Cap. 02 – Modelo de Informação

2.1 – Padrões de Gerência de Redes

2.2 – Arquitetura da Solucão SNMP

2.3 – Objetos, Instâncias e MIBs

2.4 – MIB-2

2.5 – Structure of Management Information v1

2.6 – Structure of Management Information v2

Referências Bibliográficas

- Jacques F. Sauvé - “Gerência de Redes de Computadores”, DSC, UFCG, 2002. Notas de Aula, ”http://www.dsc.ufcg.edu.br/~jacques”
- Alexandre Sztajnberg - “Conceitos Básicos sobre SNMP e CMIP”, COPPE-UFRJ, 1996, Relatório Técnico. http://www.gta.ufrj.br/alexszt/
- James F. Kurose; Keith W. Ross - “Redes de Computadores e a Internet”, Addison-Wesley, 3a Edição, ISBN-10: 8588639181 ou ISBN-13: 9788588639188

2.5 - Structure of Management Information v1

- MIB – exemplo:
  - RFC1213-MIB (MIB-2 de SNMPv1) - http://www.ietf.org/rfc/rfc1213.txt
  - SNMPv2-MIB (MIB de SNMPv2) - http://www.ietf.org/rfc/rfc1907

- Como descrever MIBs:
  - *Abstract Syntax Notation* – ASN.1 (OSI/ISO), não obstante, apenas um subconjunto de ASN.1 é utilizado na descrição das MIBs.
  - **facilitador** - Macros ASN.1 – simplificam a escrita de MIBs.

- Objetivo: descrição do SMI do SNMPv1 (RFC1155)
  - … na sequência as diferenças entre SMIv1 e SMIv2 (RFC1902)

… 2.5 - Structure of Management Information v1

- SNMPv1 (RFC1155) - http://www.ietf.org/rfc/rfc1155.txt
  - SMIv2 (RFC1902) - http://www.ietf.org/rfc/rfc1902

- SMI descreve:
  - como a informação de gerência é agrupada e nomeada;
  - quais operações e tipos de dados são permitidos;
  - sintaxe para descrever as MIBs;
    - Obs.: MIB não especifica como os recursos são implementados.
  - alguns objetos só têm uma instância e outros têm várias instâncias;
  - tabelas são usadas para agrupar objetos semelhantes;
  - identidade de um objeto e uma instância formam uma variável SNMP.

… 2.5 - Structure of Management Information v1

- SMI – *meta-schema* do banco de dados:
  - … abordagem semelhante ao modelo relacional (modelo de dados que se baseia no princípio em que todos os dados estão guardados em tabelas ou relações - baseada na lógica de predicados e na teoria dos conjuntos).

2.5.1 – Identificadores de Objetos (OIDs)

- Object Identifiers (OIDs) – identificação unívoca dos objetos:
  - sequência de inteiros não negativos;
  - organizados hierarquicamente;
  - e únicos no tempo e espaço.

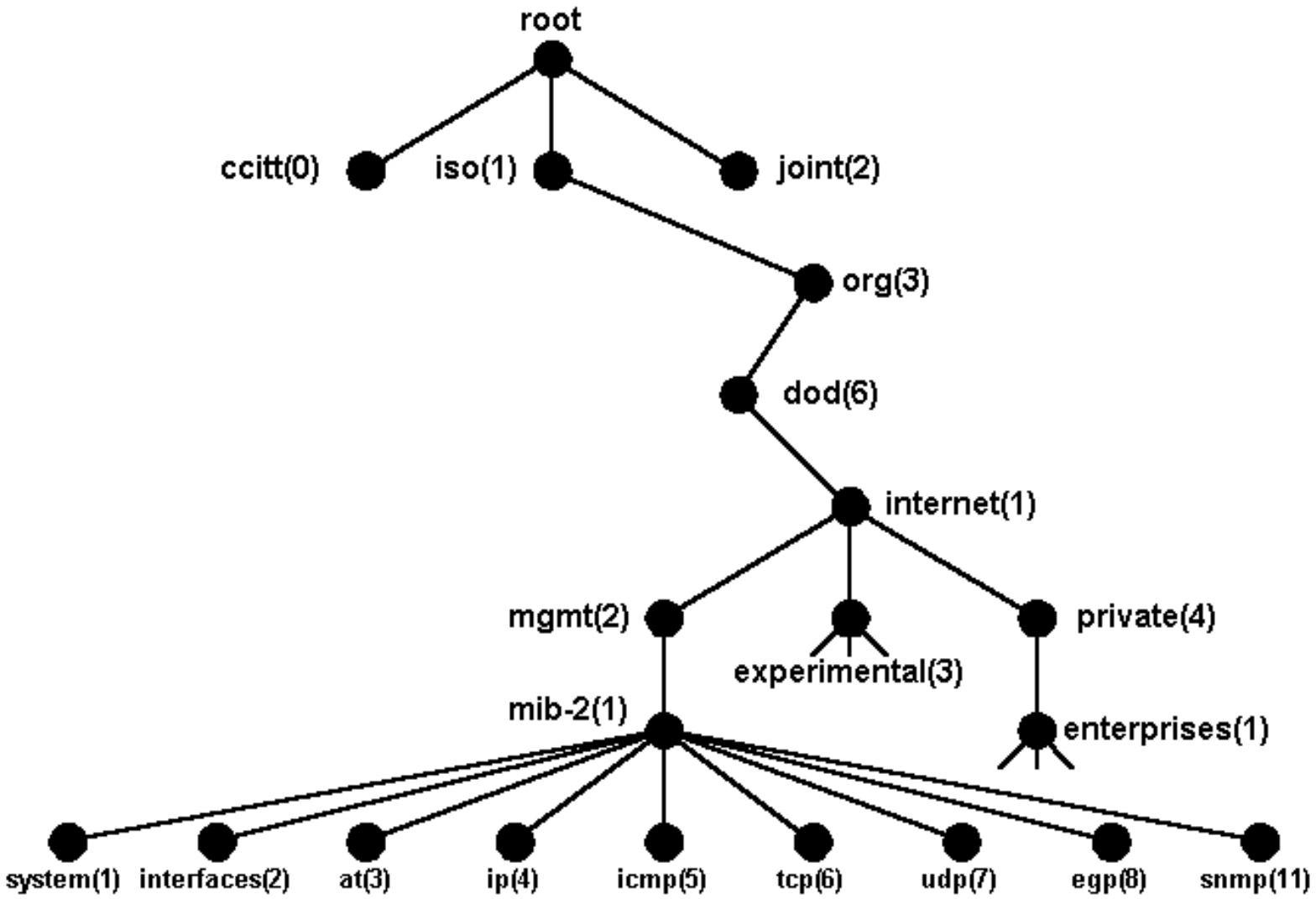
- Para facilitar a leitura, um nome textual é normalmente associado a cada componente da sequência do OID:
  - último nome é a forma curta de referenciá-lo;
  - unicidade global através de uso de prefixos (sys, if, tcp) – e.g., **sys**Descr

