSAL 16	SEQUENCE [OF]	UNIVERSAL 17	SET [OF]



BER中的TAG字段 (1)

TAG NUMBER < 31

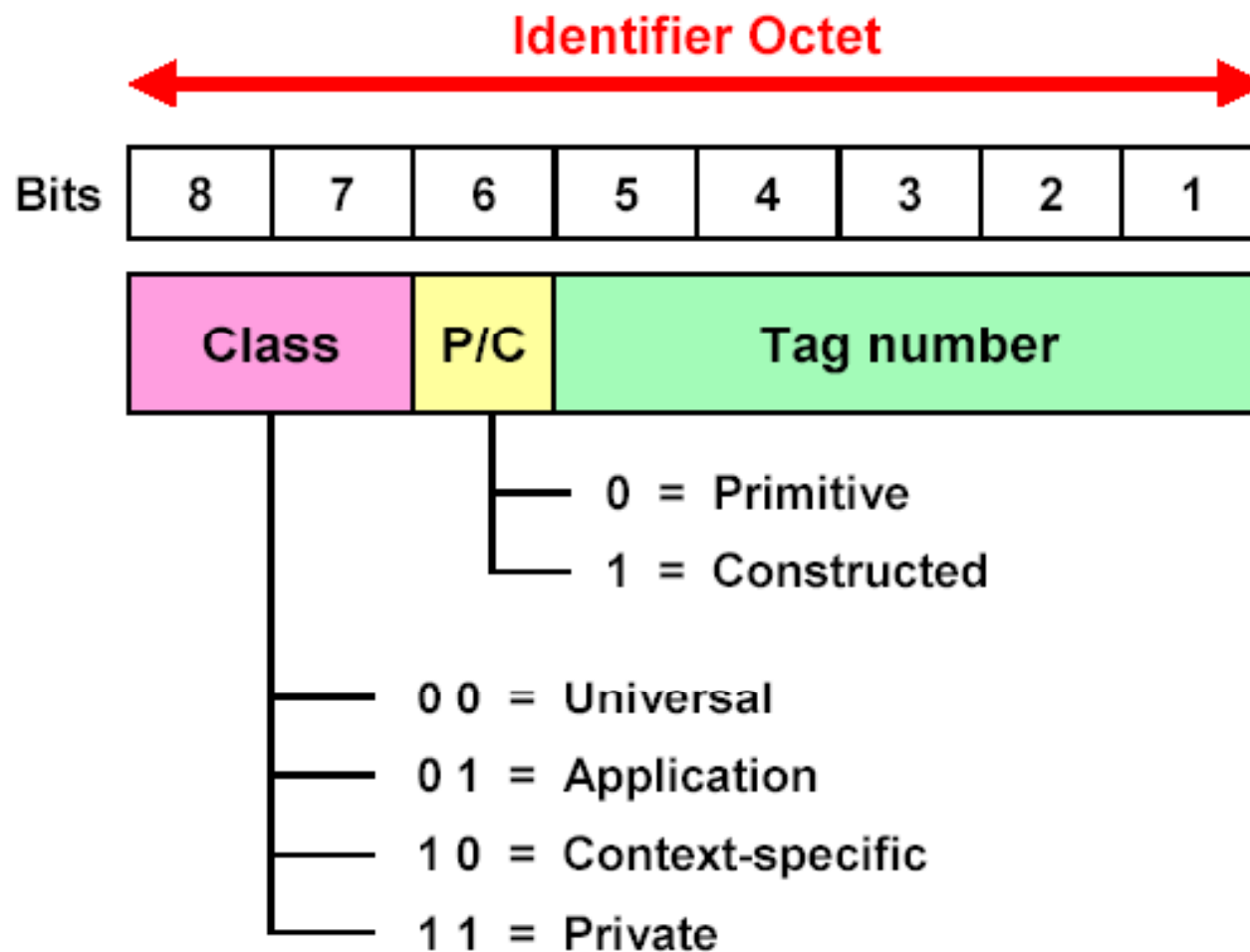




Diagram illustrating the structure of an IPv4 address, showing the leading octet, 2nd octet, and last octet, and how they are combined to form the Tag number.

The address is divided into four octets:

- Leading octet:** Contains Class (pink), P/C (yellow), and 1 1 1 1 1 (blue).
- 2nd octet:** Contains 1 (blue) and a green box.
- ...** (Intermediate octets)
- Last octet:** Contains 0 (blue) and a green box.

The green boxes represent the Tag number, which is the sum of the values in the 2nd, 3rd, and 4th octets:

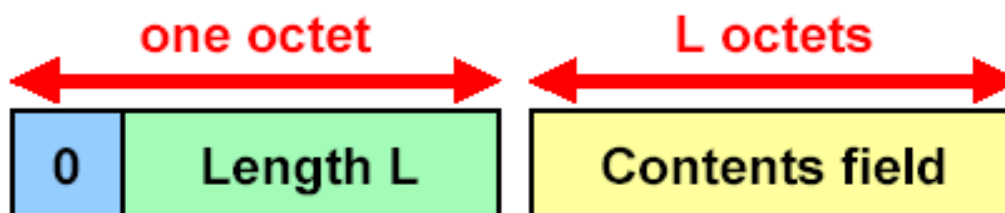
$$\text{Green Box} + \dots + \text{Green Box} + \text{Green Box} = \text{Tag number}$$



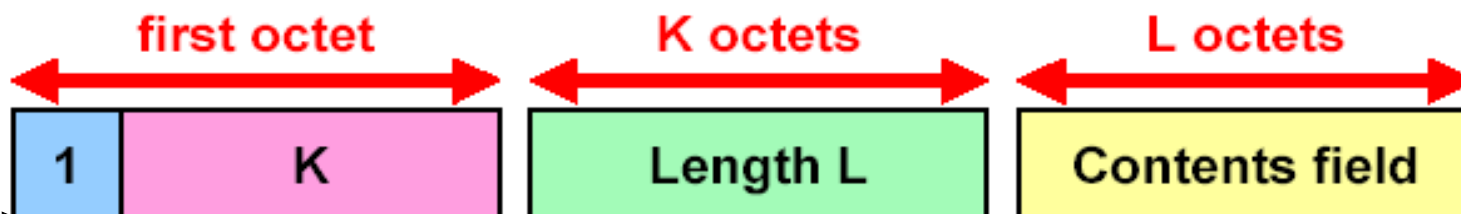
BER中的LENGTH字段(1)

定长格式

- Short definite form ($L < 128$ octets)

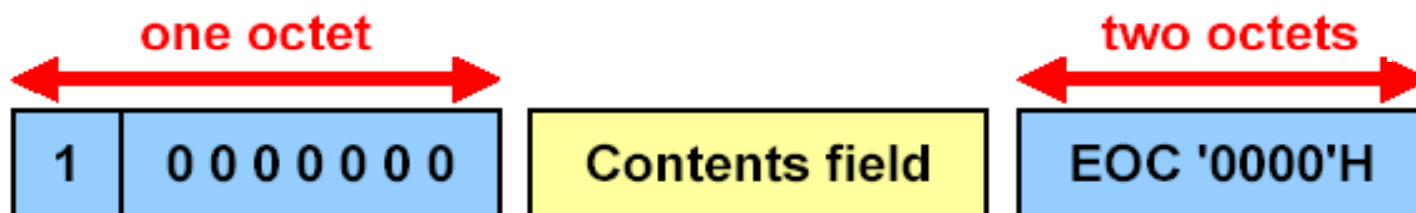


- Long definite form ($128 \leq L < 2^{1008}$ octets)



不定长格式

- Indefinite form; content field terminated by EOC





BER中的LENGTH字段(2)

- 短格式
 - 既可用于基本类型，也可用于内容长度不超过128的构造类型
- 长格式
 - 既可用于基本类型，也可用于构造类型
 - 通常内容长度大于或等于128
- 不定长格式
 - 仅用于构造类型



BER中的VALUE字段(内容字段)

内容字段由0个或多个八位位组组成，并根据不同类型数据值的不同规定对它们进行编码。

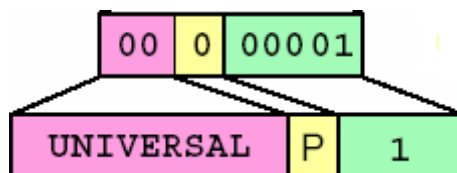


BER编码示例 – BOOLEAN

FALSE

TRUE

标签字段	长度字段	值字段
00000001	00000001	00000000
00000001	00000001	11111111

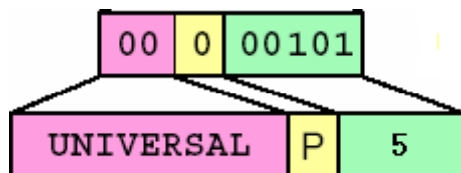




BER编码示例 -- NULL

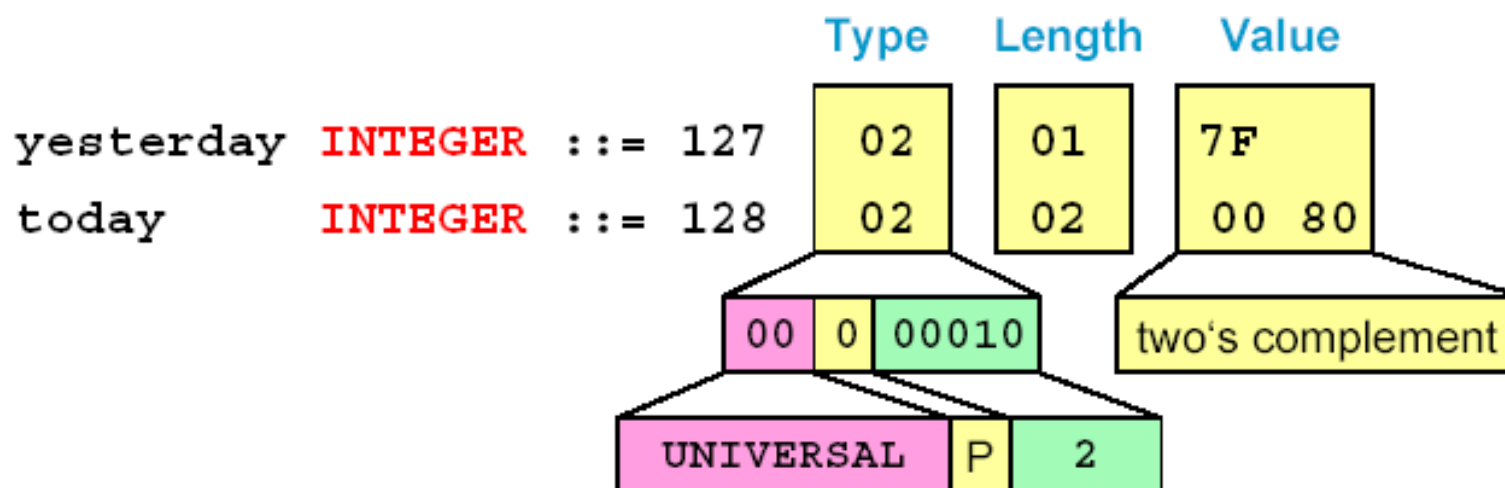
NULL的BER编码: 0500

标签字段	长度字段
00000101	00000000

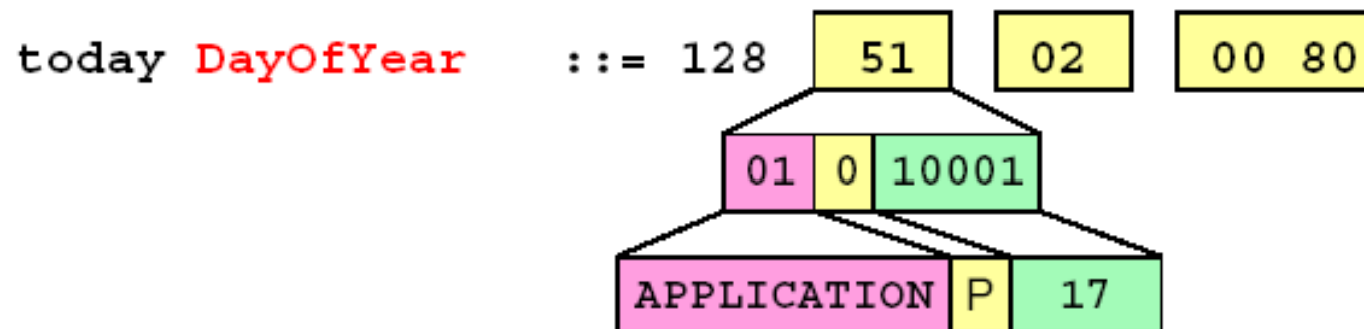




BER编码示例 -- INTEGER



DayOfYear ::= [APPLICATION 17] IMPLICIT INTEGER





BER编码示例 -- INTEGER

- BER coding of two's complement integers

- -129: 1111 1111 0111 1111 = **02 02 FF 7F**
- -128: 1111 1111 1000 0000 = **02 01 80**
- -127: 1111 1111 1000 0001 = **02 01 81**
- -1: 1111 1111 1111 1111 = **02 01 FF**
- 0: 0000 0000 0000 0000 = **02 00**
- 1: 0000 0000 0111 1111 = **02 01 01**
- 127: 0000 0000 0111 1111 = **02 01 7F**
- 128: 0000 0000 1000 0000 = **02 02 00 80**
- 129: 0000 0000 1000 0001 = **02 02 00 81**

