SMI

STRUCTURE OF MANAGEMENT INFORMATION

RFC 1155: SMIv1

RFC 1212: CONCISE MIB DEFINITIONS

RFC 2578: SMIv2

RFC 2579: TEXTUAL CONVENTIONS

MAKES THE DEFINITION OF (NEW) MIBs EASIER

SMI

MANAGEMENT INFORMATION WITHIN MANAGED SYSTEMS
MUST BE REPRESENTED AS:

- SCALARS
- TABLES

(= TWO DIMENSIONAL ARRAYS OF SCALARS)

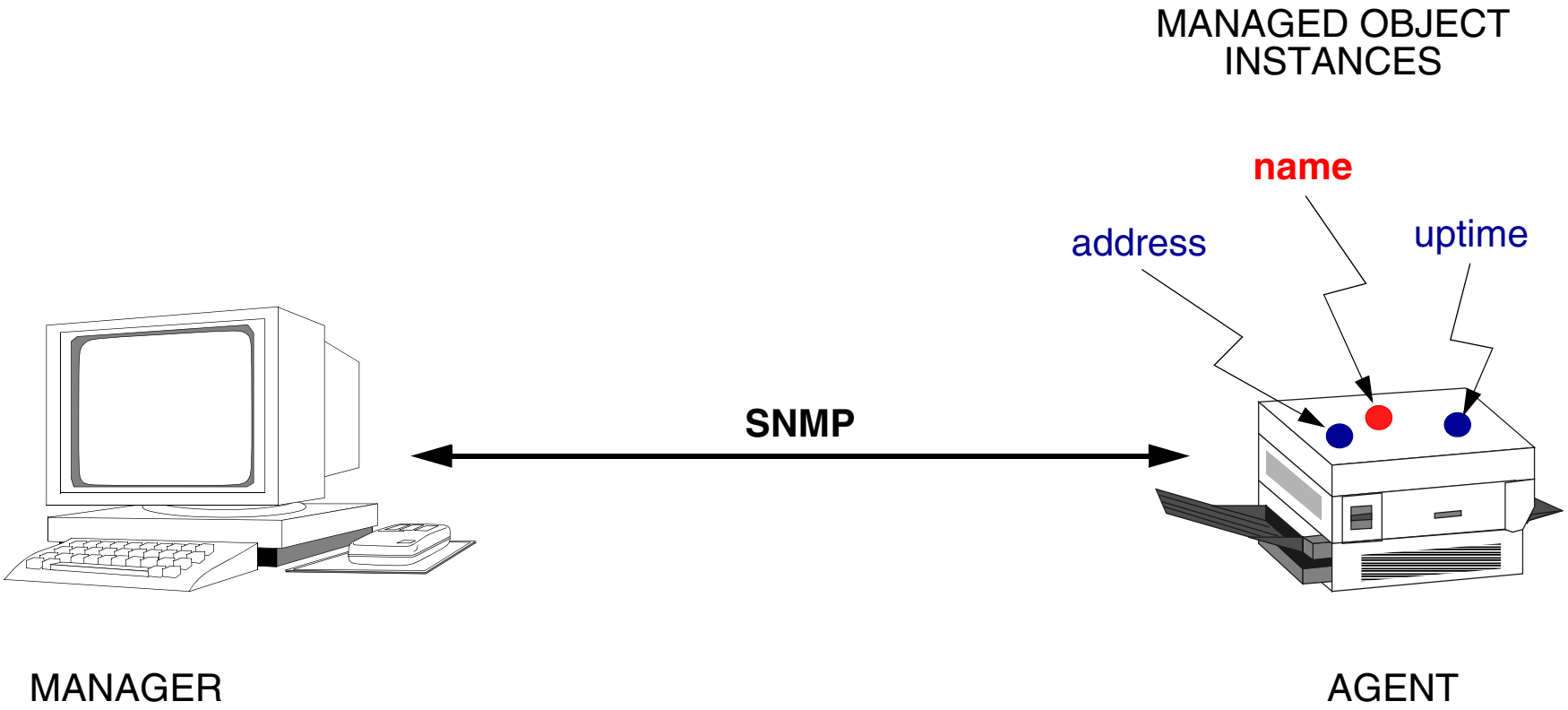
THE SNMP PROTOCOL CAN ONLY EXCHANGE
(A LIST OF) SCALARS

DEFINED IN TERMS OF ASN.1 CONSTRUCTS

SMI: DATA TYPES FOR SCALARS

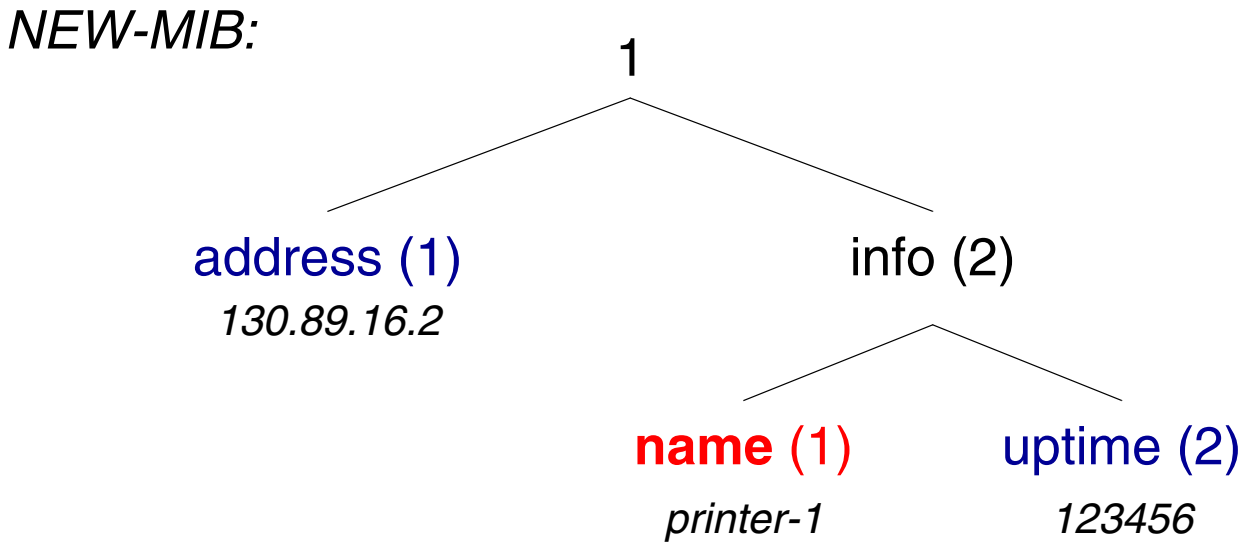
	SMIv1	SMIv2
SIMPLE TYPES:	INTEGER	INTEGER
	OCTET STRING	OCTET STRING
	OBJECT IDENTIFIER	OBJECT IDENTIFIER
	-	Integer32
APPLICATION-WIDE TYPES:	-	Unsigned32
	Gauge	Gauge32
	Counter	Counter32
	-	Counter64
	TimeTicks	TimeTicks
	IpAddress	IpAddress
	Opaque	Opaque
	NetworkAddress	-
PSEUDO TYPES:	-	BITS