… 2.5.1 – Identificadores de Objetos (OIDs)

- Sintaxe para especificar o valor do OID;
  - { iso(1) org(3) dod(6) internet(1) } ou 1.3.6.1
  - { internet 4 } ou 1.3.6.1.4
  - { tcp 4 } ou 1.3.6.1.2.1.6.4
  - primeiro Nome – deve ser único no contexto em que é utilizado;
  - não há forma curta quando se utiliza notação numérica.

- OIDs podem representar “qualquer coisa” -
  - e.g., Identificador de Produto - “sysObjectID”

… 2.5.1 – Identificadores de Objetos (OIDs)



… 2.5.1 – Identificadores de Objetos (OIDs)

- OIDs de destaque:

  - MIB-2 (Mar/1991 – SNMPv2) - http://www.ietf.org/rfc/rfc1213.txt
  - “experimental”: usado para experiências ainda não conclusivas dos grupos de trabalho da IETF (*Internet Engineering Task Force*);
  - “enterprise”: empresas definem suas MIBs proprietárias:
    - números de empresas podem ser obtidas da IANA;
    - empresa decide o que fazer na sua árvore.

2.5.2 – Módulos MIBs

- MIB SNMP – definida por um conjunto de módulos, ou seja, consiste de várias MIBs definidas em vários documentos:
  - **Obs.:** “MIB" é normalmente utilizada para descrever uma MIB específica e não uma MIB conceitual global.
- Módulo – define escopo de nomes (validade):
  - nomes devem ser únicos no módulo e únicos globalmente (MIB Padrão);
  - … assim, se nomes de módulos forem únicos e nomes internos a um módulo também, não teremos conflitos - desejável!
- Regras de formação de Nomes de Módulos:
  - começam com letra maiúscula;
  - composto de letras, números e hífen, mas não terminados em hífen;
  - … ...

… 2.5.2 – Módulos MIBs

- Regras de formação de Nomes de Módulos:
  - …
  - hífen não pode ser seguido de hífen;
  - sem restrição de tamanho, embora “softwares” e compiladores de NMS (*Network Management Systems*) possam impor restrições.

- Informações contidas em um Módulo:
  - pontos de registro na árvore de OIDs;
  - objetos gerenciados;
  - valores para OIDs;
  - “traps”;
  - sequências (para definir tabelas);
  - convenções textuais.

… 2.5.2 – Módulos MIBs

- Módulos contém:
  - pontos de registro na árvore de OIDs;
  - objetos gerenciados e valores (intâncias) para OIDs;
  - “traps”;
  - sequências para definir tabelas;
  - convenções textuais.

- Módulos não contém:
  - novas macros em ASN.1;
  - novos tipos etiquetados (tipo novo não codificado em BER) em ASN.1, que podem ser definidos em SMI mas nunca na MIB.

… 2.5.2 – Módulos MIBs

- Sintaxe da definição de um Módulo:

  - <mib> = <module> …

  - <module> = <ModName> "DEFINITIONS" "::=" "BEGIN"
    ["IMPORTS" <importList> ... ";"]
    [<smiItem>...]
    [<textConvItem>...]
    { <oidItem> | <objectItem> | <seqItem> | <trapItem> } ...
    "END"

  - <importList> = <importItem> ["," <importItem>] … "FROM" <ImportModName>

… 2.5.2 – Módulos MIBs

- Onde:
  - <ModName> - nome de um módulo MIB;
  - <smiItem> - definição de item SMI tais como tipos sintáticos, as macros OBJECT-TYPE e TRAP-TYPE;
  - <importItem> - item definido em outro módulo MIB;
  - <ImportModName> - nome de um outro módulo MIB previamente definido;
  - <textConvItem> - definição da convenção textual;
  - <oidItem> - definição de um “object identifier”;
  - <objectItem> - definição de um item com a macro OBJECT-TYPE;
  - <seqItem> - definição de uma sequência; e
  - <trapItem> - definição de um item com a macro TRAP-TYPE.

… 2.5.2 – Módulos MIBs

- Sintaxe de definição de um Módulo:
  - nomes podem ser importados de módulos anteriormente definidos;
  - convenções textuais constituem um mecanismo para estender tipos sintáticos (vantagem: não adicionam novo tipo).
  - identificadores de objetos são: pontos de registro na árvore de OIDs; valores de itens com a sintaxe OIDs e grupos de objetos SNMP.
  - macro “OBJECT-TYPE” é utilizada para definir:
    - tabelas, linhas, objetos simples e colunares.
  - sequências são utilizadas pra definir tabelas conceiturais;
  - macro “TRAP-TYPE” é utilizada para definir “TRAPS”;
  - “coments” – iniciam com “--” e terminam com “--” ou no fim da linha.