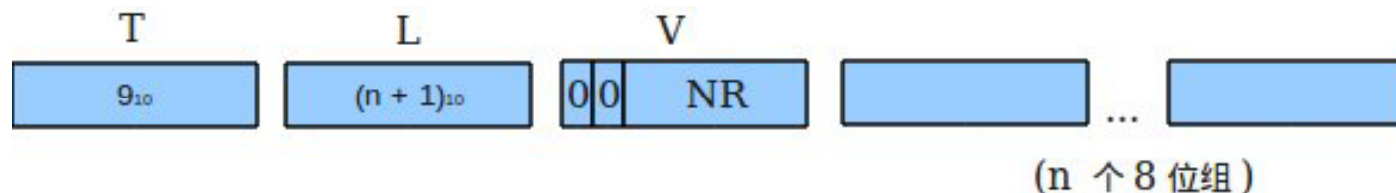


BER编码示例 – REAL

- 实数0

标签字段	长度字段
00001001	00000000

- 基于10进制且以10为底的指数编码方法



其中NR有三个可选值:

- NR1: 在V中低6位用000001表示,表示不带小数和指数的简单10进制整数。例如4902, 每个字符占一个8位组。
- NR2: 在V中低6位用000010表示, 表示含小数点的10进制数。例如4902.00
- NR3: 在V中低6位用000011表示, 在NR2的基础上扩展, 用字符‘E’代表以10为底的指数。 例如 +0.56E2, 0.2E-3



BER编码示例 -- BIT STRING

BIT STRING 可以是简单编码或构造编码

简单编码:值字段包含一个初始八位位组，后面跟0个、1个或多个后继八位位组。初始八位位组的编码是最后后继八位位组中未使用的位数。

例如，位串值（0A3B5F291CD），采用简单编码则为：
03 07 040A3B5F291CD0。

标签字段	长度字段	值字段
03H	07H	040A3B5F291CD0H





- ◆ 构造编码:值字段由0个、1个或多个数据值的完整编码组成。每个这样的编码都包括标识、长度和值字段.采用简单编码。

位串值（0A3B5F291CD），将位串值拆分为（0A3B5F291CD）两部分

最后后继八位位组中未使用的位数为0。

采用构造编码则为：23 80 03 03 000A3B

03 05 045F291CD0

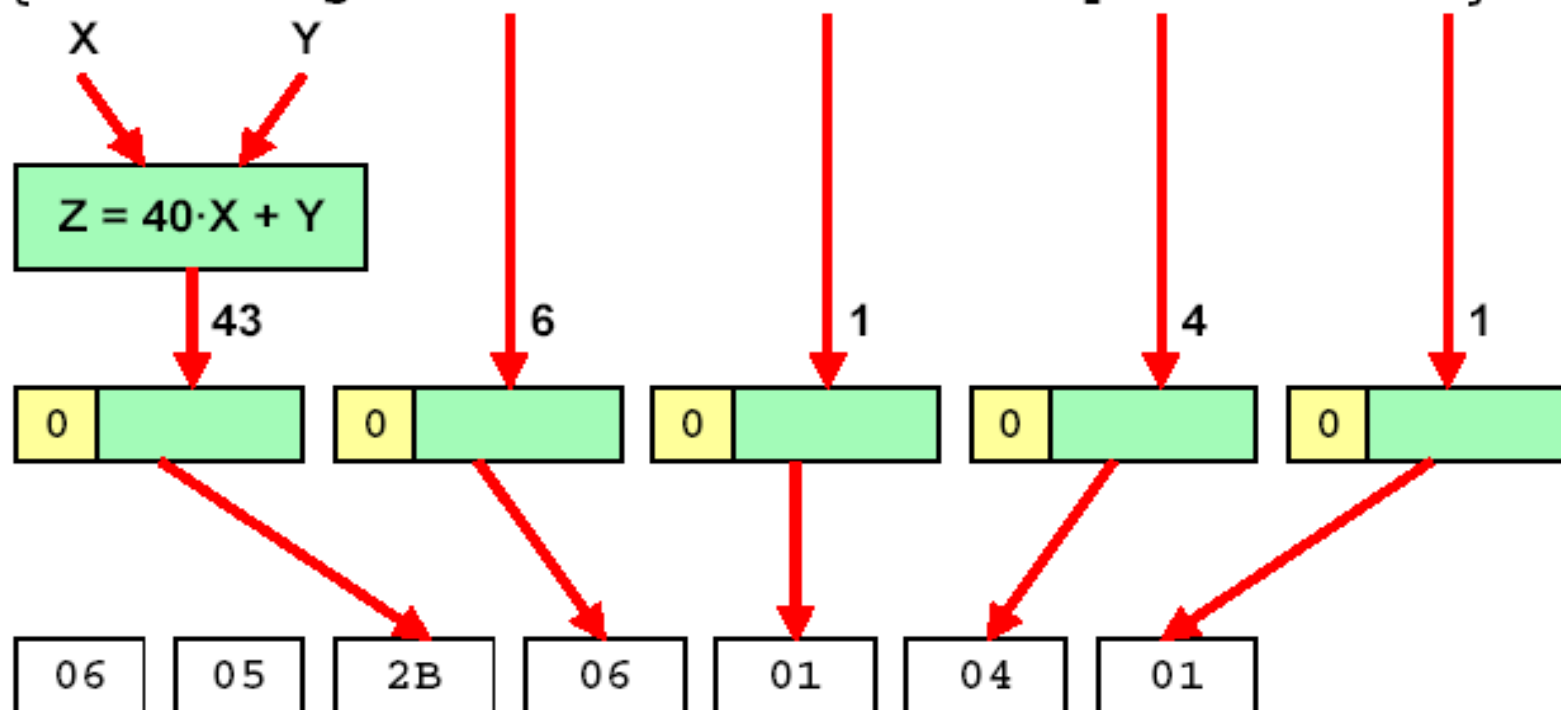
标签 字段	长度 字段	值字段		
23H	0CH	03H	03H	000A3BH
		03H	05H	045F291CD0H



BER编码示例 -- OBJECT IDENTIFIER

enterprise OBJECT IDENTIFIER ::=

{iso(1) org(3) dod(6) internet(1) private(4) 1}





BER编码示例 -- OCTET STRING

字节串值的编码与BIT STRING类似，但是不需要增加表征补充位个数的八位组。

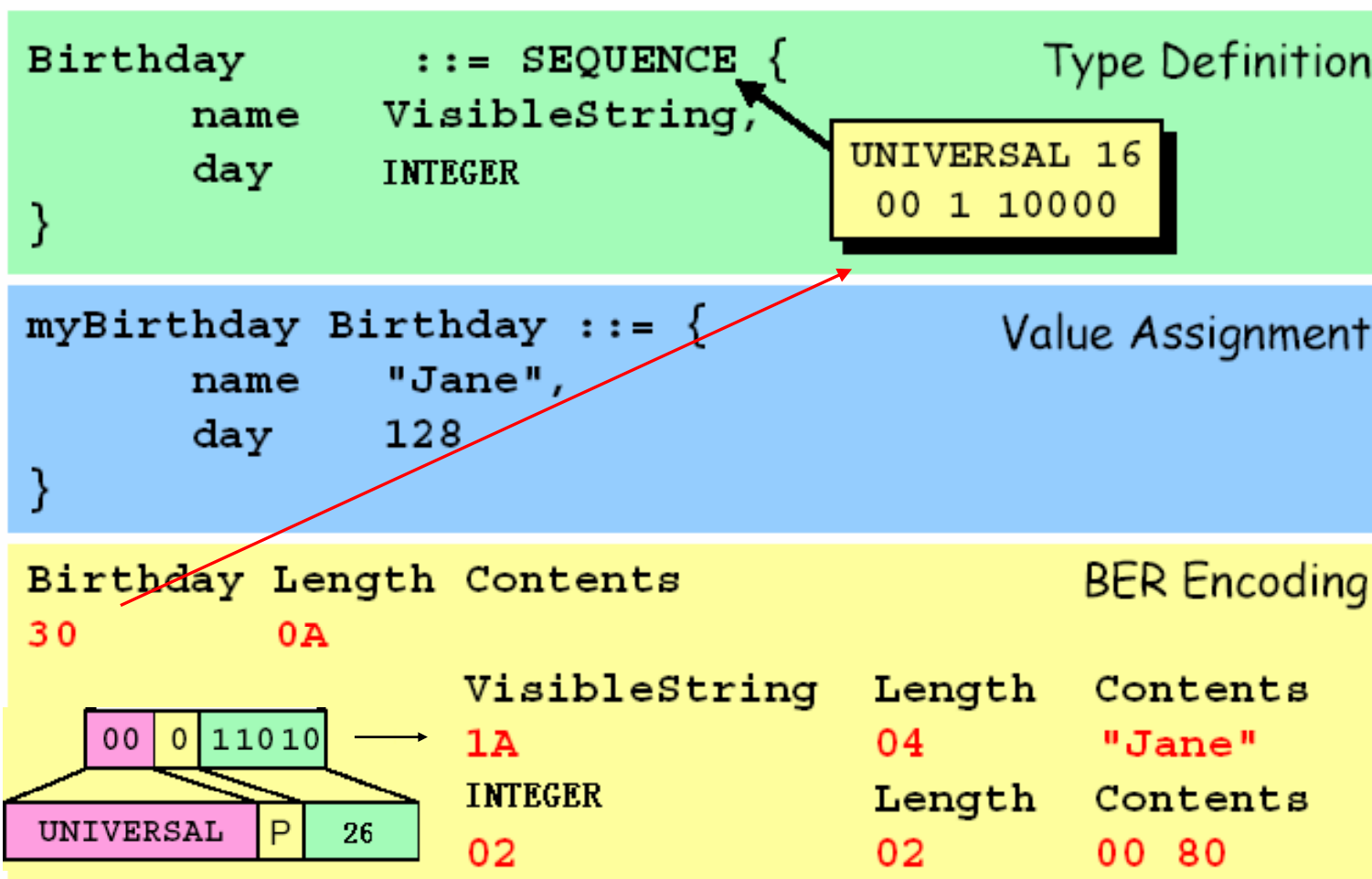
" ACE "

标签 字段	长度 字段	值字段
16H	03H	414345H





BER编码示例 -- SEQUENCE





BER编码示例 -- SET

由于集合类型的元素是无序的，故有多种编码，成员顺序由发送者决定。

SET{BREADTH INTEGER, BENT BOOLEAN}的
值{BREADTH 7, BENT FALSE}

编码为： 31 06 02 01 07 01 01 00

或 31 06 01 01 00 02 01 07





宏定义的模板

ASN.1宏提供了创建“模板”用来定义宏的方法，宏定义可以看做是类型的模板，MIB对象就是采用宏定义模板来定义。

● 宏定义模板如下：

```
<macroname> MACRO::=
```

```
BEGIN
```

```
    TYPE NOTATION ::= <user defined type notation>
```

-- 类型表示

```
    VALUE NOTATION ::= <user defined value notation>
```

-- 值表示

```
    <supporting syntax>
```

```
END
```

--支持产生式，对宏定义中类型的详细语法说明

<macroname> 是宏的名字，必须全部大写。



OBJECT-TYPE的宏定义

OBJECT-TYPE MACRO ::=

BEGIN

TYPE NOTATIN ::-“SYNTAX” type (ObjectSyntax)

“ACCESS” Access

“STATUS” Status

DescrPart

ReferPart

IndexPart

DefValPart

类型表示



OBJECT-TYPE的宏定义

值表示

VALUE NOTATION ::=value (VALUE ObjectName)

Access ::=“read-only”| “read-write”| “write-only”| “not-accessible”