EXAMPLE OF SCALAR OBJECTS



OBJECT NAMING

INTRODUCE NAMING TREE



THE LEAVES OF THE TREE REPRESENT THE MANAGED OBJECTS

NODES ARE INTRODUCED FOR NAMING PURPOSES

OBJECT NAMING

- address

Object ID = 1.1
Object Instance = 1.1.0
Value of Instance = *130.89.16.2*

- info

Object ID = 1.2

- name

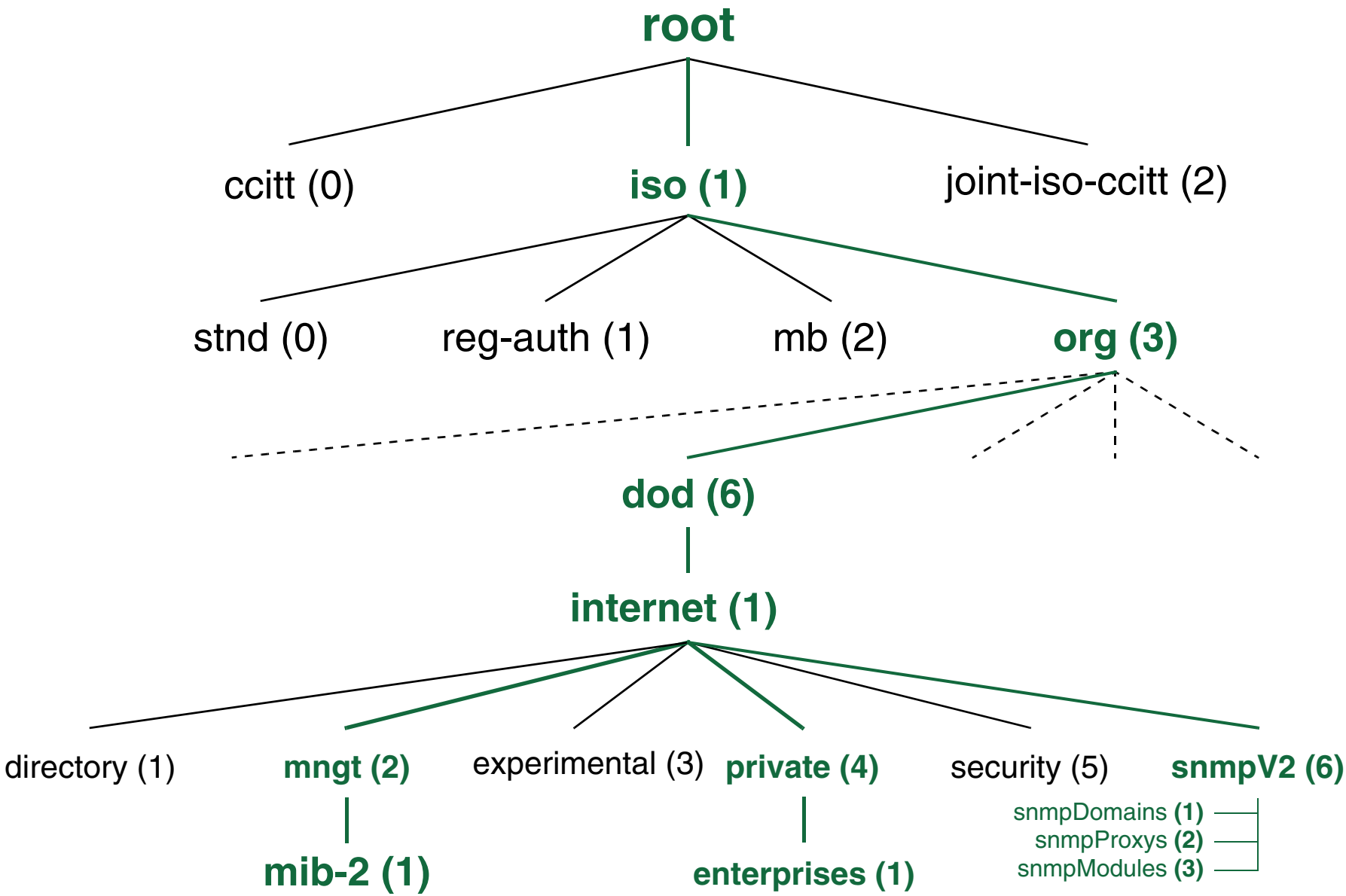
Object ID = 1.2.1
Object Instance = 1.2.1.0
Value of Instance = *printer-1*

- uptime

Object ID = 1.2.2
Object Instance = 1.2.2.0
Value of Instance = *123456*

ALTERNATIVE:
Object ID = NEW-MIB info uptime

OBJECT NAMING: MIBs



OBJECT TYPE DEFINITION

OBJECT-TYPE:

SYNTAX
- INTEGER
- OCTET STRING
- OBJECT IDENTIFIER
- BITS
- IpAddress
- Integer32
- Counter32
- Counter64
- Gauge32
- TimeTicks
- Opaque
- New Type

MAX-ACCESS
- read-only
- read-write
- read-create
- accessible-for-notify
- not-accessible

STATUS
- current
- deprecated
- obsolete

DESCRIPTION ""

OBJECT TYPE DEFINITION - EXAMPLE

```
-- Definition of address

address OBJECT-TYPE
SYNTAX          IpAddress
MAX-ACCESS      read-write
STATUS          current
DESCRIPTION     "The Internet address of this system"
::= {NEW-MIB  1}
```

DEFINITION OF NON-LEAF 'OBJECTS'

Name **OBJECT IDENTIFIER ::= {...}**

EXAMPLE:
info **OBJECT IDENTIFIER ::= {**NEW-MIB 2**}**

ALTERNATIVE CONSTRUCT: OBJECT IDENTITY

EXAMPLE:

```
info        OBJECT-IDENTITY
STATUS      current
DESCRIPTION "The node under which future scalar objects
             should be registered"
::= {NEW-MIB 2}
```