2.5.3 – Especificação de um Módulo

- Posição de um objeto na árvore de OIDs deve ser determinada antes de se escrever um Módulo MIB;
  - … pois ao publicar um MIB, itens não podem ser alterados;
  - podem ficar obsoletos ou serem recriados com outros OIDs;
  - … ou empresas podem definir um novo ramo “experimental” e mudar as MIBs até que sejam publicadas em outro local da árvore.
- Formato básico do Módulo:
  - um nome único é escolhido para o módulo;
  - importações são feitas para:
    - os pontos de registro dos objetos;
    - os tipos sintáticos usados no módulo;
    - para as macros “OBJECT-TYPE” e “TRAP-TYPE”.

… 2.5.3 – Especificação de um Módulo

- EXAMPLE-MIB DEFINITIONS ::= BEGIN
- -- A MIB is always written in English
- -- Copyright notice
- -- MIB Descriptions
- -- Some or all the following IMPORTS ...
- IMPORTS
- enterprises, Counter, Gauge, TimeTicks, IpAddress FROM RFC1155-SMI
- DisplayString, PhysAddress FROM RFC1213-MIB
- OBJECT-TYPE FROM RFC-1212
- TRAP-TYPE FROM RFC-1515;
- -- Some or all of the following:
- -- Textual Conventions
- -- Registration points
- -- Groups
- -- SNMP managed Objects
- -- SNMP traps
- END

2.5.4 – Definição de Objetos Gerenciados

- “Object Identifier” é utilizado para:
  - pontos de registro de alto nível na árvore;
  - grupos de objetos;
  - valor para itens com a sintaxe “Object Identifier”.

- Regra para nomes de OIDs – mesma que para os módulos, exceto que iniciados por letra minúscula.

… 2.5.4 – Definição de Objetos Gerenciados

- Grupos de “Object Identifiers” (organizam objetos em árvores):
  - são utilizados como unidade de conformidade;
  - grupos podem ser de implementação obrigatória ou opcional;
    - se obrigatórios, todos os sub-objetos devem ser suportados;
    - se opcionais, podem ser implementados completamente ou não implementados.

- e.g., system OBJECT IDENTIFIER ::= { mib-2 1 }

… 2.5.4 – Definição de Objetos Gerenciados

- 03 tipos de objetos podem ser definidos utilizando-se a macro “OBJECT-TYPE”:
  - tabelas;
  - linhas de tabelas;
  - objetos folha (simples ou colunares).

- <objectItem> = <objectName> "OBJECT-TYPE"
- "SYNTAX" <syntax>
- "ACCESS" <access>
- "STATUS" <status>
- ["DESCRIPTION" <description>]
- ["REFERENCE" <reference>]
- ["INDEX" "{" <indexItems>"}"]
- ["DEFVAL" "{" <defaultValue>"}"]
- "::=" "{" <parent> <number>"}"
- … …

… 2.5.4 – Definição de Objetos Gerenciados

- … …
- "::=" "{" <parent> <number>"}"

- <objectName> = <tableName> | <rowName> | <leafName>
- <syntax> = { "SEQUENCE" "OF" <SequenceName> } | <SequenceName> |
- <leafSyntax>
-
- Onde:
-
- <objectName> - nome do item sendo definido;
- <parent> - nome do item que contém este nó na árvore de OIDs;
- <number> - valor do último componente do item sendo definido; e
- Valores para <access>, <status>, <leafSyntax>, etc.

… 2.5.4 – Definição de Objetos Gerenciados

- Objetos do tipo Tabela (consiste de linhas):
  - SNMP não possibilita a manipulação de tabelas ou linhas, apenas itens individuais das colunas (itens colunares) – tabelas são conceituais.
  - sintaxe: “sequence of <sequência>”
  - por convenção, nome da tabela usa o sufixo “Table” ;
  - por convenção, o nome da sequência associada é o prefixo do nome da tabela (sem “Table”) iniciado com letra maiúscula (identificando um tipo) e o sufixo “Entry” - e.g., para a tabela “ifTable a sequência é ifEntry”;
  - o valor deve ser “not-accessible”, posto que o SNMP não manipula tabelas diretamente.

… 2.5.4 – Definição de Objetos Gerenciados

- <tableName> "OBJECT-TYPE"
- "SYNTAX" "SEQUENCE" "OF" <SequenceName>
- "ACCESS" "not-accessible"
- "STATUS" <status>
- ["DESCRIPTION" <description>]
- ["REFERENCE" <reference>]
- "::=" "{" <parent> <number>"}"
-
- <tableName> = <name>"Table"
- <SequenceName> = <Name>"Entry"
-
- Onde:
-
- <name> - prefixo da tabela sendo definida;
- <Name> - prefixo da sequência associada;
- <parent> - nome do item que contem este nó diretamente na árvore de OIDs;
- <number> - valor do último componente do item sendo definido; e
- Valores para <status>, <description>, etc.

… 2.5.4 – Definição de Objetos Gerenciados

- e.g., Objeto do tipo Tabela:

  - ifTable OBJECT-TYPE
  - SYNTAX SEQUENCE OF IfEntry
  - ACCESS not-accessible
  - STATUS mandatory
  - ::= { interfaces 2 }

… 2.5.4 – Definição de Objetos Gerenciados

- Objetos do tipo Linha (que consiste em colunas):
  - SNMP não manipula colunas diretamente, entretanto, “GET” e “GET-NEXt” podem ser usados para acessar todas as colunas de uma determinada linha de uma única vez;
  - por convenção, o nome da linha é o nome da tabela com “Table” substituido por “Entry”;
  - o tipo sintático da linha deve ser a sequência usada para definir a tabela;
  - o valor de OID para o objeto linha deve ser o OID da tabela + 1;
  - a cláusula “index” especifica como identificar instâncias da linha unicamente;