Status ::=“mandatory”| “optional”| “obsolete”| “deprecated”

DescrPart ::=“DESCRIPTION” value (description DisplayString)| empty

ReferPart ::=“REFERENCE” value (reference DisplayString)| empty

IndexPart ::=“INDEX” “{” IndexTypes “}”| empty

IndexTypes ::=IndexType | IndexTypes “,” IndexType

IndexType ::=value (indexobject ObjectName)| type (indextype)

DefValPart ::=“DEFVAL” “{” value (defvalue ObjectSyntax) “}”| empty

END

对宏定义中类型的
详细语法说明



宏实例的定义

当用一个具体的值代替宏定义中的变量时就产生了宏实例，它表示一个实际的ASN.1类型，并且规定了该类型可取的值的集合。宏实例的定义首先是对象名，然后是宏定义的名字，最后是宏定义规定的宏体部分。

例1: tcpMaxConn OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

“The limit on the total number of TCP connection the
entity can support”

::={tcp 4}

例2: icmpInMsgs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

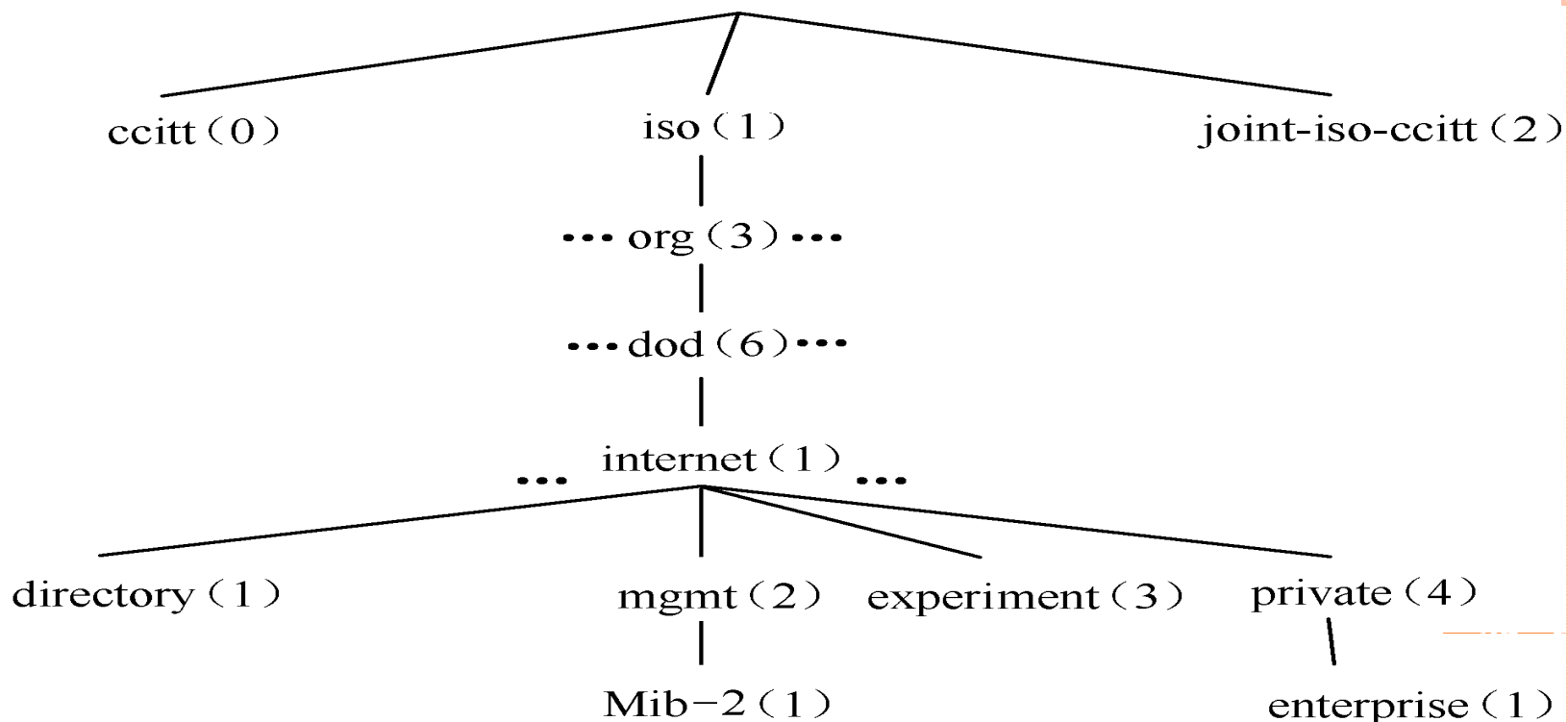
STATUS mandatory

::={icmp 1}



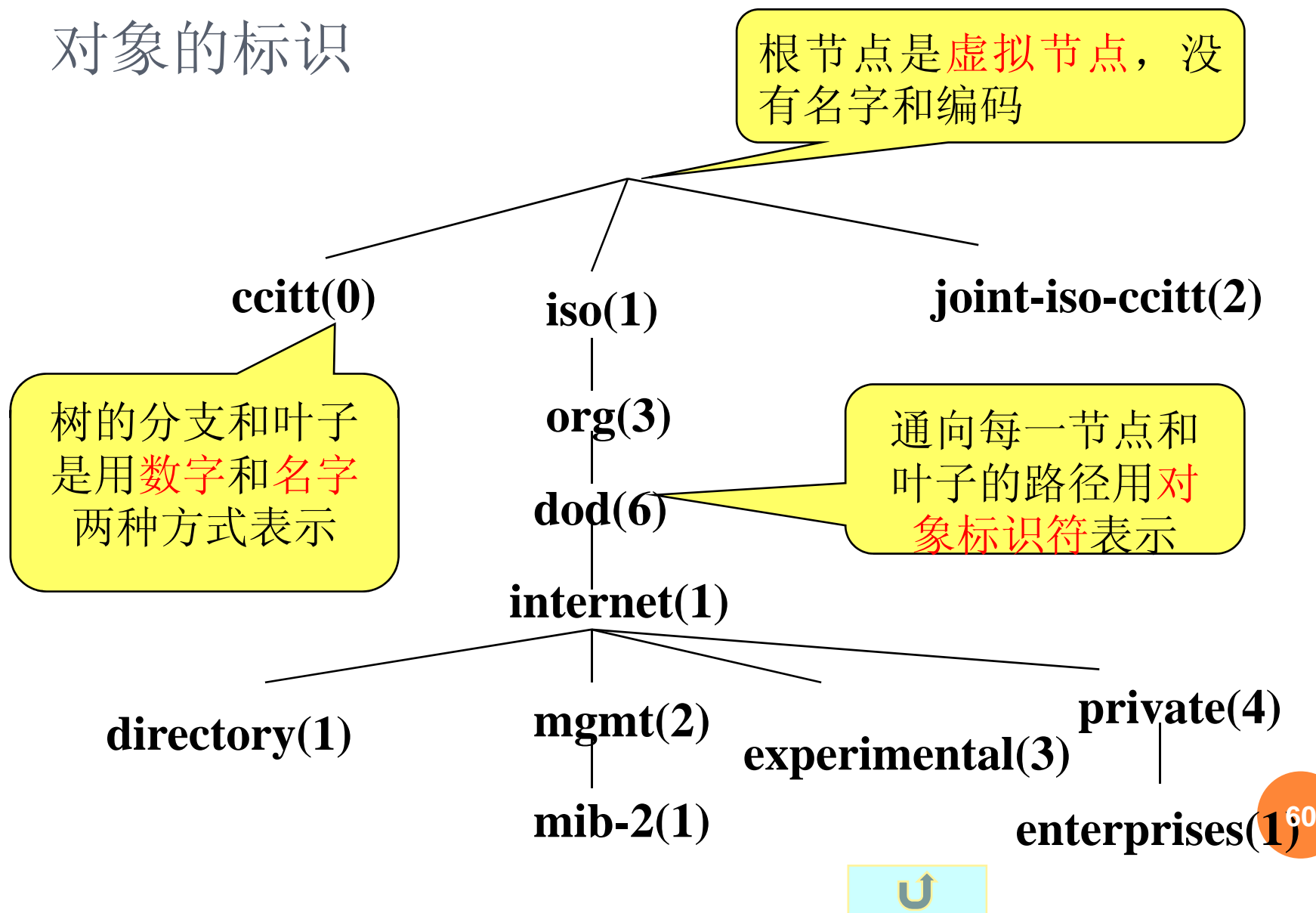
MIB树结构

SNMP环境中的所有管理对象组织成分层的树结构。采用这种层次树结构的组织方式易于管理，易于扩充。



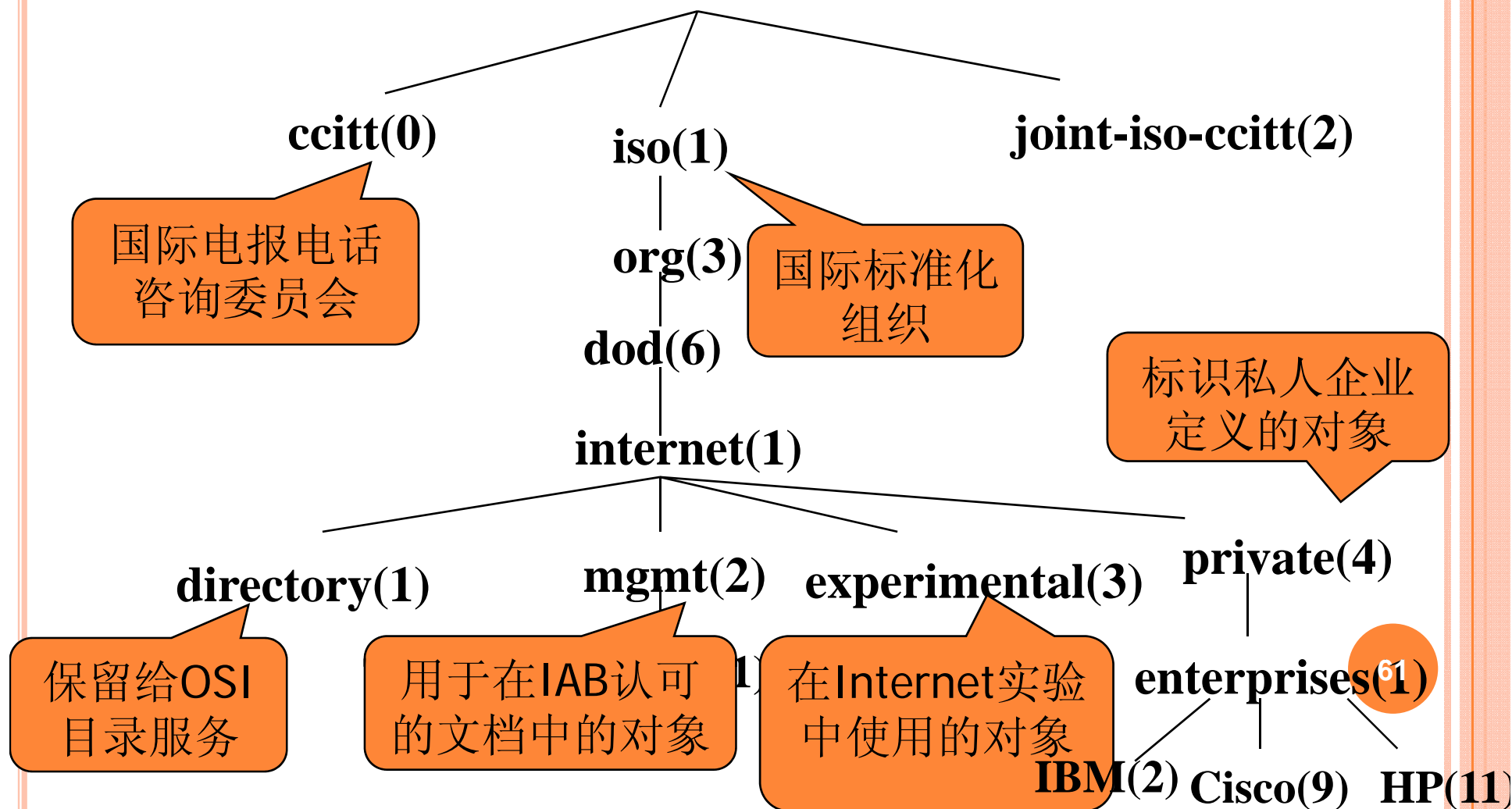


对象的标识





对象的标识





对象的标识

- 对象的命名方法例：

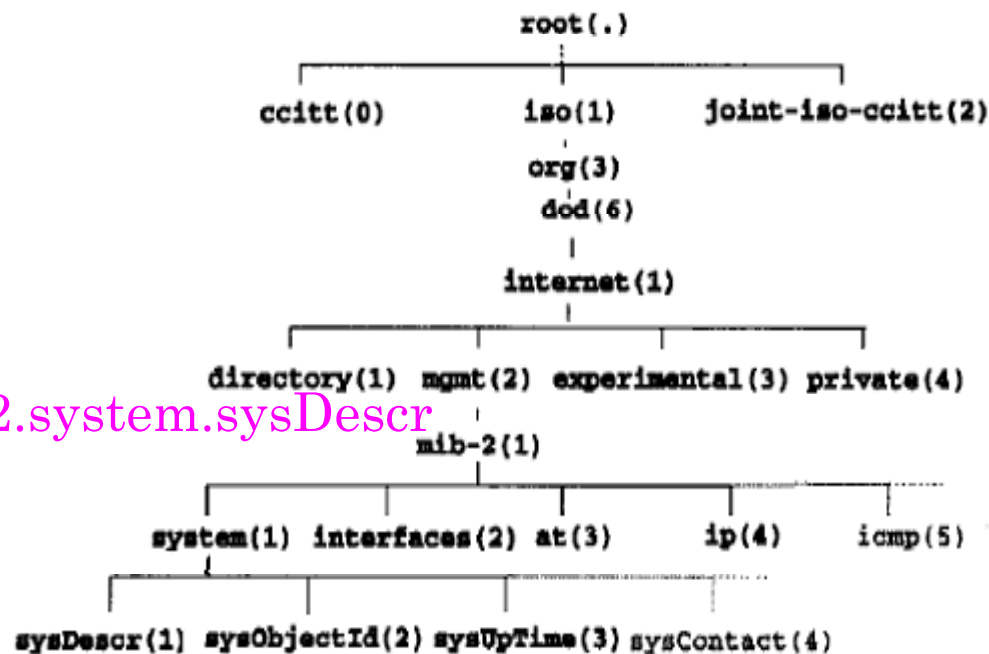
MIB对象sysDescr
的标识方法：

名字形式：

iso.org.dod.internet.mgmt.mib-2.system.sysDescr

数字形式：

1.3.6.1.2.1.1.1



可见：使用数字形式更易于存储和处理，且有共同前缀
1.3.6.1。



对象的标识

- 为了节省计算时间和存储空间，处理时可省略共同的前缀
- 对象标识符简记为父节点的名字标识和本节点的数字标识例：

SNMP定义的管理对象都在节点internet下，internet的对象标识符为：

internet OBJECT IDENTIFIER::={iso(1) org(3) dod(6) 1}

前缀：1.3.6.1

SNMP的管理对象记为：

mgmt OBJECT IDENTIFIER::={internet 2}

mib-2 OBJECT IDENTIFIER::={mgmt 1}

system OBJECT IDENTIFIER::={mib-2 1}

sysName OBJECT IDENTIFIER::={system 5}



表对象和标量对象

标量对象：指SMI中存储的简单对象和表中的列对象。

表对象：指SMI中存储的二维数组对象。



例：TCP连接表定义

tcpConnTable OBJECT-TYPE

SYNTAX SEQUENCE OF TcpConnEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

“A table containing TCP connection-specific information.”

::= { tcp 13 }



- tcpConnEntry OBJECT-TYPE
SYNTAX TcpConnEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
 "Information about a particular current TCP
 connection. An object of this type is transient,
 in that it ceases to exist when (or soon after) the
 connection makes the transition to the CLOSED state."
INDEX { tcpConnLocalAddress,
 tcpConnLocalPort,
 tcpConnRemAddress,
 tcpConnRemPort }
::= { tcpConnTable 1 }
TcpConnEntry ::=
 SEQUENCE {tcpConnState INTEGER,
 tcpConnLocalAddress IpAddress,
 tcpConnLocalPort INTEGER (0..65535),
 tcpConnRemAddress IpAddress,
 tcpConnRemPort INTEGER (0..65535)
 }



tcpConnState OBJECT-TYPE

SYNTAX INTEGER {closed(1), listen(2), synSent(3),
synReceived(4), established(5), finWait1(6),
finWait2(7), closeWait(8), lastAck(9),
closing(10), timeWait(11), deleteTCB(12)}

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The state of this TCP connection."

::= { tcpConnEntry 1 }



tcpConnLocalAddress OBJECT-TYPE