DEFINITION OF A MIB

NEW-MIB **DEFINITIONS ::=**
BEGIN

import statement(s)
module identity definition

definition of all node and leaf objects

definition of implementation requirements

END

MODULE IDENTITY - EXAMPLE

```
newMibModule MODULE-IDENTITY
        LAST-UPDATED "200104041200Z"
        ORGANIZATION "UT-ARCH"
        CONTACT-INFO "
                EWI-ARCH Group
                University of Twente
                POBox 217
                7500 AE Enschede
                The Netherlands
                Email: simpleweb@simpleweb.org "
        DESCRIPTION
                "Experimental MIB for demo purposes"
        ::= { enterprises ut(785) 7 }
```

IMPORT STATEMENT - EXAMPLE

IMPORTS

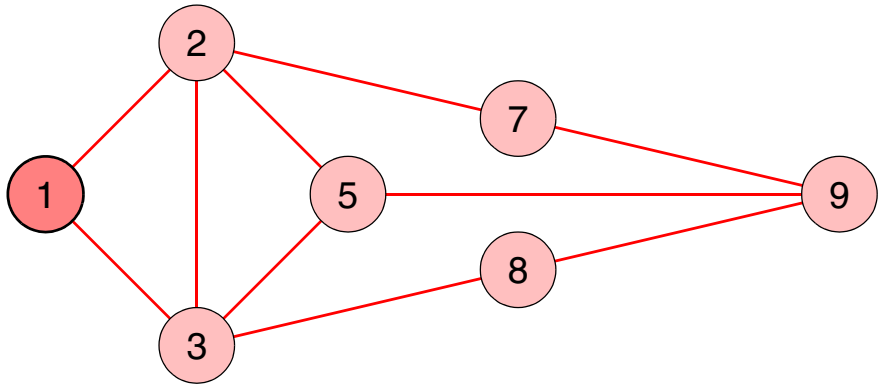
MODULE-IDENTITY, OBJECT-TYPE,
TimeTicks, enterprises

FROM SNMPv2-SMI;