… 2.5.4 – Definição de Objetos Gerenciados

- <rowName> "OBJECT-TYPE"
- "SYNTAX" <SequenceName>
- "ACCESS" "not-accessible"
- "STATUS" <status>
- ["DESCRIPTION" <description>]
- ["REFERENCE" <reference>]
- "INDEX" "{" <indexItems> "}"
- "::=" "{" <tableName> 1 "}"
- <rowName> = <name>"Entry"
- <SequenceName> = <Name>"Entry"
- <tableName> = <name>"Table"
- <indexItems> = <indexItem> [","<indexItem>] ...
-
- Onde: <name> - prefixo da linha sendo definida e o prefixo da tabela associada;
- <Name> - prefixo da sequência associada;
- <indexItem> - nome de um item na sequência para a linha;
- Valores para <status>, <description>, etc. serão

… 2.5.4 – Definição de Objetos Gerenciados

- e.g. de Objeto do tipo Linha:

  - ifEntry OBJECT-TYPE
  - SYNTAX IfEntry
  - ACCESS not-accessible
  - STATUS mandatory
  - INDEX { ifIndex }
  - ::= { ifTable 1 }

… 2.5.4 – Definição de Objetos Gerenciados

- Definições de Sequência (define as colunas da linha):
  - nome da sequência inicia com letra maiúscula;
  - itens da sequência, normalmente, são filhos do objeto linha, entretanto, quando se estende uma tabela, alguns dos objetos colunares da tabela existente podem ser adicionados à nova sequência;
  - além do nome, a sintaxe de cada objeto colunar deve ser definida e, normalmente, é um resumo da sintaxe total do item;
  - uma sequência pode ser usada apenas para uma tabela e uma linha;
  - uma sequência não pode ser importada de outro módulo.

… 2.5.4 – Definição de Objetos Gerenciados

- <seqItem> = <SequenceName> "::=""SEQUENCE" "{"
- <columnItem> <leafSyntax>
- { "," <columnItem> <leafSyntax> }...
- "}"
- <SyntaxName> = <Name>"Entry"
-
- Onde:
-
- <Name> - prefixo da sequência sendo definida;
- <columnItem> - nome de um item da sequência; e
- <leafSyntax> - sintaxe simplificada do item.

… 2.5.4 – Definição de Objetos Gerenciados

- e.g., Objeto do tipo SEQUENCE:
  - IpAddrEntry ::= SEQUENCE {
  - ipAdEntAddr IpAddress,
  - ipAdEntIfIndex INTEGER,
  - ipAdEntNetMask IpAddress,
  - ipAdEntBcastAddr INTEGER,
  - ipAdEntReasmMaxSize INTEGER
  - }

… 2.5.4 – Definição de Objetos Gerenciados

- Objeto Folha (menor objeto de agrupamento):
  - um objeto folha mais uma distância formam uma variável SNMP, ou seja, variáveis são operandos do protocolo;
  - podem ser simples:
    - e.g., número de interfaces de um dispositivo;
    - só tem uma distância;
    - instância tem OID com componente final 0;
    - objetos simples normalmente têm o mesmo prefixo do grupo ao qual pertencem.
  - podem ser colunares:
    - organizados em tabelas conceituais;
    - pode haver várias instâncias, e.g., velocidade de uma interface de rede;
    - instâncias são formadas através do OID do objeto colunar mais o valor da cláusula 'index' da linha correspondente
    - objetos colunares tem, normalmente, o mesmo prefixo da linha.

… 2.5.4 – Definição de Objetos Gerenciados

- <leafName> "OBJECT-TYPE"
- "SYNTAX" <leafSyntax>
- "ACCESS" <access>
- "STATUS" <status>
- ["DESCRIPTION" <description>]
- ["REFERENCE" <reference>]
- ["DEFVAL" "{" <defaultValue>"}"]
- "::=" "{" <parent> <number>"}"
-
- Onde:
-
- <leafName> - nome do objeto folha sendo definido;
- <parent> - nome do item que contém este nó na árvore de OIDs (linha ou um grupo);
- <number> - valor do último componente do item sendo definido; e
- Valores para <leafSyntax>, <access>, <status>, etc.

… 2.5.4 – Definição de Objetos Gerenciados

- e.g., Objeto Folha:
  - sysUpTime OBJECT-TYPE
  - SYNTAX TimeTicks
  - ACCESS read-only
  - STATUS mandatory
  - ::= { system 2 }

  - ipAdEntAddr OBJECT-TYPE
  - SYNTAX IpAddress
  - ACCESS read-only
  - STATUS mandatory
  - ::= { ipAddrEntry 1 }

… 2.5.4 – Definição de Objetos Gerenciados

- Convenções Textuais:
  - usadas para especificar semântica adicional a um tipo sintático existente, possibilitando a extensão de um tipo sintático do SMI;
  - 02 convenções textuais são definidas:
    - DisplayString (caracteres ASCII imprimíveis);
    - PhysAddress;
    - Obs.: ambas são baseadas em “octet string” não sendo necessária nova codificação.
  - restrições são descritas em comentários anexados à convenção textual.

… 2.5.4 – Definição de Objetos Gerenciados

- <textConvItem> = <textConvName> "::=" <leafSyntax>

- Onde:
  - <textConvName> - nome da convenção textual sendo definida; e
  - <leafSyntax> é qualquer tipo sintático.

- Exemplos:
  - DisplayString ::= OCTET STRING (SIZE(0..255))
  - Status ::= INTEGER { enabled(1), disabled(2) }

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax:
  - Tabelas - SEQUENCE OF <NomeDeSequência>
  - Objetos do tipo Linha - <NomeDeSequência>
  - Para Folhas, não pode usar uma SEQUÊNCIA, pois embora ASN.1 permita, não é permitido em SMI.