SYNTAX IPAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The local IP address for this TCP connection. "

::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The local port number for this TCP connection."

::= { tcpConnEntry 3 }



tcpConnRemAddress OBJECT-TYPE

SYNTAX IpAddress

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The remote IP address for this TCP connection."

::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The remote port number for this TCP connection."

::= { tcpConnEntry 5 }



TCP连接表

特点:

- (1) TCP连接表是TCP连接项组成的同类型序列
- (2) TCP连接项由5个不同类型标量组成
- (3) TCP连接表的索引由本地地址、本地端口、远程地址、远程端口组成



对象实例的标识

列对象：表中的标量对象

索引对象：用于区分表中的行

列对象的对象标识符与索引对象的值组合来说明列对象的一个实例；若索引对象有多个，则联上表中同一行的所有索引对象的值



TCP 连接表的实例

列对象
tcpConnState

TcpConnTable(1.3.6.1.2.1.6.13)

tcpConnEntry (1.3.6.1.2.1.6.13.1)

TcpConnState	TcpConnLocal Address	TcpConnLocalP ort	TcpConnRem Address	TcpConnRem Port
(1.3.6.1.2.1.6.13.1.1)	(1.3.6.1.2.1.6.13.1.2)	(1.3.6.1.2.1.6.13.1.3)	(1.3.6.1.2.1.6.13.1.4)	(1.3.6.1.2.1.6.13.1.5)
5	10.0.0.99	12	9.1.2.3	15
2	0.0.0.0	99	0.0.0.0	0
3	10.0.0.99	14	89.1.1.42	84
INDEX	INDEX	INDEX	INDEX	INDEX

实例标识符

索引对象

x.1.10.0.0.99. 12.9.1.2.3.15	x.2.10.0.0.99. 12.9.1.2.3.15	x.3.10.0.0.99.1 2.9.1.2.3.15	x.4.10.0.0.99. 12.9.1.2.3.15	x.5.10.0.0.99. 12.9.1.2.3.15
---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------

x=1.3.6.1.2.1.6.13.1=tcpConnEntry的对象标识符



对象实例的标识

对象实例标识的表示：**列对象的对象标识符+索引对象的值。**

x.1.10.0.0.99.1 2.9.1.2.3.15	x.2.10.0.0.99.12. 9.1.2.3.15	x.3.10.0.0.99.12. 9.1.2.3.15	x.4.10.0.0.99.12. .9.1.2.3.15	x.5.10.0.0.99.1 2.9.1.2.3.15
x.1.0.0.0.0.99. 0.0.0.0.0	x.2.0.0.0.0.99.0. 0.0.0.0	x.3.0.0.0.0.99.0. 0.0.0.0	x.4.0.0.0.0.99.0. 0.0.0.0	x.5.0.0.0.0.99. 0.0.0.0.0
x.1.10.0.0.99.1 4.89.1.1.42.84	x.2.10.0.0.99.14. 89.1.1.42.84	x.3.10.0.0.99.14. 89.1.1.42.84	x.4.10.0.0.99.14 .89.1.1.42.84	x.5.10.0.0.99.1 4.89.1.1.42.84

x=1.3.6.1.2.1.6.13.1=tcpConnEntry的对象标识符

列对象tcpConnState有3个实例，而3个实例的对象标识符都是1.3.6.1.2.1.6.13.1.1。索引对象的值是用于区分表中的行，这样，把列对象的对象标识符与索引对象的值组合起来就说明了列对象的一个实例。tcpConnTable的所有实例标识符的形式为

x.i.(tcpConnLocalAddress).(tcpConnLocalPort).(tcpConnRemAddress).(tcpConnRemPort)其中x为tcpConnEntry的对象标识符1.3.6.1.2.1.6.13.1，i为列对象的对象标识符的最后一个子标识符（指明列对象在表中的位置）。

一般规律：假定列对象的对象标识符是y，该对象所在的表有N个索引对象 i_1, i_2, \dots, i_N ，则它的某一行的实例标识符是：**y.(i₁). (i₂) \cdots (i_N)**



对象实例的标识

- 对象实例的值如何转换成子标识符？

转换规则：

如果索引对象实例的值为：

- 整数值，则把整数值作为一个子标识符。
- 定长字符串，则把每个字节(OCTET)编码为一个子标识符
- 可变长字符串，则把串长度 n 编码为第一个子标识符，每个字节编码为一个子标识符，共 $n+1$ 个
- 对象标识符，则把长度 n 编码为第一个子标识符，后续该对象标识符的各个子标识符，共 $n+1$ 个
- IP地址，则变为4个子标识符





对象实例的标识

标量对象

标量对象标识符之后级联一个**0**，表示该对象的实例标识符。

概念表和概念行

概念表和概念行是没有对象实例标识符。表对象和行对象就是概念表和概念行。其访问特性为“**not-accessible**”。



词典顺序访问技术

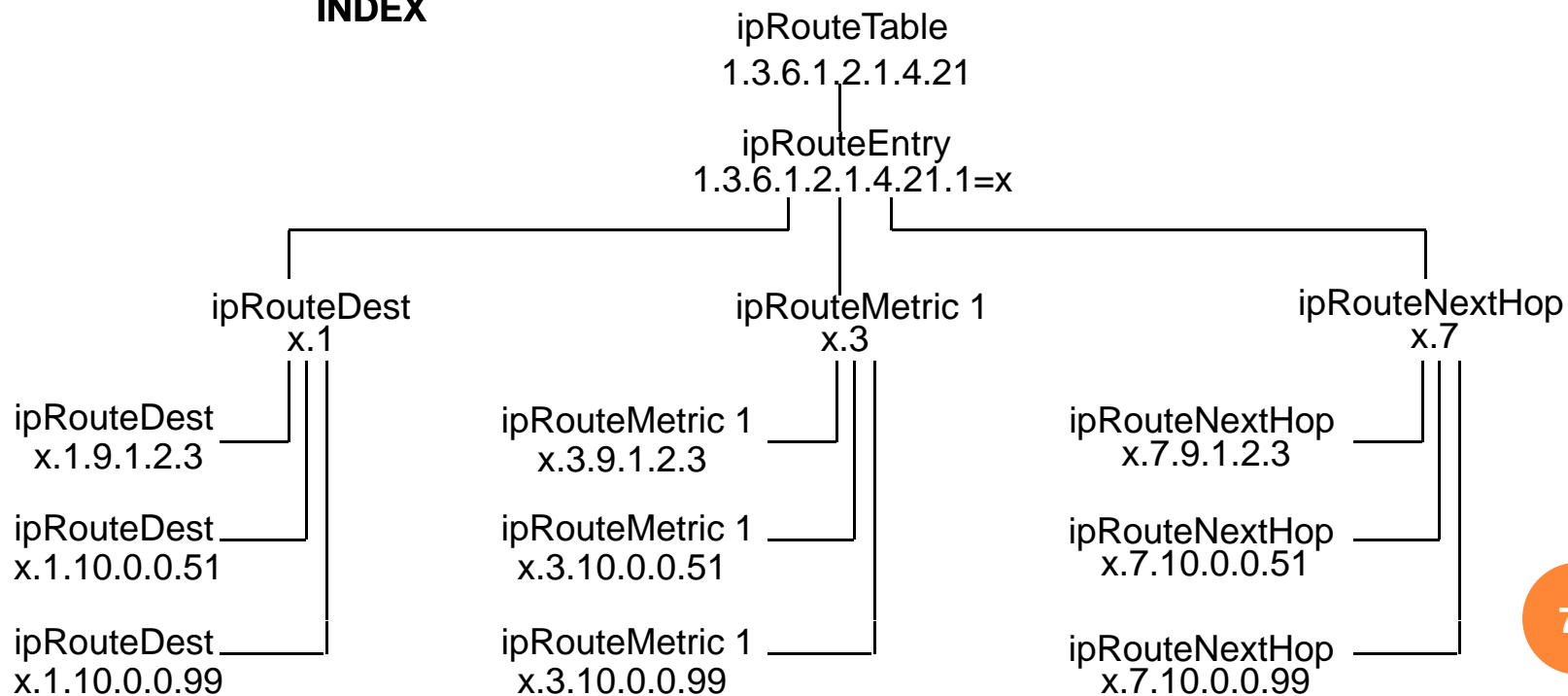
在MIB的树型结构中，跟踪从root开始到某个特定对象的路径，便可以得到该对象的对象标识符。即，可以通过遍历MIB中的对象标识符树来生成对象实例的词典顺序。



