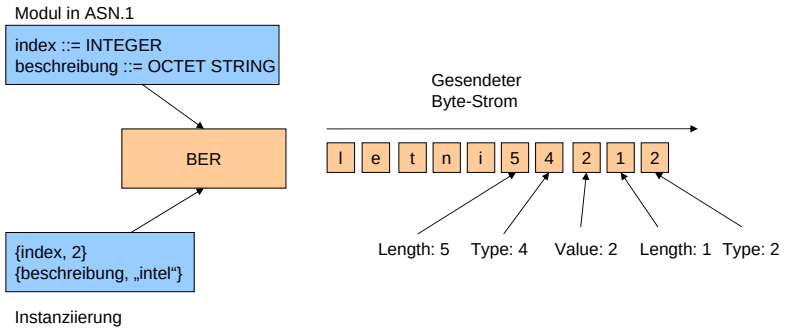
Netzmanagement – Übungsfolien

ASN.1 und BER

9.2.1 Abstrakte Syntax (ASN.1)

- Abstract Syntax Notation One (ASN.1)
  - Eine Sprache zur Spezifikation der Parameter von Benutzerdaten
  - Erlaubt die Spezifikation von Typen und Werten, ohne die Wertrepräsentation zu bestimmen
  - Angegeben als formale Grammatik in EBNF
- Kodierregeln
  - Spezifizieren die Abbildung der Benutzerdaten in die Form der zu übertragenden Daten
- Basic Encoding Rules (BER) für ASN.1
  - Eine spezielle Menge von Kodierregeln, die international standardisiert sind
- Verwendung unter anderem bei Netzmanagementsystemen (z.B. SNMP: Simple Network Management Protocol)
  - Datenrepräsentation via ASN.1
  - Datenaustausch via UDP/IP
- Trend geht bei neueren Protokollen zur Verwendung von XML als Spezifikations- und Beschreibungssprache

Beispiel



Definierbare Objekte

- Drei Arten definierbarer Objekte
  - Typen
    - definieren neue Datenstrukturen
    - werden mit einem Großbuchstaben am Anfang notiert
  - Werte
    - definieren die Ausprägungen von Typen
    - werden mit Kleinbuchstaben am Anfang notiert
  - Makros
    - ändern die Grammatik einer Sprache
    - werden mit Blockbuchstaben notiert (wie die Schlüsselwörter in ASN.1)
- Beispiele
  - Definition eines Typs
    - `TypName ::= TYPE`
  - Definition einer Variablen
    - `werteName TypName ::= WERT`
- Unterschiedene Datentypen
  - Einfache, zusammengesetzte und tagged Typen

ASN.1-Typen im Überblick

- Einfache Datentypen
  - BOOLEAN
  - INTEGER
  - BIT STRING
  - OCTET STRING
  - REAL
  - ENUMERATED
  - OBJECT IDENTIFIER
  - CHARACTER STRING
  - IA5STRING
  - NULL
- Zusammengesetzte Datentypen
  - SEQUENCE
  - SEQUENCE OF
  - SET
  - SET OF
  - CHOICE

Einfache Datentypen

- Beispiele
  - Beispiel BOOLEAN
    - aktiv ::= BOOLEAN --true or false
  - Beispiel INTEGER
    - index ::= INTEGER
    - InterfaceTyp ::= INTEGER {other(1), ethernetCsmacd(6), basicISDN(20), atm(37) }
      meinInterface InterfaceTyp ::= ethernetCsmacd --oder 6
      - In SNMP sind nur die explizit aufgezählten Werte zulässig
  - Beispiel BITSTRING
    - DienstKlassenTyp ::= BITSTRING {lesen(0), schreiben(1), dateizugriff(2)}
      meinDienst DienstKlassenTyp ::= {lesen, schreiben}
      - Wird oft für eine Liste von Optionen benutzt
      - Optionen können unabhängig voneinander gesetzt oder gelöscht werden
  - Beispiel IA5STRING
    - Beschreibung ::= IA5STRING
  - Beispiel ENUMERATED
    - Woche ::= ENUMERATED {Montag(0), Dienstag(1), ... Sonntag(6)}
  - Beispiel NULL
    - Enthält kein Datum
    - Wird vor allem beim zusammengesetzten Datentyp **CHOICE** benutzt

Einfache Datentypen

- Beispiel OBJECT IDENTIFIER
  - Sequenz von nicht-negativen Integer-Werten, die einen Pfad in einem Baum spezifizieren
  - z.B. Identifizierung von Objekten in einer Management Information Base (MIB)
  - `interfaces.table.entry.inOctets` oder 2.2.1.10