- <leafSyntax> = { "INTEGER" [<range> | [<enums>] } |
- { "OCTET" "STRING" [<size>] } | { "OBJECT" "IDENTIFIER" } |
- { <smiApplType> } | { <textConvName> [<range> | <size>] }
- <smiApplType> = "NetworkAddress" | "IpAddress" | "Counter" | "Gauge" |
- "TimeTicks" | "Opaque"

… 2.5.4 – Definição de Objetos Gerenciados

- <leafSyntax> = { "INTEGER" [<range> | <enums>] } |
- { "OCTET" "STRING" [<size>] } | { "OBJECT" "IDENTIFIER" } |
- { <smiApplType> } | { <textConvName> [<range> | <size>] }
- <smiApplType> = "NetworkAddress" | "IpAddress" | "Counter" | "Gauge" |
- "TimeTicks" | "Opaque"
- <range> = "(" <lower> ".." <higher> ")"
- <enums> = "{" <enumItem> ["," <enumItem>]... "}"
- <enumItem> = <enumName> "(" <enumValue> ")"
- <size> = "(" "SIZE" "(" <smallest> [".." <largest>] ")" ")"

Onde:

- <lower> and <higher> inteiros + ou -, strings de bits constantes, ou constantes hexstring;
- <enumName> - inicia com letra minuscula seguida de um número arbitrário de letras, dígitos, e hifen.
- <enumValue> - inteiro positivo, constante bitstring ou hexstring que não tenha valor zero
- <smallest> e <largest> - inteiros não negativos, constantes bitstring ou hexstring.

… 2.5.4 – Definição de Objetos Gerenciados

- e.g, Inteiro:
  - 32 BITS;
  - podem especificar uma faixa.

- SYNTAX INTEGER (0..65535)
- SYNTAX INTEGER (0..'ffff'h)
- SYNTAX INTEGER (0..'ff'H)

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - <enums>:
  - caso especial para INTEGER;
  - não pode usar zero ou negativo;
  - variáveis só podem ter os valores especificados;
  - deveria ter uma opção “other” mas muitas MIBs não usam;
  - descrição deveria explicar valores não óbvios.

- SYNTAX INTEGER { gateway(1), host(2) }
- SYNTAX INTEGER { other(1), invalid(2), direct(3), indirect(4) }

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - OCTET STRING:
  - “string” de bytes com informação binária;
  - pode ter um tamanho.

- SYNTAX OCTET STRING (SIZE (0..9)) tamanho variável
- SYNTAX OCTET STRING tamanho variável
- SYNTAX OCTET STRING (SIZE (6)) tamanho fixo

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - DisplayString:
  - convenção textual;
  - é um OCTET STRING com caracteres ASC-II imprimíveis;
  - deve ter uma tamanho mas é frequentemente omitido.

- SYNTAX DisplayString (SIZE (0..255))

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - PhysAddress:
  - convenção textual;
  - é um OCTET STRING onde os bytes representam um endereço físico em “Network Order” (Big-Endian).

- SYNTAX PhysAddress (SIZE (6))

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - OBJECT IDENTIFIER:
  - um valor de OID.

- SYNTAX OBJECT IDENTIFIER

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - NetworkAddress:
  - só endereços IP são suportados;
  - tipo descontinuado – não use !!

- Valores para Syntax - IpAddress:
  - OCTET STRING de 4 bytes em “Network Order”.

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - Counter:
  - inteiro não negativo que conta até $2^{32} – 1$ e volta para zero;
  - não pode ter faixa;
  - valor absoluto não significa nada.

- Valores para Syntax - Gauge:
  - inteiro não negativo que pode aumentar ou diminuir mas retorna a $2^{32} – 1$ como valor máximo;
  - não pode ter faixa;
  - e.g., ifOutQLen
-

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Syntax - TimeTick:
  - inteiro não negativo que conta centésimos de segundos desde um marco de tempo (EPOCH) - especificado na descrição;
  - não pode ter faixa.

- Valores para Syntax - Opaque:
  - usados em MIBS privadas para representar um valor qualquer quando não tem outro disponível;
  - não deve ser usado mais (*deprecated*).

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Acesso – ACCESS:
  - read-only
  - read-write
  - write-only
  - not-accessible
  - necessário para tabelas e linhas.

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Status - STATUS:
  - “mandatory” – se uma implementação não suporta o objeto, ela é “non-conformant” para esta MIB;
  - “optional” – não deve ser usado (SNMP admite que foi um ERRO!);
    - grupos inteiros podem ser opcionais mas não objetos individuais.
  - “obsolete” – indica que o objeto não deve ser usado mais;
  - “deprecated” – indica que o objeto pode ser usado, mas vai desaparecer com o tempo.

… 2.5.4 – Definição de Objetos Gerenciados

- Valores para Descrição – DESCRIPTION:
  - STRING entre aspas duplas (pode ter várias linhas);
  - descreve a semântica do objeto;
  - ajuda os escritores de gerentes e agentes;
  - serve de ajuda para o operador quando faz MIB “Browsing”

- Valores de Referência – REFERENCE:
  - STRING de descrição para apontar um outro lugar (objeto, documento) que descreve o mesmo objeto ou do qual o objeto depende.

… 2.5.4 – Definição de Objetos Gerenciados

- Valores de Índice – INDEX:
  - somente para objeto do tipo linha, e responsável pela listagem das colunas que servem de especificadores de instâncias para objetos colunares;
  - ordem dos itens indica como a instância é montada.
  - descrição deve informar a semântica dos itens do INDEX.

- Valores para Definição de Valor - DEFVAL:
  - usado com dica para a implementação de agentes para a criação de linhas usando “SET” e quando o “SET” não especifica todos os valores das colunas
  - só deve ser usado para objetos colunares que não participam do INDEX.

2.5.5 – Regras para Definição de Objetos

- Objetivo: colocar objetos em grupos lógicos hierárquicos e, neste sentido, o grupo completo pode ser designado como OPCIONAL ou OBRIGATÓRIO;
- Componente zero (0) não pode ser usado para um objeto;
- OID de um objeto do tipo linha para uma tabela deve estar um nível abaixo da tabela e deve ter o último componente igual a 1:
  - não deve haver irmãos para este objeto linha;
  - os objetos colunares devem estar um nível abaixo do objeto linha.