一个简化的IP路由表

ipRouteDest	ipRouteMetric 1	ipRouteNextHop
9.1.2.3	3	99.0.0.3
10.0.0.51	5	89.1.1.42
10.0.0.99	5	89.1.1.42

INDEX





IP路由表对象及其实例的词典顺序

对象	对象标识符	下一个对象实例
ipRouteTable	1.3.6.1.2.1.4.21	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteEntry	1.3.6.1.2.1.4.21.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest	1.3.6.1.2.1.4.21.1.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest.9.1.2.3	1.3.6.1.2.1.4.21.1.1.9.1.2.3	1.3.6.1.2.1.4.21.1.1.10.0.0.51
ipRouteDest.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.99
ipRouteDest.10.0.0.99	1.3.6.1.2.1.4.21.1.1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1	1.3.6.1.2.1.4.21.1.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3
ipRouteMetric1.9.1.2.3	1.3.6.1.2.1.4.21.1.3.9.1.2.3	1.3.6.1.2.1.4.21.1.3.10.0.0.51
ipRouteMetric1.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.51	1.3.6.1.2.1.4.21.1.3.10.0.0.99
ipRouteMetric1.10.0.0.99	1.3.6.1.2.1.4.21.1.3.10.0.0.99	1.3.6.1.2.1.4.21.1.7.9.1.2.3
ipRouteNextHop	1.3.6.1.2.1.4.21.1.7	1.3.6.1.2.1.4.21.1.7.9.1.2.3
ipRouteNextHop.9.1.2.3	1.3.6.1.2.1.4.21.1.7.9.1.2.3	1.3.6.1.2.1.4.21.1.7.10.0.0.51
ipRouteNextHop.10.0.0.51	1.3.6.1.2.1.4.21.1.7.10.0.0.51	1.3.6.1.2.1.4.21.1.7.10.0.0.99
ipRouteNextHop.10.0.0.99	1.3.6.1.2.1.4.21.1.7.10.0.0.99	



MIB- II

- MIB- II 是目前使用最广泛、最通用的MIB
- 任何声明支持S N M P标准的设备，都要求支持M I B - II。通过查询指定的M I B - II对象可获得关于特定设备或设备组的大量信息。
- M I B所包含的对象提供系统配置和网络性能信息。

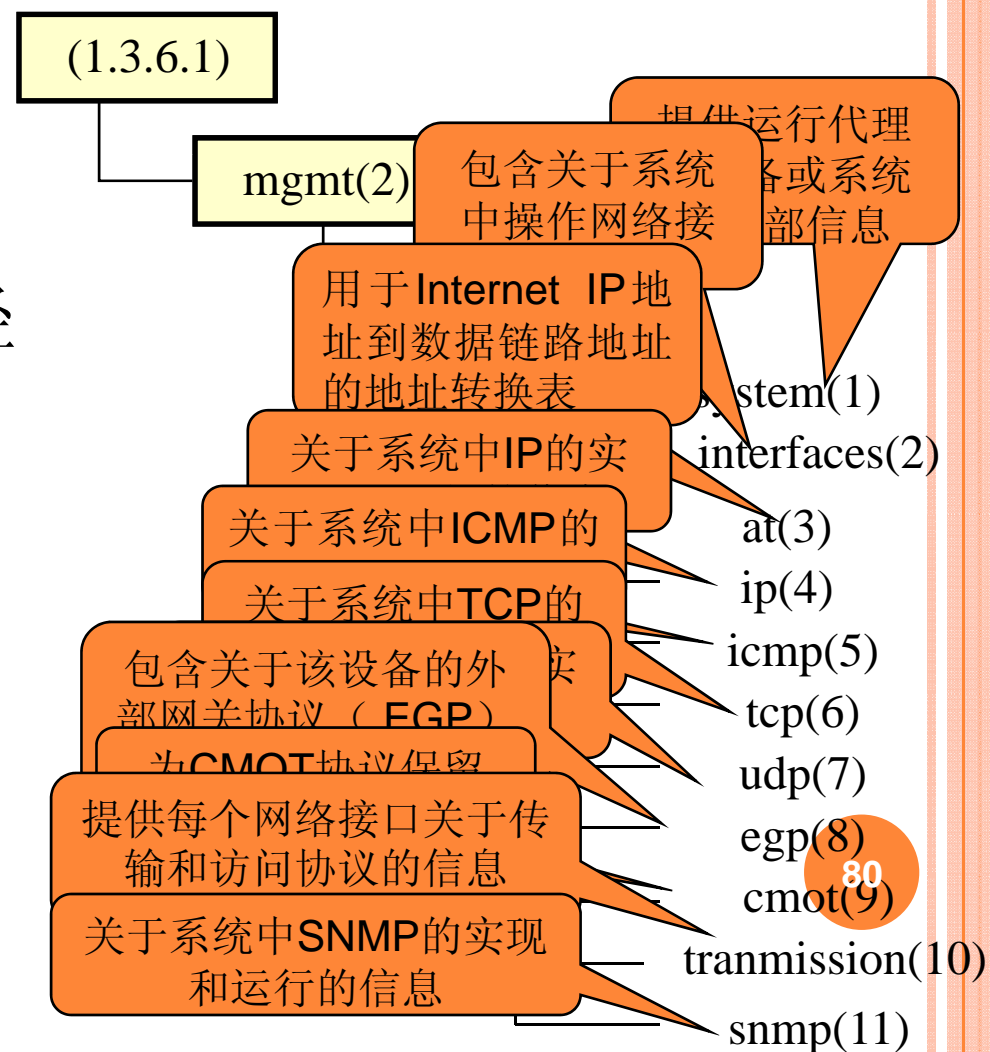


MIB- II

○ MIB- II的组成

其中:

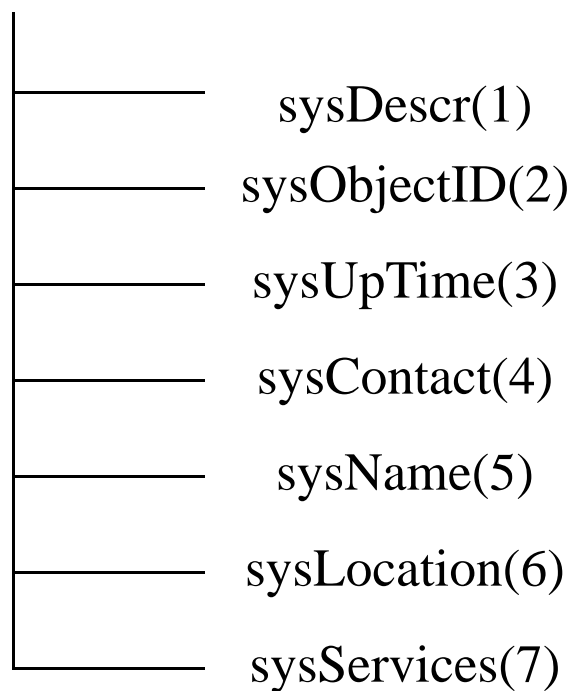
at(3), egp(8)已经
不用





3.3.1 SYSTEM组

system(mib-2 1)





3.3.1 SYSTEM组

■ **system(系统)**组提供了关于被管系统的总体信息。

1. 配置管理

(1) **sysDescr:**

OID: `system.1`

对象类型: `Display String [255]`

访问模式: 只读

描述: 关于该设备或实体的描述, 如设备类型、硬件特性、操作系统信息等



3.3.1 SYSTEM组

(2) sysContact:

OID: `system.4`

对象类型: `Display String [255]`

访问模式: 读写

描述: 记录其他提供该设备支持的机构和（或）联系人的信息



3.3.1 SYSTEM组

(3) sysName: 设备的名称

OID: `system.5`

对象类型: `Display String [255]`

访问模式: 读写

描述: 设备的名字, 可能是官方的主机名或者是分配的管理名字



3.3.1 SYSTEM组

(4) sysLocation:

OID: `system.6`

对象类型: `Display String [255]`

访问模式: 读写

描述: 该设备安装的物理位置



3.3.1 SYSTEM组

2. 故障管理

(1) sysObjectID:

O I D: s y s t e m . 2

对象类型: Object Identifier

访问模式: 只读

描述: 标识系统的生产厂商。

格式: 1.3.6.1.4.1.ve.vs.vs...

ve是厂商企业号，vs是厂商特定信息



3.3.1 SYSTEM组

(2) sysUpTime:

OID: `system.3`

对象类型: `TimeTick`

访问模式: 只读

描述: 给出SNMP代理已经运行的时间。以百分之一秒为单位



3.3.1 SYSTEM组

(3) `sysServices`: 显示一个设备可能提供的协议层服务。

`OID: system.7`

对象类型: `Integer`

访问模式: 只读

描述: 该设备提供的服务。 `sysServices` 对象表示一个7位代码，此代码与该设备提供服务的组合值相应。该代码中的每一位与OSI模型中的一层相关联，并且如果该设备在特定的层提供某服务，那么对应那一层的位被置位。



例如，对于一个交换机设备来说，该设备是1层和2层设备，可得：

$$2^{1-1} + 2^{2-1} = 3$$

一个应用服务器，可得：

$2^{4-1} + 2^{7-1} = 72$ 说明一个应用服务器在7层和4层提供服务。

问题：sysServices.0:-->6 表示该设备在哪些层提供服务？

3层和2层

sysServices.0:-->76 表示该设备在哪些层提供服务？

7层、4层和3层



3.3.2 INTERFACE组

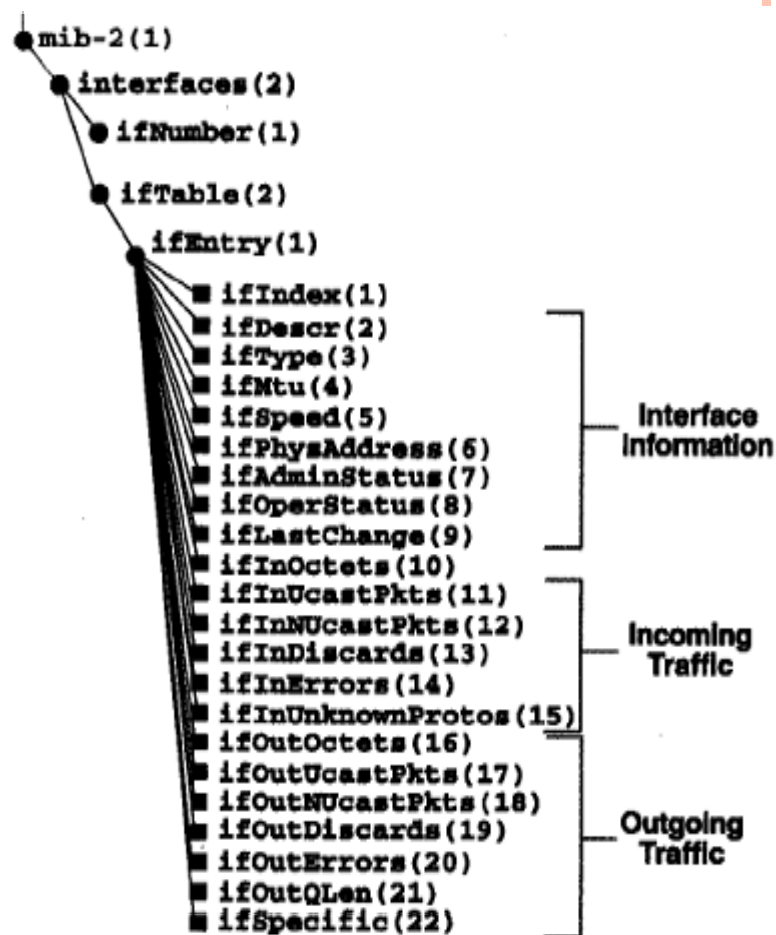
interface(接口)组提供了网络实体物理层接口的信息。接口组有两个顶级对象组成。

(1) ifNumber:

包含安装在该系统上的网络接口总数，并不管这些特定接口的操作状态。

(2) ifTable:

该对象包含一个表，表中每个接口一行。该表由ifIndex对象索引(其取值范围是1至ifNumber的值)。ifTable包含22个对象。



interfaces组视图



3.3.2 INTERFACE组

1. 配置管理

(1) ifDescr:

O I D: i f E n t r y . 2 .