TABLES

EXAMPLE: ROUTING TABLE

destination	next
2	2
3	3
5	2
7	2
8	3
9	3



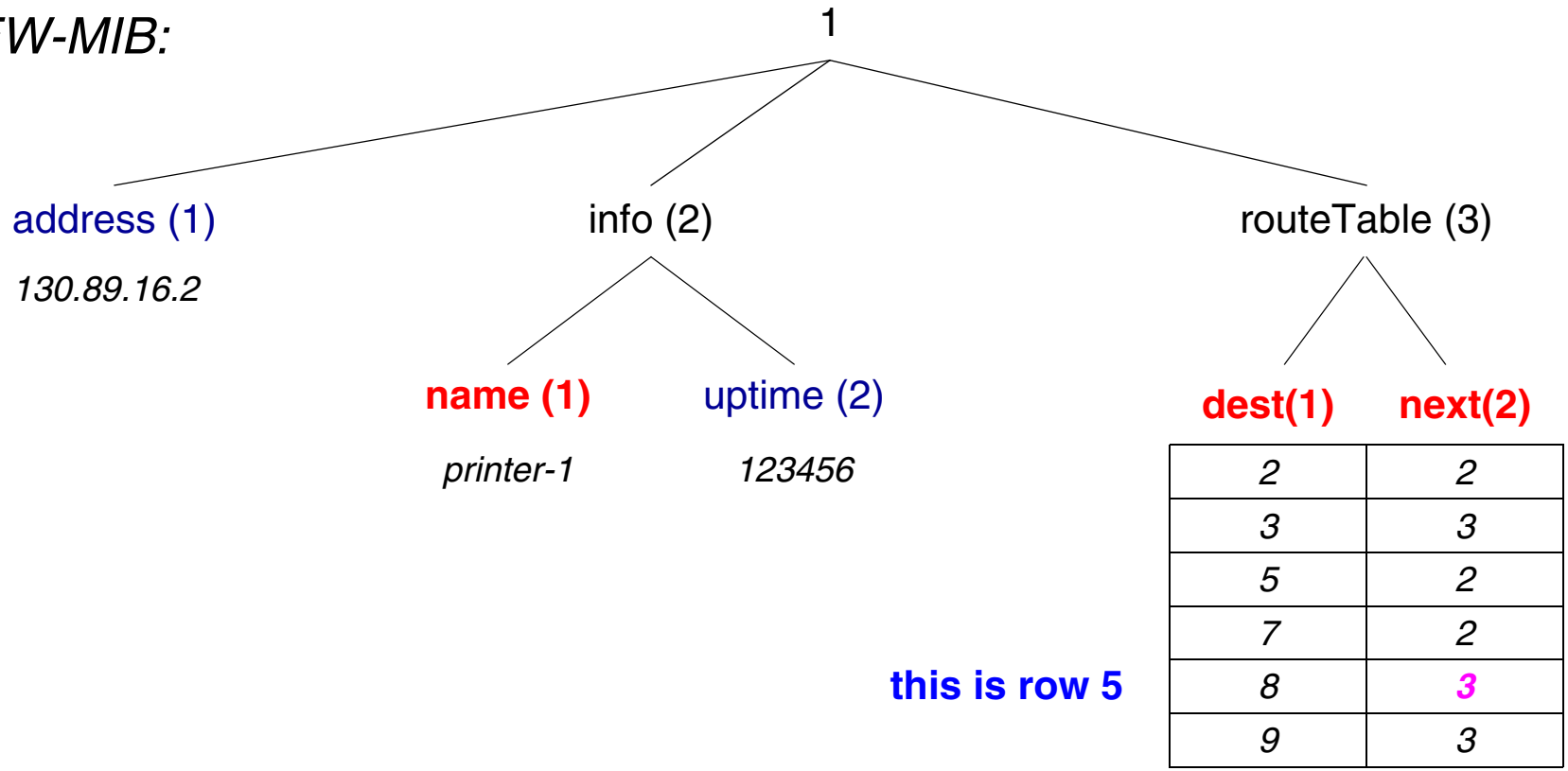
TO RETRIEVE INDIVIDUAL TABLE ENTRIES

EACH ENTRY SHOULD GET AN IDENTIFIER

NAMING OF TABLE ENTRIES - I

POSSIBILITY 1 (NOT BEING USED BY SNMP): USE ROW NUMBERS

NEW-MIB:



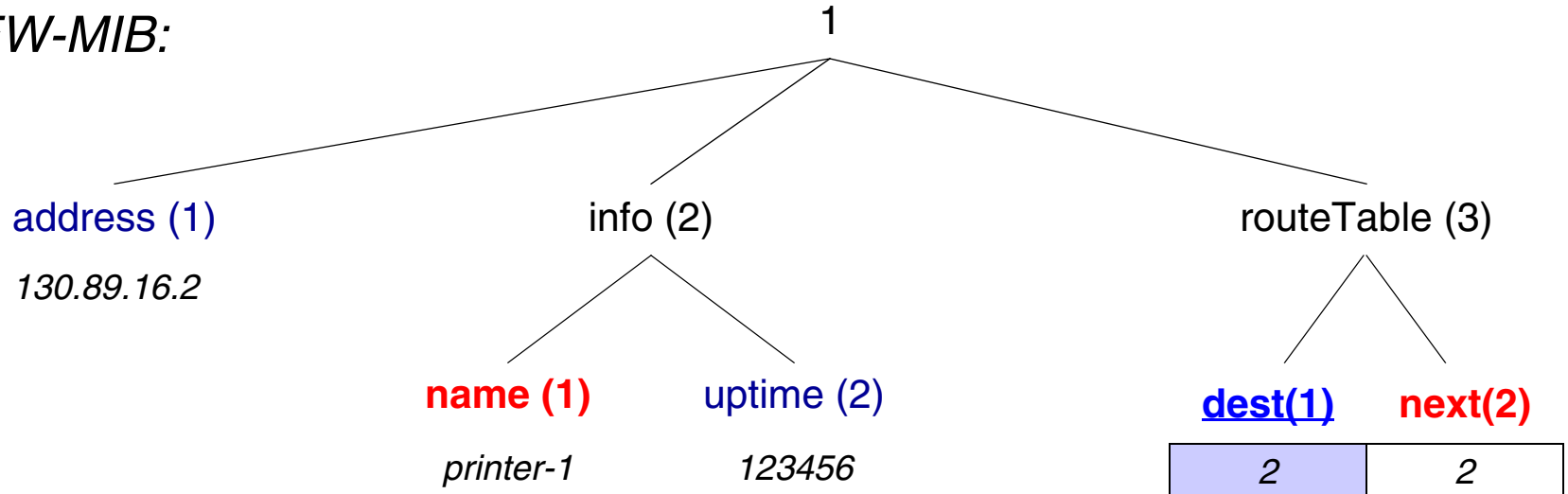
dest(1)	next(2)
2	2
3	3
5	2
7	2
8	3
9	3

EXAMPLE: THE VALUE OF *NEW-MIB routeTable next 5* IS *3*

NAMING OF TABLE ENTRIES - II

POSSIBILITY 2 (USED BY SNMP): INTRODUCE AN INDEX COLUMN

NEW-MIB:



dest(1)	next(2)
2	2
3	3
5	**2**
7	2
8	3
9	3

EXAMPLE: THE VALUE OF *NEW-MIB routeTable next 5* IS *2*

TABLE INDEXING

GENERAL SCHEME

X.C.I

OID of Table Column number Index value

EXAMPLES:

OID of Table = 1.3

1.3.1.5 => *5*
1.3.2.5 => *2*
1.3.1.9 => *9*
1.3.2.9 => *3*
1.3.2.7 => *2*
1.3.1.1 => *entry does not exist*
1.3.2.1 => *entry does not exist*

TABLE INDEXING - NON-INTEGER INDEX

AN INDEX NEED NOT BE AN INTEGER

routeTable (3)

dest (1)	next (2)
130.89.16.1	*130.89.16.1*
130.89.16.4	*130.89.16.4*
130.89.16.23	*130.89.16.1*
130.89.19.121	*130.89.16.1*
192.1.23.24	*130.89.16.4*
193.22.11.97	*130.89.16.4*

EXAMPLES:

OID of Table = 1.3

1.3.1.130.89.16.23 => *130.89.16.23*

1.3.2.130.89.16.23 => *130.89.16.1*

1.3.1.193.22.11.97 => *193.22.11.97*

1.3.2.193.22.11.97 => *130.89.16.4*

1.3.2.130.89.19.121 => *130.89.16.1*

TABLE INDEXING - MULTIPLE INDEX FIELDS

USE OF MULTIPLE INDEX FIELDS

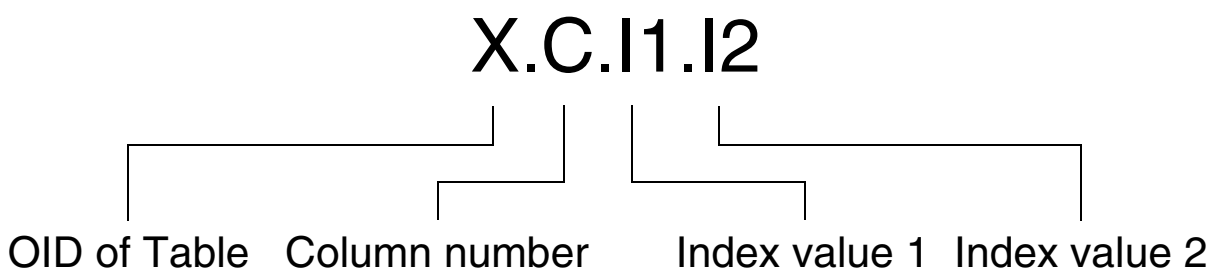


TABLE INDEXING - MULTIPLE INDEX FIELDS: EXAMPLE

EXAMPLE:

1 = low costs
2 = high reliability

routeTable (3)

dest (1)	**policy (2)**	**next (3)**
130.89.16.23	*1*	*130.89.16.23*
130.89.16.23	*2*	*130.89.16.23*
130.89.19.121	*1*	*130.89.16.1*
192.1.23.24	*1*	*130.89.16.1*
192.1.23.24	*2*	*130.89.16.4*
193.22.11.97	*1*	*130.89.16.1*

1.3.3.192.1.23.24.1 => *130.89.16.1*

1.3.3.192.1.23.24.2 => *130.89.16.4*

TABLE DEFINITION

-- Definition of the route table

```
routeTable      OBJECT-TYPE
SYNTAX          SEQUENCE OF RouteEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION     "This entity's routing table"
::= {NEW-MIB  3}

routeEntry      OBJECT-TYPE
SYNTAX          RouteEntry
MAX-ACCESS      not-accessible
STATUS          current
DESCRIPTION     "A route to a particular destination"
INDEX           {dest, policy}
::= {routeTable  1}
```

TABLE DEFINITION (cont. 1)

```
RouteEntry::=
SEQUENCE{
        dest  ipAddress,
        policyINTEGER,
        next  ipAddress
        }
```

TABLE DEFINITION (cont. 2)

this is the table

this is a row

this is a new type

```
routeTable  OBJECT-TYPE
SYNTAX  SEQUENCE OF RouteEntry
...

routeEntry  OBJECT-TYPE
SYNTAX  RouteEntry
...

RouteEntry::=
SEQUENCE
...
```