… 2.5.5 – Regras para Definição de Objetos

- No SNMP, objetos agregados são definidos como tabelas:
  - uma ou mais colunas são designadas como índices das linhas;
  - não pode ter tabelas dentro de tabelas;
  - para simular tabelas aninhadas, eleve o nível da outra tabela ao nível da primeira, e colunas que são índices da tabela original podem ser adicionadas à tabela nova com outro nome
  - … os índices da nova tabela serão os índices da tabela original (renomeados) mais os índices naturais da nova tabela.

… 2.5.5 – Regras para Definição de Objetos

- Tabelas que permitem a criação e remoção de linhas devem ter uma coluna xxxType ou xxxStatus que é uma enumeração:
  - por convenção, o primeiro valor deve ser “other” ou “valid” e o segundo valor deveria ser “invalid”;
  - para remover uma linha, colocar “invalid” nesta variável;
  - uma nova linha é criada com uma única operação SET
    - as variáveis do set são as colunas obrigatórias.
- Cláusula DESCRIPTION deve descrever a semântica da criação e remoção para cada objeto folha (descrição da função e uso).

… 2.5.5 – Regras para Definição de Objetos

- Todos os nomes de um módulo MIB devem ser únicos:
  - nomes de objetos iniciam com letra minúscula;
  - nomes de objetos que são contadores devem terminar com "s" (plural).

- Objetos que são strings imprimíveis (ASC-II) devem usar a convenção textual “DisplayString”:
  - objetos que contêm informação binária devem usar OCTET STRING.

- Endereço físicos devem usar “PhysAddress” e não OCTET STRING

… 2.5.5 – Regras para Definição de Objetos

- Objetos devem ser projetados para a idempotência:
  - SNMP usa UDP e o gerente pode duplicar o pedido se não houver resposta após um *timeout*
  - *… em matemática, um objeto é idempotente quando ele é o resultado de sua composição consigo mesmo, no sentido que se torna mais preciso;*
  - *e.g.: os únicos números reais idempotentes em relação à multiplicação são 0 e 1 (“a * a = a” ou “f o f = f”)*
- Número de colunas de uma tabela deve ser pequeno o suficiente para que uma linha inteira possa ser recuperada com uma única operação GET.

... 2.5.5 – Regras Gerais para Definição de Objetos

- Informação demais cria tanto problema quanto informação insuficiente, assim, inicie lentamente e só inclua objetos importantes para a gerência;
- Inicie com os objetos que são importantes para a gerência de configuração e faltas (as duas áreas mais importantes)
- Lembre que a segurança do SNMPv1 e SNMPv2 é fraca:
  - não dependa demais do controle remoto usando SET.
- Não coloque objetos para "guardar lugar" para adições futuras:
  - cada objeto deve ser usado de fato.
- Evite redundância:
  - não defina objetos que podem ser calculados com o valor de outros.

… 2.5.5 – Regras Gerais para Definição de Objetos

- Use diagramas CASE (Jeffrey Case – 1989) - (vide à frente) para mostrar a relação entre contadores.
- Escolha objetos genéricos que poderão ser usados em outros produtos.
- Seções críticas de código não devem ser instrumentadas demais.
- Após colocar instrumentação SNMP num dispositivo, este deve ainda funcionar bem no seu papel original.