对象类型: DisplayString [255]

访问模式: 只读

描述: 提供接口的描述。

例: BD6808/G16X3-Port 1:2

第1块板

第2个口

型号

该板16口



3.3.2 INTERFACE组

(2) ifType:

O I D: i f E n t r y. 3

对象类型: INTEGER

访问模式: 只读

描述: 标识接口的类型, 用一个整数表示。

例: ethernet-csmacd(6)



3.3.2 INTERFACE组

(3) ifMtu:

O I D: `i f E n t r y . 4`

对象类型: `I n t e g e r`

访问模式: 只读

描述: 在该接口上可以发送或接收的最大协议数据单元的大小。

例: 1500



3.3.2 INTERFACE组

(4) ifSpeed:

O I D: i f E n t r y . 5

对象类型: G a u g e

访问模式: 只读

描述: 指定一个接口的传输速率。单位为位/秒 (bps)，网络管理员可以重新设置

例: 1000000000 (1000Mbps)



3.3.2 INTERFACE组

(5) ifAdminStatus:

OID: `ifEntry.7`

对象类型: `Integer`

访问模式: 读写

描述: 用于配置接口的状态。该状态是

如下三种状态之一: `up(1)`、

`down(2)`和`testing(3)`



3.3.2 INTERFACE组

2.故障管理

(1) ifOperStatus:

OID: `ifEntry.8`

对象类型: `Integer`

访问模式: 只读

描述: 提供一个接口的当前工作状态。已定义的状态也是包括`up(1)`、`down(2)`和`testing(3)`

`ifAdminStatus`和`ifOperStatus`联合起来使用可确定接口的当前状态。



3.3.2 INTERFACE组

(2) ifLastChange:

O I D: i f E n t r y . 9

对象类型: T i m e T i c k s

访问模式: 只读

描述: 接口最后更新成当前操作状态的时间



3.3.2 INTERFACE组

(3) ifPhysAddress:

O I D: i f E n t r y . 6

对象类型: P h y s A d d r e s s

访问模式: 只读

描述: 该接口介质的物理地址。



3.3.2 INTERFACE组

3. 性能管理

(1) ifInOctets:

O I D: i f E n t r y . 1 0

对象类型: C o u n t e r

访问模式: 只读

描述: 从该接口上接收到的总字节数。



3.3.2 INTERFACE组

(2) ifInUcastPkts:

OID: ifEntry.11

对象类型: Counter

访问模式: 只读

描述: 传递给上层网络协议的单播报文包数



3.3.2 INTERFACE组

(3) ifInNUcastPkts:

OID: `ifEntry.12`

对象类型: `Counter`

访问模式: 只读

描述: 传递给上层网络协议的非单播报文包数（广播/多点发送包）



3.3.2 INTERFACE组

(4) ifInDiscards:

O I D: i f E n t r y . 1 3

对象类型: C o u n t e r

访问模式: 只读

描述: 由于资源局限导致丢弃的包的数目。
可能是设备存在拥塞问题造成。



3.3.2 INTERFACE组

(5) ifInErrors:

O I D: i f E n t r y . 1 4

对象类型: C o u n t e r

访问模式: 只读

描述: 由于错误导致丢弃的接收包的数目。
可能是接收器问题或坏线路问题造成。



3.3.2 INTERFACE组

(6) ifInUnkownProtos:

OID: `ifEntry.15`

对象类型: `Counter`

访问模式: 只读

描述: 由于未知或不支持的网络协议的导致丢弃的接收包的数目。



3.3.2 INTERFACE组

(7) ifOutOctets:

OID: `ifEntry.16`

对象类型: `Counter`

访问模式: 只读

描述: 从该接口上发送的字节总数。



3.3.2 INTERFACE组

(8) ifOutUcastPkts:

O I D: **i f E n t r y . 1 7**

对象类型: **C o u n t e r**

访问模式: 只读

描述: 请求传输到一个子网单点广播地址的包的总数目。该数量包括丢弃的和发送的报文数



3.3.2 INTERFACE组

(9) ifOutNUcastPkts:

OID: `ifEntry.18`

对象类型: `Counter`

访问模式: 只读

描述: 请求要发送的广播/多点发送包的总数目。该数量包括丢弃的和发送的报文数。



3.3.2 INTERFACE组

(10) ifOutDiscards:

O I D: **i f E n t r y . 1 9**

对象类型: **C o u n t e r**

访问模式: 只读

描述: 由于资源局限导致丢弃的发出包的总数目。可能是缓冲区不足造成。



3.3.2 INTERFACE组

(11) ifOutErrors:

O I D: **i f E n t r y . 2 0**

对象类型: **C o u n t e r**

访问模式: 只读

描述: 由于错误导致丢弃的发出包的总数目。
可能存在硬件问题



3.3.2 INTERFACE组

(12) ifOutQlen:

O I D: **i f E n t r y . 2 1**

对象类型: **G a u g e**

访问模式: 只读

描述: 输出包队列中包的总数。该值反映网络的拥挤情况



3.3.2 INTERFACE 组

- 接口性能特性
 - 利用率:

$$\text{利用率} = \frac{(\text{delta}(\text{ifInOctets}) + \text{delta}(\text{ifOutOctets})) \times 8}{\text{ifSpeed} \times \text{delta}(\text{seconds})} \times 100$$



3.3.2 INTERFACE组

- 接口性能特性

- 错误率:
- 丢包率:

$$\text{接收丢包率} = \frac{\text{delta}(iflInDiscards)}{\text{delta(seconds)}}$$

$$\text{发送丢包率} = \frac{\text{delta}(ifOutDiscards)}{\text{delta(seconds)}}$$

$$\text{接收错误率} = \frac{\text{delta}(iflInErrors)}{\text{delta(seconds)}}$$

$$\text{发送错误率} = \frac{\text{delta}(ifOutErrors)}{\text{delta(seconds)}}$$



3.3.2 INTERFACE组

- 接口性能特性
 - 接收和发送广播/多点发送包速率:

$$\text{接收广播/多点发送包速率} = \frac{\text{delta}(ifInNUcastPkts)}{\text{delta(seconds)}}$$

$$\text{发送广播/多点发送包速率} = \frac{\text{delta}(ifOutNUcastPkts)}{\text{delta(seconds)}}$$



- 接口性能的衡量标准有很多，几个主要计算方法包括：

$$\text{错误包比例} = \frac{\Delta \text{ifInErrors} + \text{ifOutErrors}}{\Delta \text{ifInErrors} + \Delta \text{ifOutErrors} + \Delta \text{ifInUcastPkts} + \Delta \text{ifInNUcastPkts} + \Delta \text{ifOutUcastPkts} + \Delta \text{ifOutNUcastPkts}} * 100\%$$

$$\text{错误包比例} = \frac{\Delta \text{ifInErrors}}{\Delta \text{ifInUcastPkts} + \Delta \text{ifInNUcastPkts}} * 100\%$$



$$\text{广播包比例} = \frac{\Delta \text{ifInNUcastPkts} + \Delta \text{ifOutNUcastPkts}}{\Delta \text{ifInUcastPkts} + \Delta \text{ifInNUcastPkts} + \Delta \text{ifOutUcastPkts} + \Delta \text{ifOutNUcastPkts}} * 100\%$$



3.3.2 INTERFACE组

4. 计费管理

ifInOctets

ifInUcastPkts

ifInNUcastPkts 3.3.3 地址转换组

ifOutOctets

ifOutUcastPkts

ifOutNUcastPkts



3.3.3 地址转换组

- 地址转换组(Address Translation)包含了一个表口，表中的每一行对应系统的一个物理接口，表示网络地址到接口的物理地址的映像。
- MIB-2中地址转换组的对象已被收编到各个网络协议组中，保留地址转换组仅仅是为了与MIB-1兼容。



3.3.4 IP组

- IP组：提供与IP协议有关的信息



} 表

ip组视图



3.3.4 IP组

1. 配置管理

(1) **ipForwarding:**

OID: ip.1

对象类型: Integer

访问模式: 读写

描述: 指出系统是否作为一个**IP**网关（路由器）或者仅作为一个不提供转发服务的正规主机。可取的值有**Forwarding** (1)和**not Forwarding** (2)



3.3.4 IP组

(2) **ipAddrTable**: 设备的IP地址表

ipAdEntAddr	物理网络接口相应的IP地址
ifAdEntIfIndex	索引, 从 interfaces 组的 ifIndex 对象处获得的接口号
ipAdEntNetMask	指出网络掩码或对IP地址 (ipAdEntAddr) 可用的子网掩码
ipAdEntBcastAddr	表示IP广播地址最低位的值。
ipAdEntReasmMaxSize	可以重组的最大IP数据报



3.3.4 IP组

(3) **ipRouteTable**: 设备的IP路由表

A. **ipRouteDest**

OID: **ipRouteEntry.1**

对象类型: **IpAddress**

访问模式: 读写

描述: 该路由定义的目标**IP**地址, 是索引项



3.3.4 IP组

B. `ipRouteIfIndex`

OID: `ipRouteEntry.2`

对象类型: `Integer`

访问模式: 读写

描述: 标识本地接口的索引，可通过该接口到达该路由的下一跳



3.3.4 IP组

C. **ipRouteMetric1**

OID: ipRouteEntry.3

对象类型: Integer

访问模式: 读写

描述: 该路由的主路由选择度量, 度量的实际定义由 **ipRouteProto** 值指定的路由选择协议来确定。

例: ipRouteMetric1.203.38.202.0:-->10

ipRouteMetric1.202.38.203.0:-->1

D. **ipRouteMetric2**

E. **ipRouteMetric3**

F. **ipRouteMetric4**

G. **ipRouteMetric5**

该路由的一个可选路由选择度量, 如果这个度量没有使用, 该值为-1



3.3.4 IP组

H. **ipRouteNextHop**

OID: ipRouteEntry.7

对象类型: **IpAddress**

访问模式: 读写

描述: 该路由下一跳的**IP**地址



3.3.4 IP组

I. **i p R o u t e T y p e**

O I D: i p R o u t e E n t r y. 8

对象类型: **I n t e g e r**

访问模式: 读写

描述: 该路由的类型, 用来表示直接和非直接路由选择。有效的类型包括: **other (1)**、**invalid(2)**、**direct(3)**和**indirect(4)**。

- **other**: 没有其他列出的值;
- **invalid**: 表示该路由无效;
- **direct**: 表示该目标地址直接连接到子网;
- **indirect**: 表示目标没直接连接到该网络, 并在到达目标之前至少通过另一个路由



3.3.4 IP组

J. ipRouteProto

OID: ipRouteEntry.9

对象类型: Integer

访问模式: 读写

描述: 用于了解路由的路由选择机制

路由协议: 1=其他, 4=ICMP重定向, 8=RIP, 13=OSPF, 14=BGP 等

例: ipRouteProto.192.168.11.0:-->ospf(13)



