ASN.1 Reference Card

NOTE – Optional items and stylistic advice are greyed-out

Basic syntax

Names: Letters (case-sensitive), digits, and hyphen.

Module names: Start upper case.

Type reference names: Start upper case.

Identifiers (for sequence components, choice alternatives, named bits, named numbers, and enumerations): Start lower case.

Value reference names: Start lower case.

It is common to use either or both of hyphens or upper case within names to separate parts of the name. (e.g. My-type or MyType)

Single-line comments start with -- and end with -- or new line.

Block comments start with /* and end with */.

Block comments can contain other block comments and/or double-hyphen comments. For historical reasons, multiple single-line comments are often used instead of block comments.

Module boiler-plate

```
          MY-MODULE { <oid> }
                  DEFINITIONS
                  AUTOMATIC TAGS ::=
                  BEGIN
                  EXPORTS <exports clause>;
                  IMPORTS <import clause>;
                  <Type and value assignments>
                  END
```

Type and value assignments

```
<TypeReferenceName> ::= <TypeDefinition>
e.g.:
Age          ::= INTEGER
Country-name ::= UTF8String
Greeting     ::= UTF8String
Hex-string   ::= OCTET STRING

<valueReferenceName> <TypeOfValue> ::= <ValueNotation>
e.g.:
twenty-one-today Age ::= 21
spain Country-name   ::= "ESPAÑA"
when GeneralizedTime ::= "200208192349.57894Z"
large-prime INTEGER  ::= 1999999973

<XMLvalueReferenceName> ::= <XMLValueNotation>
e.g.:
sixty-five-today ::= <Age>65</Age>
my-octets ::= <Hex-string>89aef764AEF</Hex-string>
plain-greeting ::= <Greeting>Hello World!</Greeting>
bells-and-whistles-greeting ::=
          <Greeting>
              <bel/><stx/>Hello World!<etx/>
          </Greeting>
```

More type definition examples

(This also illustrates use of simple value notation in DEFAULTs)

```
My-sequence ::= SEQUENCE {
        first   BOOLEAN,
        second  INTEGER OPTIONAL,
        third   INTEGER DEFAULT 129,
        fourth  BOOLEAN DEFAULT TRUE,
        fifth   REAL    DEFAULT 0.629,
                  -- Or DEFAULT 62.9E-2
        sixth   UTF8String DEFAULT "گعهلإچﷲهمئ",
                  -- Unicode characters
        seventh IA5String  DEFAULT "James Morrison",
                  -- ASCII characters
        eighth  BIT STRING   DEFAULT '101100011'B,
        ninth   OCTET STRING DEFAULT '89AEF764'H,
        tenth   Alternatives }

Alternatives  ::= CHOICE {
        first-alternative       TypeA,
        second-alternative      TypeB,
        third-alternative       NULL  }

DailyMaxTemperaturesForMonth ::=
       SEQUENCE (SIZE(28..31)) OF temperature INTEGER
```

And yet more type definitions …….

```
VersionsSupported ::= BIT STRING {
        version1  (0),
        version2  (1),
        version3  (2) }

Message ::= SEQUENCE { ....... ,
        version-bit-map  VersionsSupported
                         DEFAULT {version1} }

Color ::= INTEGER {
        red(10), orange(20), yellow(30), green(40),
        blue(50), indigo(60), violet(70) } (0..80)

Codes ::= ENUMERATED {code1(0),code2(1),code3(2)}
```

Export and import clauses

```
EXPORTS TypeA, TypeB, valueC ;
        -- Note the semi-colon

IMPORTS TypeA, TypeB, valueC FROM
                  MODULE-A { ....... }
        TypeD, TypeE FROM
                  MODULE-B { ....... } ;
```

Example Object Identifier Values

```
oid1 OBJECT IDENTIFIER ::=
 {iso standard 2345 modules (0) basic-types (1)}
oid2 OBJECT IDENTIFIER ::= {joint-iso-itu-t ds(5)}
oid3 OBJECT IDENTIFIER ::= { oid2 modules(0) }
oid4 OBJECT IDENTIFIER ::= { oid3 basic-types(1) }
oid5 OBJECT IDENTIFIER ::= { 2 5 0 1 } -- equals oid4
```

Constraining types

```
INTEGER (0..MAX)          -- only non-negative values
INTEGER (-6..3 | 10..30) -- only -6 to 3 or 10-30
INTEGER (ALL EXCEPT 0)    -- 0 not allowed
SEQUENCE (SIZE (0..10)) OF INTEGER
IA5String (SIZE (1..25))(FROM ("A" .. "Z"))
     -- Only sizes 1-25 and characters "A"-"Z" allowed
OCTET STRING (CONTAINING My-Type
              ENCODED BY perBasicAligned)
              -- perBasicAligned is imported
              -- from the ASN.1 standards
UTF8String (PATTERN "\d#4-\d#2-\d#2")
BIT STRING (CONSTRAINED BY ....... )
SEQUENCE {.......} (WITH COMPONENTS .......)
-- PATTERN, CONSTRAINED BY and WITH COMPONENTS
-- are out of the scope of this reference card
```