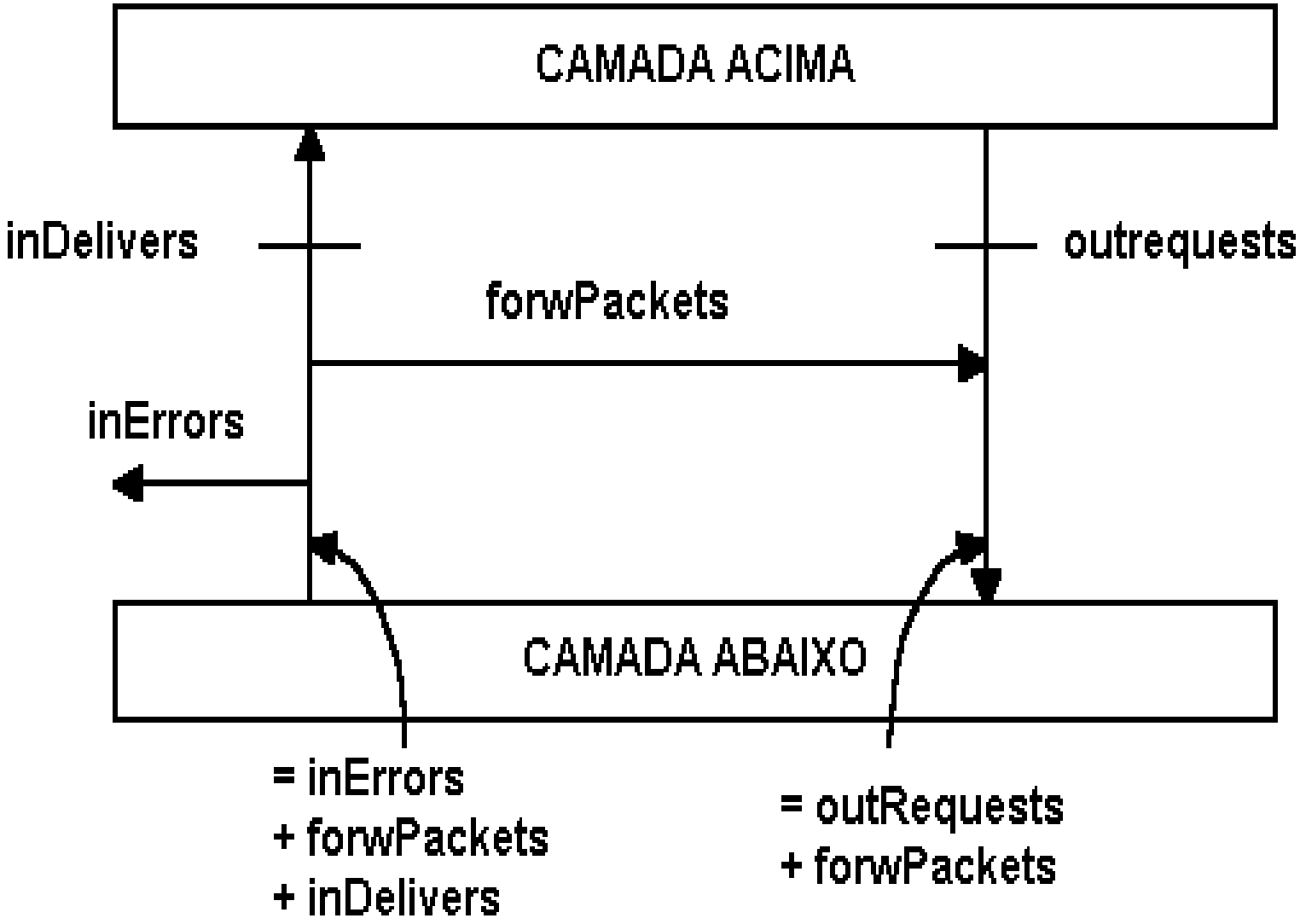
2.5.6 – Diagramas CASE

- Proposto por Jeffrey Case em 1989:
  - Idéia: para grupos de MIBs, faz-se necessário conter o padrão de tráfego em uma particular camada de protocolo;
  - … deve se garantir que cada PDU recebida/transmitida em/por uma camada deve ser contabilizada na mesma incluindo PDUs inválidas ou com erros.
- Neste sentido “Diagramas Case” podem ser utilizados para descrever o fluxo dos pacotes dentro de camadas individuais.
- Ref. Bibliográfica: J. Won-Ki Hong - “CASE Diagrams & SNMP Standard MIBs” - Dept. of Computer Science and Engineering; POSTECH DP&NM Lab; jwkhong@postech.ac.kr.

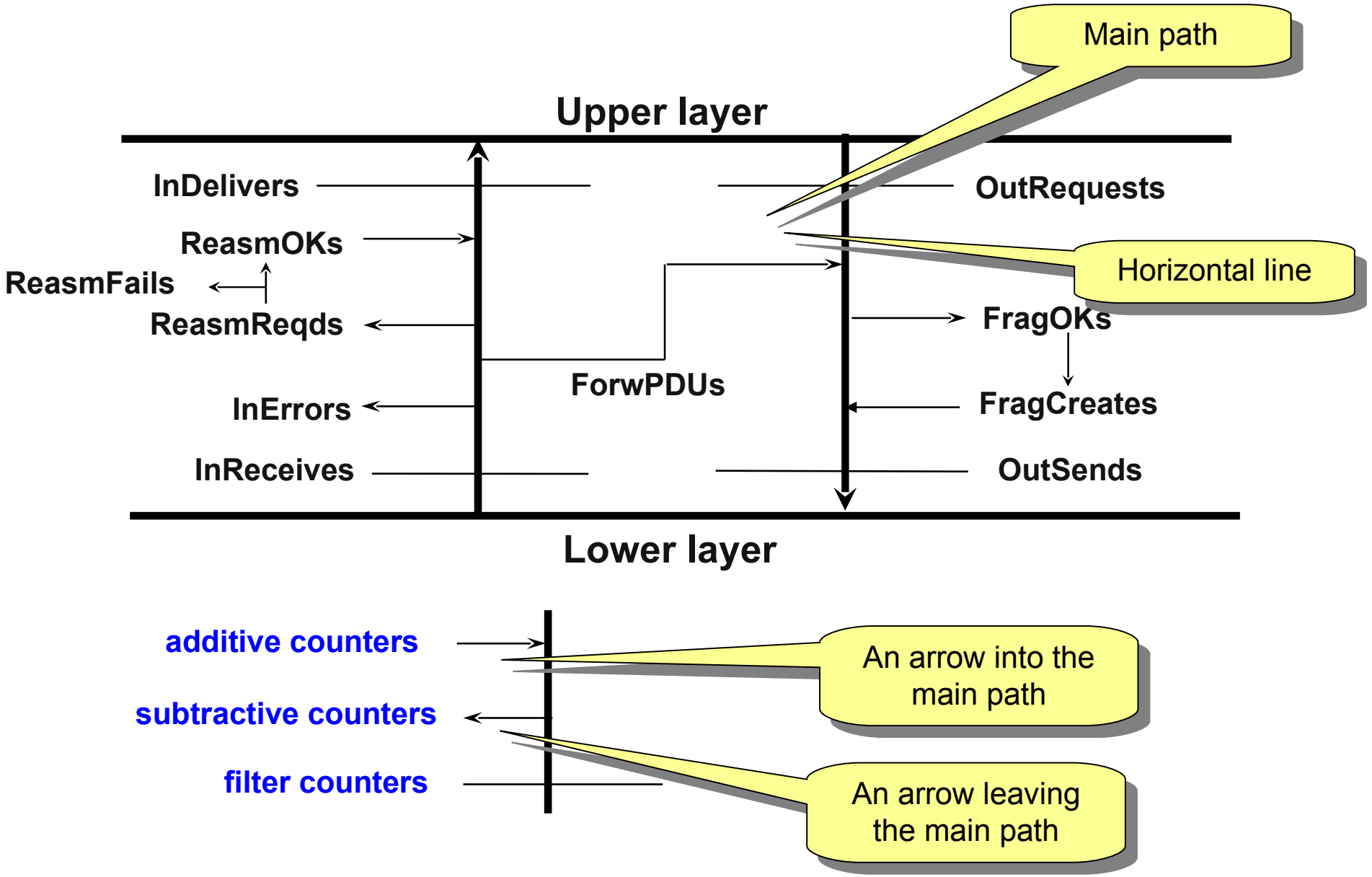
… 2.5.6 – Diagramas CASE

- Regras para construção de um diagrama “Case”:
  - caminho principal (main path) em cada direção entre a camada abaixo (n-1) e camada acima (n+1);
  - linha horizonta (horizontal line) cortando o caminho principal corresponde ao contador de PDUs que por ali passam;
  - flecha (arrow) partindo do caminho principal indica o contador para condições de erro ou fluxo que resultam PDUs que não continuam pelo caminho principal;
  - flecha (arrow) entrando no caminho principal contabiliza para aquele ponto PDUs que são injetadas no caminho principal.