3.3.4 IP组

K. `ipRouteAge`

OID: `ipRouteEntry.10`

对象类型: `Integer`

访问模式: 读写

描述: 从上次路由被更新以来经过的时间（秒）



3.3.4 IP组

L. **i p R o u t e M a s k**

O I D: **i p R o u t e E n t r y. 11**

对象类型: **I p A d d r e s s**

访问模式: 读写

描述: 目标地址使用（与）的掩码地址



3.3.4 IP组

M. ipRouteInfo

OID: ipRouteEntry.13

对象类型: Object Identifier

访问模式: 只读

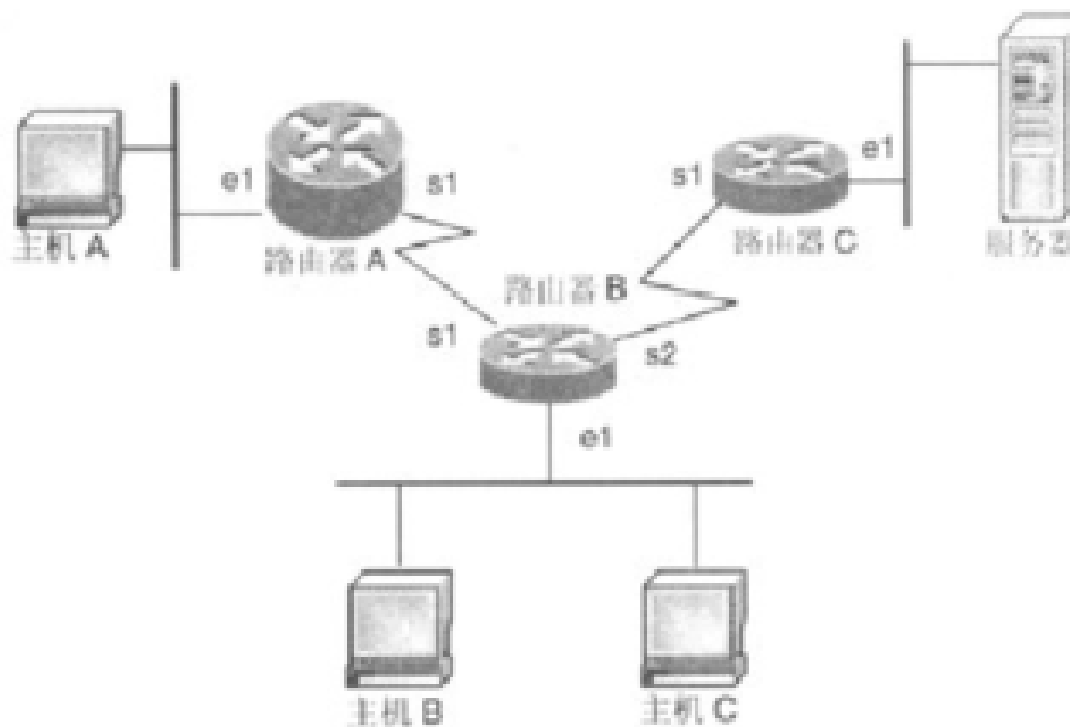
描述: 该字符串指定该路由所用的特定路由选择协议相关的MIB定义



3.3.4 IP组

2. 故障管理

(1) ipRouteTable: 设备的IP路由表





3.3.4 IP组

2. 故障管理

(2) **ipNetToMediaTable**: 设备的IP地址表, 记录了IP地址到物理地址的映射。

A. **ipNetToMediaIfIndex**

OID: **ipNetToMediaEntry.1**

对象类型: **Integer**

访问模式: 读写

描述: 用来标识符本地接口的索引, 网络地址和物理地址映射是从该接口获得的



3.3.4 IP组

B.i p N e t T o M e d i a P h y s A d d r e s s

O I D: i p N e t T o M e d i a E n t r y. 2

对象类型: **P h y s A d d r e s s**

访问模式: 读写

描述: 介质相关的物理地址;



3.3.4 IP组

C.i p N e t T o M e d i a N e t A d d r e s s

O I D: i p N e t T o M e d i a E n t r y. 3

对象类型: **I p A d d r e s s**

访问模式: 读写

描述: 对应该介质相关物理地址的**I P**地址



3.3.4 IP组

D.i p N e t T o M e d i a T y p e

O I D: i p N e t T o M e d i a E n t r y. 4

对象类型: **C o u n t e r**

访问模式: 读写

描述: 产生地址映射的类型。类型包括: **other(1)**、**invalid(2)**、**dynamic(3)**和**static(4)**



3.3.4 IP组

3. 性能管理

(1) ipInReceives

OID: `ip.3`

对象类型: `Counter`

访问模式: 只读

描述: 从系统所有可操作接口接收的输入报文的总数，包括错误的数据报。



3.3.4 IP组

(2) ipInHdrErrors

OID: `ip.4`

对象类型: `Counter`

访问模式: 只读

描述: 由于**IP**报文头部错误而丢弃的输入报文数量



3.3.4 IP组

(3) IpInAddrErrors

O I D: `ip.5`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 因为最终**IP**目的地址无效而被丢弃的
输入报文数量



3.3.4 IP组

(4) IpForwDatagrams

O I D: `i p . 6`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 本地系统作为网关或路由器转发的报文数量



3.3.4 IP组

(5) IpInUnknownProtos

O I D: **i p . 7**

对象类型: **Network Address**

访问模式: 只读

描述: 从网络上成功接收，但由于系统对报文所请求的网络层协议不支持或者未知，而丢弃的报文数量



3.3.4 IP组

(6) IpInDiscards

OID: `ip.8`

对象类型: `Counter`

访问模式: 只读

描述: 由于资源有限（例如缓冲空间不足）
而丢弃的输入报文的数量



3.3.4 IP组

(7) IpInDelivers

OID: ip.9

对象类型: Counter

访问模式: 只读

描述: 成功传递给上层协议的输入报文的数量



3.3.4 IP组

(8) IpOutRequests

O I D: `ip.10`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 上层协议为发送而传递给IP协议的IP报文的数量。



3.3.4 IP组

(9) IpOutDiscards

O I D: `i p . 11`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 由于资源有限（例如缓冲空间不足）而丢弃的输出报文的数量



3.3.4 IP组

(10) ipOutNoRoutes

O I D: `ip.12`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 因为无法找到到达目标地址的路由而
丢弃的输出的报文数量

如果该值较高,可能设备路由表配置不正确或
没有正确接收路由更新



3.3.4 IP组

(11) ipReasmTimeout

O I D: i p . 1 3

对象类型: Interger

访问模式: 只读

描述: 接收到的**I P**分组报文在等待被重组之前保留的时间间隔（以秒为单位）



3.3.4 IP组

(12) IpReasmReqds

O I D: `ip.14`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 接收到的需要重组的**I P**分组报文数量
如果该值较高, 可能是MTU不匹配引起的.



3.3.4 IP组

(13) ipReasmOKs

OID: ip.15

对象类型: Counter

访问模式: 只读

描述: 成功重组的IP报文的数量



3.3.4 IP组

(14) ipReasmFails

OID: `ip.16`

对象类型: `Counter`

访问模式: 只读

描述: 检测到的重组失败的次数

如果该值较高,可能分段已损坏或由于资源不足等原因使IP分段被丢弃



3.3.4 IP组

(15) IpFragOKs

OID: `ip.17`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 已经被成功分组的报文数量



3.3.4 IP组

(16) IpFragFails

OID: `ip.18`

对象类型: `Counter`

访问模式: 只读

描述: 需要分组但因为IP头部包含不分组标志而不得不丢弃的IP数据报文数量

如果该值较高,可能需要修改设备的Don't Fragment标志



3.3.4 IP组

(17) IpFragCreates

OID: `ip.19`

对象类型: `Counter`

访问模式: 只读

描述: 该系统上产生的**IP**报文分组的数量
如果该值较高, 可能是MTU不匹配引起的.