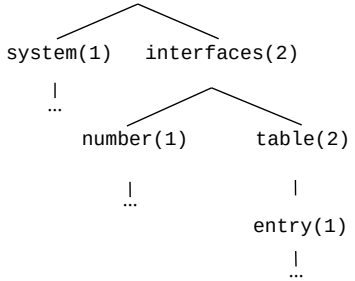
```
MIB-Objekt ::= OBJECT IDENTIFIER
Daten MIB-Objekt ::= {
        2 2 1 10
}
```

oder

```
Daten MIB-Objekt ::= {
        interfaces table entry 10
}
```

oder

```
Daten MIB-Objekt ::= {
        interfaces(2) table(2) entry(1) inOctets(10)
}
```



Zusammengesetzte Datentypen

- SEQUENCE
  - Geordnete Liste von Datentypen mit fester Anzahl von Elementen
  - Einige Datentypen können als optional gekennzeichnet sein
- SET
  - Ungeordnete Menge von eindeutig unterscheidbaren Datentypen
  - Feste Anzahl von Elementen
  - Einige Datentypen können als optional gekennzeichnet sein
- SEQUENCEOF
  - Geordnete Liste von Elementen des gleichen Datentyps
  - Anzahl der Elemente kann fest oder variabel sein
- SETOF
  - Ungeordnete Menge von Elementen des gleichen Datentyps
  - Anzahl der Elemente kann fest oder variabel sein
- CHOICE
  - Ungeordnete Menge von eindeutig unterscheidbaren Datentypen
  - Aus den Datentypen kann eine Alternative ausgewählt werden

Zusammengesetzte Datentypen

- Beispiel SEQUENCE
  - ```
    InterfaceEntry ::= SEQUENCE {
            index            INTEGER,
            beschreibung     IA5STRING,
            dateneinheitenIN INTEGER,
            aktiv            BOOLEAN
    }
    ```
- Beispiel SET
  - ```
    interface ::= SET {
            index            INTEGER,
            beschreibung     IA5STRING
    }
    ```


Zusammengesetzte Datentypen: Beispiel

```
Interface ::= SEQUENCE {
      index                INTEGER,
      beschreibung         IA5STRING,
      alias                IA5STRING    OPTIONAL,
      typ                  INTEGER {other(1), ethernetCsmacd(6)},
      dateneinheitenIN     INTEGER,
      aktiv                BOOLEAN
  }
```

- Instanziierung:

```
LAN-Interface Interface ::= {
      index                0,
      beschreibung         "3Com",
      typ                  6,
      dateneinheitenIN     42,
      aktiv                true
}
```

  - Neben Optional-Werten können auch Default-Werte angegeben werden

Datentypen mit Tag

- Tag hat die Aufgabe, einen Datentyp eindeutig zu identifizieren
  - Vergleichbar mit Index, wie er bei Arrays in vielen Programmiersprachen verwendet wird
  - 4 Klassen
    - UNIVERSAL
      - Global eindeutige Identifikation. Werden im ASN.1-Standard definiert (BOOLEAN, INTEGER, SEQUENCE, ...)
    - CONTEXT-SPECIFIC
      - Eindeutig "innerhalb" eines Konstruktortyps (SEQUENCE, SET)
    - APPLICATION
      - Eindeutig innerhalb eines ASN.1-Moduls
    - PRIVATE
      - Definiert durch Unternehmen bzw. Anwendung; dort eindeutig
  - Beispiele
    - ```
      Typ1 ::= [4] INTEGER
      Typ2 ::= [APPLICATION 2] OCTETSTRING
      Typ3 ::= [PRIVATE 0] BOOLEAN
      ```
    - Auslassen der Klasse impliziert CONTEXT-SPECIFIC