… 2.5.6 – Diagramas CASE



… 2.5.6 – Diagramas CASE



… 2.5.6 – Diagramas CASE

InReceives = InErrors + ReasmReqds
+ ForwPDUs - ReasmOKs
+ InDelivers

OutSends = OutRequests + ForwPDUs
- FragOKs + FragCreates

… 2.5.6 – Diagramas CASE

- inErrors: número de mensagens ICMP recebidas, mas que contém erros específicos ICMP (e.g., checksum ou length X);
- ReasmReqds: número de fragmentos IP recebidos que necessitam ser remontados na sua entidade;
- ReasmOKs: números de datagramas IP remontados com sucesso;
- FragOKs: número de datagramas IP que foram fragmentos com sucesso na entidade;
- FragCreates: número de datagramas IP que foram gerados como resultado de fragmentação neste entidade.

2.5.7 – TRAPS

- Traps – utilizados pelo agente de gerenciamento para indicar um evento extraordinário para o gerente.

- Sintaxe fraca em SNMPv1:
  - não usa OIDs para identificar traps;
  - usa numeração "flat" com 6 eventos;
  - mecanismo de extensão quando o valor do campo de trap é “enterpriseSpecific(6)”
  - … mecanismo pouco utilizado, posto que o valor do “trap” e o campo “enterprise” são usados conjuntamente para identificar o “trap”.

- Mudanças grandes em SNMPv2

… 2.5.7 – TRAPS

- <trapItem> = <trapName> "TRAP-TYPE"
  - "ENTERPRISE" <oidName>
  - ["VARIABLES" "{" <varName>["," <varName>]... "}"]
  - ["DESCRIPTION" <description>]
  - ["REFERENCE" <reference>]
  - "::=" <value>

- Onde:
  - <trapName> - nome do trap sendo definido;
  - <oidName> - pode ser SNMP ou o valor a retornar (campo "enterprise");
  - <value> - valor do trap retornado em um dos campos
  - "generic-trap" ou "specific-trap";
  - Valores para <varName>, <description>, e <reference> serão definidos ...

… 2.5.7 – TRAPS

- Valores para “ENTERPRISE”:
  - determina o valor a retornar no campo “enterprise” da PDU “trap” do protocolo SNMP;
  - … se o valor especificado for "snmp", usa-se o valor de “sysObjectID” do agente que gerou o “trap” - “trap” é do tipo “snmp-generic”;
  - … com outro valor, este deve ser retornado na PDU e, neste caso, o “trap” é do tipo “enterprise-specific”.

… 2.5.7 – TRAPS

- Valores para “variables”:
  - “OPCIONAL” - indica quais objetos interessantes devem ser retornados no “TRAP”
  - “DESCRIPTION” deve indicar que instâncias retornar;
  - agente pode retornar mais variáveis
    - o até um máximo de 484 bytes;
    - o gerente deve estar pronto para receber quaisquer variáveis, não só aquelas especificas na cláusula variables.
- Valores para “DESCRIPTION”:
  - provê toda a semântica necessária à implementação
- Valores para “REFERENCE”:
  - como para OBJECT-TYPE

… 2.5.7 – TRAPS

- Valores para TRAP-TYPE:
- se for ENTERPRISE snmp, o valor deve estar entre 0 e 5 (GENERIC TRAP)
  - campo GENERIC-TRAP da pdu contém este valor
  - campo SPECIFIC-TRAP da pdu contém zero
- caso contrário, é um ENTERPRISE-SPECIFIC TRAP
  - campo GENERIC-TRAP da PDU contém enterpriseSpecific(6)
  - campo SPECIFIC-TRAP da PDU contém este valor.

… 2.5.7 – TRAPS

- e.g., TRAP:

- coldStart TRAP-TYPE
- ENTERPRISE snmp
- ::= 0
- fooTrap TRAP-TYPE
- ENTERPRISE foo
- ::= 45

- Onde:
  - coldStart - generic trap;
  - fooTrap - enterprise-specific trap.

… 2.5.7 – TRAPS

- Considerações para TRAPS:

- TRAPS não são confirmados (podem não ser recebidos) e não têm identificação única (não tem campo REQUEST-ID na pdu)
  - agente não sabe se o gerente recebeu o TRAP;
  - gerente não sabe se o TRAP é uma duplicata;
  - não há forma de garantir a recepção do TRAP.

2.5.8 – Dicas para criar MIBs Proprietárias

- Por que criar uma MIB proprietária ?!
  - MIBs padrões oferecem poucos objetos para controlar dispositivos – razão: … fraca segurança do SNMP (v1 e v2);
  - para monitorar/controlar características específicas dos dispositivos (*devices*) do fabricante;
  - para estender as MIBs padrões e aumentar a monitoração e controle.
- Deve-se adicionar na árvore “enterprises.nomeDaEmpresa”.

… 2.5.8 – Dicas para criar MIBs Proprietárias

- Organização de uma árvore proprietária:
  - cada fabricante organiza como quiser;
  - linha de produtos do fabricante afeta a organização da árvore;

- Proposta de um “Padrão” - utiliza 04 galhos:

  - 01 galho para registro de OIDs de produtos (para “sysObjectID”);
  - 01 galho experimental - (para experimentos, demos em feiras, etc);
  - 01 galho para estender MIBs padrão (duplicar a árvore - “shadow tree”);
  - 01 galho para MIBs proprietárias organizadas por linha de produto.