3.3.4 IP组

(18) IpRoutingDiscards

O I D: `ip.23`

对象类型: `C o u n t e r`

访问模式: 只读

描述: 即使有效但是被选定要释放的路由表项的数量。导致丢弃这样一个表项的一种可能的原因是为其他的路由表项释放缓存空间



3.3.4 IP组

4. 计费管理

(1) **ipOutRequests**

(2) **ipInDelivers**



3.3.4 IP组

5. 其他

ipDefaultTTL

OID: `ip.2`

对象类型: `Integer`

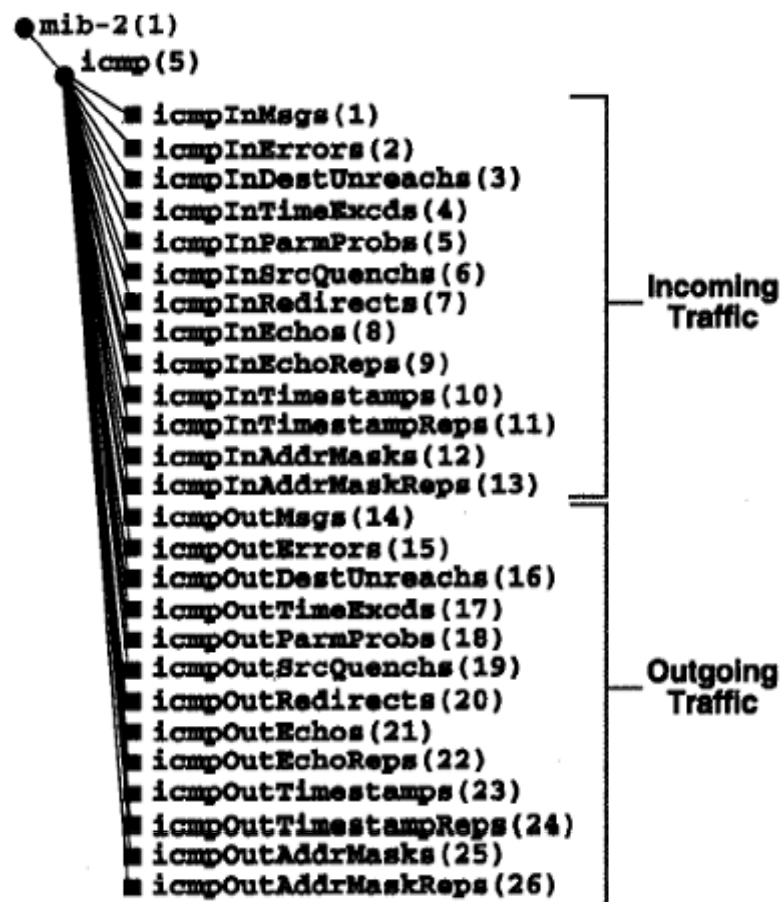
访问模式: 读写

描述: 传输层协议没有提供TTL值时置于**IP**报文的**TTL**字段中的生存期值。



3.3.5 ICMP组

- ICMP: IP的伴随协议, 主要用于性能管理。



icmp组视图



3.3.5 ICMP组

(1) icmpInMsgs

描述： 接收到的ICMP消息的数量

(2) icmpInErrors

描述： 接收到的包含某些ICMP指定错误的ICMP消息的数量

(3) icmpInDestUnreachs

描述： 接收到的ICMP目标不可达消息的数量

(4) icmpInTimeExcds

描述： 接收到的ICMP超时消息的数量

(5) icmpInParmProbs

描述： 接收到的ICMP参数错误消息的数量

(6) icmpInSrcQuenches

描述： 接收到源端抑制的ICMP消息的数量

(7) icmpInRedirects

描述： 接收到的ICMP重定向消息的数量



3.3.5 ICMP组

(8) icmpInEchos

描述： 接收到的ICMP回应请求消息的数量

(9) icmpInEchoReps

描述： 接收到的ICMP回应应答消息的数量

(10) icmpInTimestamps

描述： 接收到的请求时间戳ICMP消息的数量

(11) icmpInTimestampReps

描述： 接收到的返回时间戳ICMP消息的数量

(12) icmpInAddrMasks

描述： 接收到的请求地址掩码ICMP消息的数量

(13) icmpAddrMasksReps

描述： 接收到的应答地址掩码ICMP消息的数量

(14) icmpOutMsgs

描述： 该系统试图发送的ICMP消息的数量



3.3.5 ICMP组

(15) icmpOutErrors

描述： 该系统由于ICMP问题(例如缓冲区不足)而不能发送的ICMP消息的数量

(16) icmpOutDestUnreachs

描述： 该系统发送的ICMP目标不可达消息的数量

(17) icmpOutTimeExcds

描述： 该系统发送的ICMP超时消息的数量

(18) icmpOutParmProbs

描述： 该系统发送的ICMP参数有问题的消息的数量

(19) icmpOutSrcQuenches

描述： 该系统发送的ICMP源端抑制消息的数量

(20) icmpOutRedirects

描述： 该系统发送的ICMP重定向消息的数量

(21) icmpOutEchos

描述： 该系统发送的ICMP回应请求消息的数量



3.3.5 ICMP组

(22) icmpOutEchoReps

描述： 该系统发送的ICMP回应应答消息的数量

(23) icmpOutTimestamps

描述： 该系统发送的请求ICMP时间戳消息的数量

(24) icmpOutTimestampReps

描述： 该系统发送的应答ICMP时间戳消息的数量

(25) icmpOutAddrMasks

描述： 该系统发送的请求ICMP地址掩码消息的数量

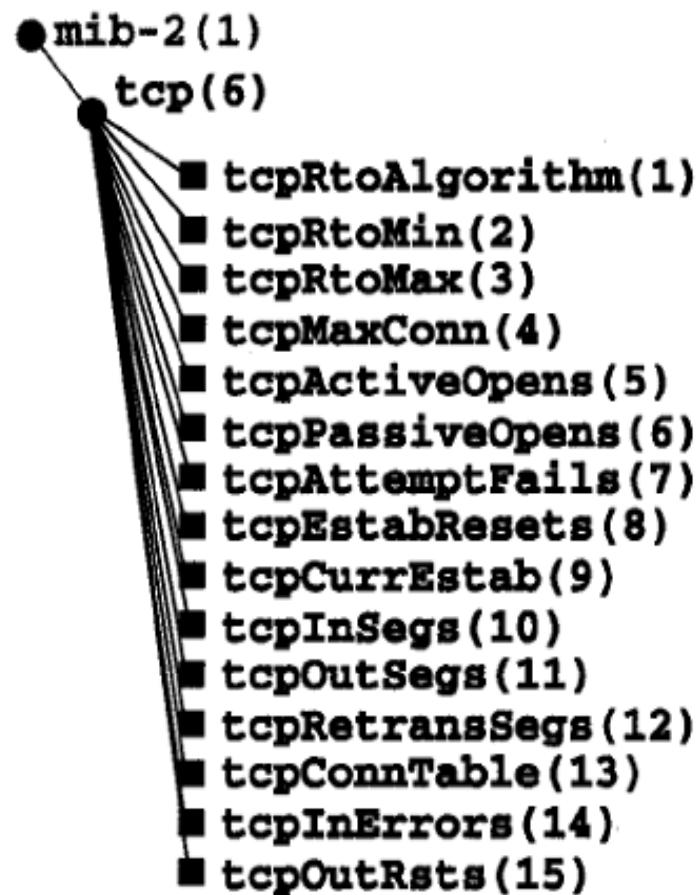
(26) icmpOutAddrMaskReps

描述： 该系统发送的应答ICMP地址掩码消息的数量



3.3.6 TCP组

TCP组存放关于系统上传输控制协议（**TCP**）的统计数据 and 操作信息。



tcp组视图



3.3.6 TCP组

1. 配置管理

对象	说明
tcpRtoAlgorithm	TCP重传算法
tcpRtoMin	TCP应用允许的超时重传的最小值(以毫秒为单位)。
tcpRtoMax	TCP应用允许的超时重传的最大值(以毫秒为单位)。
tcpMaxConn	允许最大 TCP 连接数。在实体中最大连接数是动态变化的，因此这种对象应该包含数值-1。
tcpCurrEstab	当前 TCP 连接数(处于 ESTABLISHED 或 CLOSE-WAIT 状态)



3.3.6 TCP组

2. 性能管理

对象	说明
tcpAttempFails	建立连接失败次数
tcpEstabResets	已经直接从 ESTABLISHED 状态或 CLOSE- WAIT 状态转换到 CLOSED 状态的 TCP 连接的次数。
tcpInSegs	接收的 TCP 段总数，包括接收到的错误分组
tcpOutSegs	发送的 TCP 段总数,但除了那些只包含重发的字节的分组
tcpRetransSegs	重传的 TCP 段数
tcpInErrs	接收出错的 TCP 段数
tcpOutRsts	发出的含 RST 标志的段数



3.3.6 TCP 组

- 计算TCP段进入和离开实体的速率

1. TCP段输入速率：

2.
$$\text{TCP 段输入速率} = \frac{\text{delta(tcpInSegs)}}{\text{delta(seconds)}}$$

$$\text{TCP 段输出速率} = \frac{\text{delta(tcpOutSegs)}}{\text{delta(seconds)}}$$



3.3.6 TCP组

3. 计费管理

对象	说明
tcpActiveOpens	实体打开 TCP 连接的次数
tcpPassiveOpens	实体收到的请求连接的次数
tcpInSegs	输入的 TCP 段数
tcpOutSegs	输出的 TCP 段数
tcpConnTable	TCP 连接表



3.3.6 TCP组

4. 安全管理

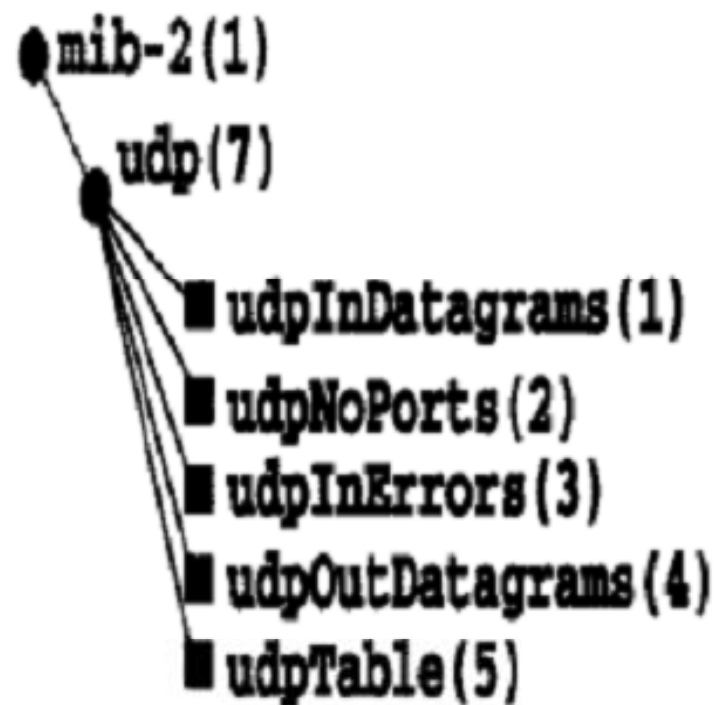
使用**tcpConnTable**的数据跟踪连接情况

对象	说明
tcpConnState	连接状态
tcpConnLocalAddress	本地IP
tcpConnLocalPort	本地TCP端口
tcpConnRemAddress	远程IP
tcpConnRemPort	远程TCP端口



3.3.7 UDP组

u d p组包含的对象是
关于系统中用户数
据报协议 (**UDP**)
执行和操作及本地
接收点的详细信息



udp组视图



3.3.7 UDP组

1. 性能管理

对象	说明
udpInDatagrams	接收到的 UDP 数据报数目
udpNoPorts	没有发送到有效 UDP 端口数据报的数量
udpInErrors	
udpOutDatagrams	

udpInDatagrams.0:-->140920184

udpNoPorts.0:-->691654

udpInErrors.0:-->104

udpOutDatagrams.0:-->141241761

不
目
目
目



3.3.7 UDP组

2. 计费管理

使用**udpInDatagrams**和**udpOutDatagrams**



3.3.7 UDP组

3. 安全管理

对象	说明
udpLocalAddress	UDP 侦听的本地地址
udpLocalPort	UDP 侦听的本地端口



3.3.8 EGP 组

对象	语法	访问方式	功能描述
egpInMsgs	Counter	RO	收到的无错的EGP消息数
egpInErrors	Counter	RO	收到的有错的EGP消息数
egpOutMsgs	Counter	RO	本地产生的EGP消息总数
egpOutErrors	Counter	RO	由于资源限制没有发出的本地产生的EGP消息数
egpNeighTable	SEQUENCE OF EgpNeighEntry	NA	相邻网关的EGP表 (表内的对象略)
egpAs	INTEGER	RO	本EGP实体的自治系统数




3.3.9 TRANSMISSION组

- **transmission**针对各种传输介质提供详细的管理信息。该组的主要目的是用特定接口**MIB**的形式提供特定接口信息。



3.3.10 SNMP组

snmp组包含
SNMP操作和
执行的相关信息。该组中的一些对象是与
SNMP站管理
或代理功能相关的



```
snmp(11)
■ snmpInPkts(1)
■ snmpOutPkts(2)
■ snmpInBadVersions(3)
■ snmpInBadCommunityNames(4)
■ snmpInBadCommunityUses(5)
■ snmpInASNParseErrs(6)
■ snmpInTooBigs(8)
■ snmpInNoSuchNames(9)
■ snmpInBadValues(10)
■ snmpInReadOnlys(11)
■ snmpInGenErrs(12)
■ snmpInTotalReqVars(13)
■ snmpInTotalSetVars(14)
■ snmpInGetRequests(15)
■ snmpInGetNexts(16)
■ snmpInSetRequests(17)
■ snmpInGetResponses(18)
■ snmpInTraps(19)
■ snmpOutToBigs(20)
■ snmpOutNoSuchNames(21)
■ snmpOutBadValues(22)
■ snmpOutGenErrs(24)
■ snmpOutGetRequests(25)
■ snmpOutGetNexts(26)
■ snmpOutSeqRequests(27)
■ snmpOutGetResponses(28)
■ snmpOutTraps(29)
■ snmpEnableAuthenTraps(30)
```

snmp组视图



3.3.10 SNMP组

1. 配置管理

snmpEnableAuthenTraps: 控制代理是否允许产生认证失败（ authentication failure）的trap操作。有效值为**enabled(1)**或**disabled(2)**



3.3.10 SNMP组

2. 故障管理

对象	说明
snmpInASNParseErrs	对接收到的 S N M P 消息解码时，发生 BER 和 ASN.1 错误的 SNMP 消息数
snmpInTooBigs	接收到的具有 tooBig 错误的 PDU 数
snmpInNoSuchName	接收到的具有 noSuchName 错误的 PDU 数
snmpInBadValues	接收到的具有 badValue 错误的 PDU 数
snmpInReadOnlyls	接收到的具有 readOnly 错误的 PDU 数
snmpInGenErrs	接收到的具有 genErr 错误的 PDU 数



3.3.10 SNMP组

对象	说明
snmpInBadVersions	传递给该设备，但该设备不支持的 SNMP 消息的数量
snmpOutTooBig	由该设备产生的 SNMP PDU 发生 tooBig 错误的数量
snmpOutNoSuchNames	由该设备产生的 SNMP PDU 发生 noSuchName 错误的数量
snmpOutBadValues	由该设备产生的 SNMP PDU 发生 badValue 错误的数量
snmpOutGenErrs	由该设备产生的 SNMP PDU 发生 genErr 错误的数量



3.3.10 SNMP组

3. 性能管理

(1) snmpInPkts

描述：接收到的S N M P消息的数量

(2) snmpOutPkts

描述：发送的S N M P消息的数量

(3) snmpInTotalReqVars

描述：由该设备通过一个有效的SNMP **get-request**和
getnext - request成功取回的MIB对象的数量

(4) snmpInTotalSetVars

描述：由该设备通过一个有效的SNMP **set-request**成功
修改的M I B对象的数量



3.3.10 SNMP组

(5) **snmpInGetRequests**

描述： 由该设备接收并处理的SNMP get-requests的数量

(6) **snmpInGetNexts**

描述： 由该设备接收并处理的SNMP getnext-requests的数量

(7) **SnmpInSetRequests**

描述： 由该设备接收并处理的SNMP set-requests的数量

(8) **snmpInGetResponses**

描述： 由该设备接收并处理的SNMP get-responses的数量



3.3.10 SNMP组

(9) snmpInTraps

描述： 由该设备接收并处理的SNMP traps的数量

(10) snmpOutGetRequests

描述： 由该设备发送的SNMP get-request PDU的数量

(11) SnmpOutGetNexts

描述： 由该设备发送的SNMP getnext-request PDU的数量

(12) SnmpOutSetRequests

描述： 由该设备发送的SNMP set-request PDU的数量

(13) SnmpOutGetResponses

描述： 由该设备发送的SNMP get- responses PDU的数量

(14) SnmpOutTraps

描述： 由该设备发送的SNMP traps PDU的数量



3.3.10 SNMP组

4. 计费管理

- (1) **snmpInPkts**
- (2) **snmpOutPkts**
- (3) **snmpInTraps**
- (4) **snmpOutTraps**

5. 安全管理

- (1) **snmpInBadCommunityNames**: 接收含团体名错误的报文数
- (2) **snmpInBadCommunityUses**: 传递给该代理的某种SNMP消息的数量, 这种SNMP消息包含一个SNMP操作, 而该操作根据提供的团体名不允许执行



本章小结

- 本章首先介绍了管理信息库（MIB）的概念；然后介绍了管理信息结构（SMI）的概念、MIB的结构、MIB的数据类型、标量对象和表结构的表示方法等内容；最后对MIB-2功能组进行了介绍。
- SMI定义了一些通用规则，包括命名对象，定义对象类型，以及表示如何把对象和值进行编码。MIB在需要被管理的实体中创建了命名对象，它们的值以及它们彼此之间关系的集合。MIB如同程序设计语言中的定义变量，包括变量类型和名称；SMI则如同程序设计语言中定义的编码规则、语法、变量的结构、数据类型等。
- 本章重点是掌握MIB、SMI的相关概念；理解对象标识、实例表示的规则和方法；了解MIB-2组中相关对象的功能。



作业

1. 对象标识符1.3.6.1.2.1.4.1的 BER 编码为多少?
2. 对象标识符是由什么组成的? 网络中的设备是如何表示的?
3. 什么是标量对象? 什么是表对象? 它们的实例是如何标识的?
4. 日常网络监控,当发现网络不通时可以通过MIB哪个信息进行检查和判断

选做题

画出Host Resources MIB (rfc2790)的树型结构