ASN.1 as an XML schema definition

```
Message ::= SEQUENCE {
   sender-id    OBJECT IDENTIFIER,
   urgency      ENUMERATED { high, normal, low },
   actions      SEQUENCE OF CHOICE {
        insert  InsertionDetails,
        remove  RemovalDetails,
        update  UpdateDetails},
   names        SEQUENCE OF name UTF8String,
   confirm      BOOLEAN  }

This can be used to validate the following XML
document:
xml-document ::=
  <Message>
     <sender-id>2.39.6.45</sender-id>
     <urgency><normal/></urgency>
     <actions>
        <remove>.......</remove>
        <insert>.......</insert>
        <insert>.......</insert>
     </actions>
     <names>
        <name>.......</name>
        <name>.......</name>
     </names>
     <confirm><true/></confirm>
  </Message>
```

Extensibility

```
Message ::= SEQUENCE {
       first   TypeA,
       second  TypeB,
       ... ,
       [[2: version2        TypeC ]],
       [[4: version4-first  TypeD,
            version4-second TypeE ]],
       [[6: version6        TypeF ]] }

Alternatives ::= CHOICE {
       first    TypeA,
       second   TypeB,
       ... ,
       version2 TypeC }

Codes ::= ENUMERATED {code1, code2, ...,
                            v2-code, another-code}

NameSizes ::= INTEGER (1..64, ..., 65..MAX)
```

Other syntax

Obsolete, not commonly used or deprecated syntax is greyed out below. Other syntax is for advanced use. These constructions are out of the scope of this reference card.

```
       MODULE-NAME.TypeName
       ABSTRACT-SYNTAX
       IMPLICIT TAGS
       EXPLICIT TAGS
       EXPORTS ALL
       EXTENSIBILITY IMPLIED
       selection < ChoiceTYpe
       COMPONENTS OF SequenceType
       SEQUENCE {
            first    [0] INTEGER OPTIONAL,
            second   [1] EXPLICIT INTEGER,
            last     [99] IMPLICIT UserData }
       SEQUENCE { ......., ... !29 }
       [APPLICATION 29], [PRIVATE 6]
       SET { ....... }
       SET OF
       RELATIVE-OID
       EMBEDDED PDV
       EXTERNAL
       INSTANCE OF
       My-values INTEGER ::=
        {Set1 INTERSECTION (Set2 UNION Set3)EXCEPT Set4}
       PrintableString (SIZE (NameSizes) )
            --where-- NameSizes ::= INTEGER (0..64)
       CHARACTER STRING
       ObjectDescriptor
       UTCTime
```

```
       BMPString
       GeneralString
       GraphicString
       ISO646String
       NumericString
       PrintableString
       T61String
       TeletexString
       UniversalString
       VideotexString
       VisibleString
```

Definitions for information objects

Information object class names and words in WITH SYNTAX clauses are all upper case.

Information object names start with lower-case, information object set names start with upper case.

```
MY-SIMPLE-CLASS ::= TYPE-IDENTIFIER

MY-CLASS ::= CLASS {
   -- Note use of upper/lower case after &.
   -- This is semantically significant.
   &id                   OBJECT IDENTIFIER  UNIQUE,
   &simple-value         ENUMERATED
               {high, medium, low} DEFAULT medium,
   &Set-of-values        INTEGER OPTIONAL,
   &Any-type,
   &an-inform-object     SOME-CLASS,
   &A-set-of-objects     SOME-OTHER-CLASS }
        WITH SYNTAX
          { KEY &id
            [ URGENCY &simple-value ] -- Optional
            [ VALUE-RANGE &Set-of-values ]
            PARAMETERS &Any-type
            SYNTAX  &an-inform-object
            MATCHING-RULES &A-set-of-objects
            -- WORDS are optional and commas can be used
            -- as separators --}

my-object  MY-CLASS ::= {
   KEY      { ....... }
   URGENCY  high
   VALUE-RANGE { 1..10 | 20..30 }
   PARAMETERS  My-type
   SYNTAX  defined-syntax
   MATCHING-RULES { at-start | at-end | exact } }

My-object-set MY-CLASS ::= { object1|object2|object3,
                             ...,
                             version2-object }
```

```
Message ::= SEQUENCE {
   key      MY-CLASS.&id ({My-object-set}),
             -- Has to be an OBJECT-ID from the set
   parms    MY-CLASS.&Any-type ({My-object-set} {@key})
             -- Has to be the PARAMETERS for the
             -- object with KEY. -- }

-- Variable type value fields and value set fields
-- are out of the scope of this reference card
```

Parameterisation

```
Invoke-message {INTEGER:normal-priority, Parameter} ::=
     SEQUENCE {
        component1    INTEGER DEFAULT normal-priority,
        component2    Parameter }

Messages ::= CHOICE {
       first  Invoke-message { low-priority,  Type1 },
       second Invoke-message { high-priority, Type2 },
       ... }
```

Encodings

PER:	A compact binary encoding transferring the minimum information needed to identify a value.
XER:	Encoding ASN.1 values as XML syntax.
BER:	A tag-length-value style of binary encoding very popular in the 1980s.
DER:	An encoding with only one way to encode a given value, used in security work.
CER:	Another security-related encoding, not much used.

An encoding control notation (ECN) is available to completely determine the encoding of ASN.1 values. There are also Encoding Instructions that can vary XER and other encodings, for example, to determine which components of a sequence are to be encoded as XML attributes. These are not in the scope of this Reference Card.

Further information

The following URL provides further detail and links:

http://www.oss.com/asn1/tutorial/ReferenceCard.html