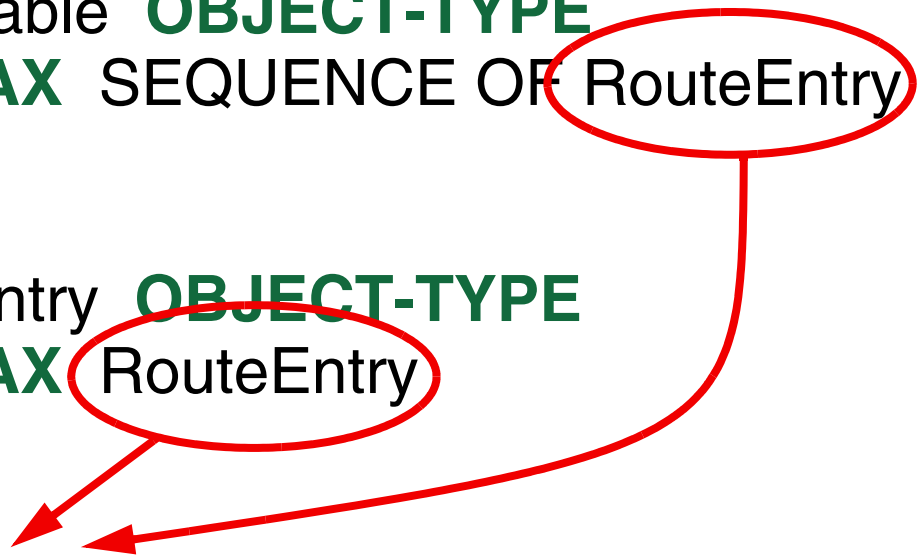


TABLE DEFINITION (cont. 3)

```
dest OBJECT-TYPE
SYNTAX      ipAddress
ACCESS      not-accesible
STATUS      current
DESCRIPTION"The address of a particular destination"
::= {routeEntry  1}

policy OBJECT-TYPE
SYNTAX      INTEGER {
             costs(1)  -- lowest delay
             reliability(2) }  -- highest reliability
ACCESS      not-accesible
STATUS      current
DESCRIPTION"The routing policy to reach that destination"
::= {routeEntry  2}

next OBJECT-TYPE
SYNTAX      ipAddress
ACCESS      read-write
STATUS      current
DESCRIPTION"The internet address of the next hop"
::= {routeEntry  3}
```

DEFINITION OF NEW TYPES

TEXTUAL CONVENTIONS

TO REFINE SEMANTICS OF EXISTING TYPES

EXAMPLE:

```
RunState ::= TEXTUAL CONVENTION
   STATUS current
   DESCRIPTION "..."
   SYNTAX INTEGER{
      running(1)
      runable(2)
      waiting(3)
      exiting(4)}
```

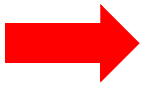
TEXTUAL CONVENTIONS

- PhysAddress
- MacAddress
- TruthValue
- AutonomousType
- InstancePointer
- VariablePointer
- RowPointer
- RowStatus
- TimeStamp
- TimeInterval
- DateAndTime
- StorageType
- TDomain
- TAddress

- Inet-Address...