Beispiele zur Verwendung von Tags

- Beispiel „InterfaceEntry“
  - ```
    InterfaceEntry ::= SEQUENCE {
            index            [APPLICATION 1]  INTEGER,
            beschreibung     [1]              IA5STRING,
            dateneinheitenIN [2]              INTEGER,
            aktiv            [3]              BOOLEAN
    }
    ```
  - index
    - Annahme: wird in anderen Typdefinitionen ebenfalls verwendet und erhält deshalb einen eindeutigen Tag innerhalb dieser Anwendung
  - Die weiteren Elemente müssen nur innerhalb dieses Datentyps referenziert werden
- Weitere Beispiele
  - ```
    interface  ::=  SET {
            index            [0] INTEGER,
            beschreibung         IA5STRING,
            MTU              [1] INTEGER
    }
    ```
  - ```
    datenIN ::= CHOICE {
            Dateneinheiten   [0] INTEGER,
            Byte             [1] INTEGER
    }
    ```
  - Typen müssen bei SET und CHOICE unterschiedlich sein, deshalb sind die Tags wichtig

Datentypen mit IMPLICIT

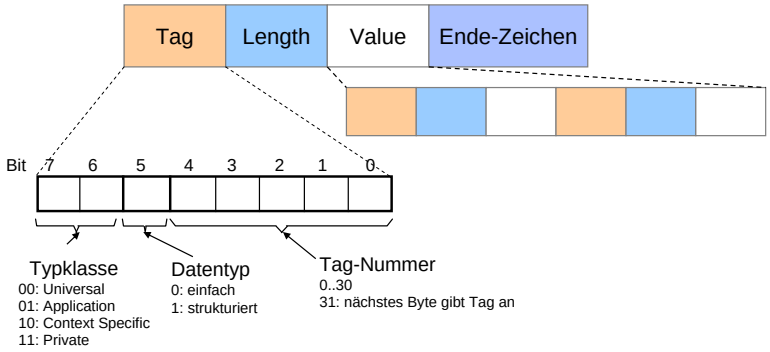
- IMPLICIT
  - Der ursprüngliche Typ des Datums muss bei der Übertragung nicht angegeben werden
  - Ursprünglicher Datentyp kann durch den äußeren Tag erkannt werden
  - Einsparung bei der zu übertragenden Datenmenge führt zu besserer Effizienz
  - Nachteil: Typüberprüfung nicht mehr möglich
- Beispiel „interface Entry“
  - ```
    interfaceEntry ::= SEQUENCE {
                       index             [APPLICATION 1] INTEGER,
                       beschreibung      [1] IMPLICIT    IA5STRING,
                       dateneinheitenIN  [2] IMPLICIT    INTEGER,
                       aktiv             [3] IMPLICIT    BOOLEAN
    }
    ```
    - Beim impliziten Tagging wird nur noch der CONTEXT-SPECIFIC Tag übertragen, der UNIVERSAL-Tag wird ausgelassen

9.2.2 Transfersyntax (BER)

- Jede Kodierung eines Datenwertes umfasst gemäß BER (*Basic Encoding Rules*) die folgenden Teile
  - Tag
    - Ein oder mehrere Oktette, welche eine Kodierung für Datentyp-Klasse und Tag-Nummer enthalten
  - Länge
    - Gibt die Länge des Inhalts in Oktetten an oder kennzeichnet, dass ein spezielles Ende-Zeichen vorhanden ist
  - Ende-Zeichen
    - Ein Ende-Zeichen, bestehend aus zwei 0-Oktetten
    - Kann vorhanden sein
  - Inhalt
    - Hier wird der eigentliche Datenwert kodiert. Der Inhalt eines zusammengesetzten Typs wird entsprechend auf naheliegende Weise gebildet, z.B.
      - SequenceType: Der Dateninhalt ist eine Folge von Datenwerten korrespondierend zu den in der Definition aufgeführten Typen
      - SequenceOfType: Der Dateninhalt besteht aus null oder mehreren Datenwerten zu dem angegebenen Typ
      - ChoiceType: Der Dateninhalt ist gleich dem Dateninhalt zu dem ausgewählten Typ
  - Anmerkung: Oktett (engl.: Octet) = Byte mit 8 Bits

BER-Kodierung

- Grundlegender Aufbau



- Beispiel

Längenfeld

- Bekannte Länge
  - Kurze Form [1 Byte]
    - Erstes Bit 0
    - Die anderer 7 Bits kodieren die Länge als eine binäre Ganzzahl
  - Lange Form [n Bytes, n>1]
    - Erstes Bit 1
    - Alle anderen Bits des ersten Bytes kodieren die Anzahl der Längenoktette
    - Alle anderen Bits aller folgenden Längenoktette kodieren die Länge als eine binäre Ganzzahl
- Unbekannte Länge
  - Kennung [1 Byte]
    - Erstes Bit 1
    - Alle anderen Bits 0
  - Wird bei der Kodierung zusammengesetzter Datentypen verwendet, bei denen die Länge nicht sofort verfügbar ist
    - Im SNMP nicht zugelassen
    ```

Beispiele

- BOOLEAN: TRUE

00	0	00001	00000001	11111111

$= 1_{16}\ 1_{16}\ FF_{16}$

- INTEGER: 100

00	0	00010	00000001	01100100

$= 2_{16}\ 1_{16}\ 100_{10}$

- INTEGER: 256

00	0	00010	00000010	00000001	00000000

$= 2_{16}\ 2_{16}\ 256_{10}$

- BITSTRING: "0111110111"

00	0	00011	00000010	01111101	11000000

$= 3_{16}\ 2_{16}\ 0111110111_{2}$

- OCTET STRING: "abcd"

00	0	00100	00000100	01100001	01100010	01100011	01100100

$= 4_{16}\ \ 4_{16}\ \ 97_{10}\ \ 98_{10}\ 99_{10}\ 100_{10}$

Beispiele

- Sequence
  - interface ::= SEQUENCE {
    index INTEGER,
    beschreibung IA5STRING
    }

    LAN-Interface ::= {
    index 0,
    beschreibung "3Com"
    }

48 9

00I1I10000	00001001

2 Länge=1 0

00I0I00010	00000001	00000000

22 Länge=4 "3" "C" "o" "m"

00I0I10110	00000100	00110011	01000011	01101111	01101101