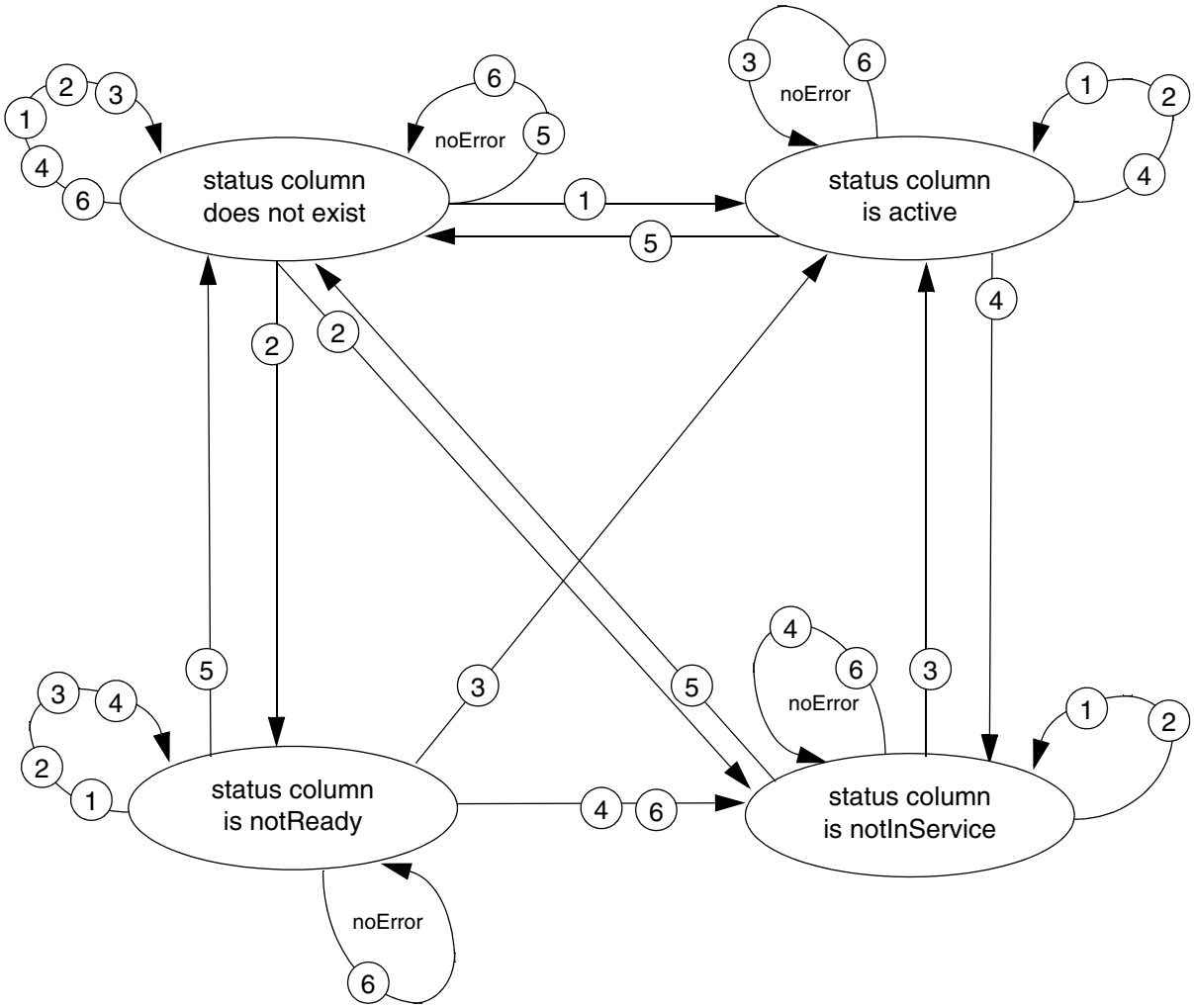
ROW-STATUS TEXTUAL CONVENTION

USED TO CHANGE TABLE ROWS



TO:	VIA:	STATUS:
130.89.16.4	130.89.1.1	ACTIVE
130.89.17.6	130.89.1.1	NOT READY
130.89.18.2	130.89.1.4	ACTIVE
130.89.18.7	130.89.1.4	ACTIVE

ROW-STATUS - STATE DIAGRAM



1	set status column to createAndGo
2	set status column to createAndWait
3	set status column to active
4	set status column to notInService
5	set status column to destroy
6	set any other column to some value

4 6 — 4 or 6

NOTIFICATION TYPES

SMIv2:

- MIBs MAY NOW INCLUDE NOTIFICATION TYPE MACROS

EXAMPLE:

```
linkUp NOTIFICATION-TYPE
   OBJECTS   {ifIndex}
   STATUS    current
   DESCRIPTION
      "A linkUp trap signifies that the
      entity has detected that the
      ifOperStatus object has changed to Up"
::= {snmpTraps 4}
```

DEFINITION OF IMPLEMENTATION REQUIREMENTS

THE MODULE-COMPLIANCE CONSTRUCT
DEFINES IMPLEMENTATION REQUIREMENTS FOR AGENTS

---

newMibCompliance **MODULE-COMPLIANCE**
STATUS ...
DESCRIPTION ...

MODULE 1	**MODULE** ... **MANDATORY-GROUPS** ... **GROUP** ... **OBJECT** ...
MODULE n	

::= { ... }

OBJECT GROUP CONSTRUCT

TO DEFINE A SET OF RELATED OBJECT TYPES

EXAMPLE:

```
newMibScalarGroup OBJECT-GROUP
   OBJECTS { address, name, uptime }
   STATUS current
   DESCRIPTION "The collection of scalar objects."
::= { demoGroups 1 }
```