

NAMES: A..z, 0..9, - (hyphen), Case-sensitive.

AUTOMATIC TAGS: eliminate any potential issues with manual tag assignment.

MODULE: a collection of ASN.1 definition statements. Module body starts with BEGIN, ends with END.

IMPORTS: used when an ASN.1 module needs to use a definition from a different module.

TYPE NAMES: start with an upper case letter
<TypeName> ::= <TypeDefinition>

VALUE NAMES & FIELD NAMES: start with a lower case letter

OPTIONAL & DEFAULT: both indicate that the field is optional and need not be present in a SEQUENCE. The DEFAULT value is assumed when the field is absent.

VALUES: used mostly as examples and for testing. In real life data is assigned dynamically at runtime.

COMMENTS: start with -- (two hyphens) and end with -- or new line, or start with /* and end with */.

UNICODE: support for UTF8 strings.

XML: -- alternative way to assign values using XML notation.

ASN.1 is a notation used to describe data types. **ASN.1** also specifies the data encoding rules (serialization)

CONSTRAINTS: single value | value range | SIZE | permitted alphabet | PATTERN (regex) | duration.

EXTENSIBILITY: ensures new versions of the protocol will not be disruptive to existing implementations.

Version brackets (greyed out) can be optionally used to group extension items together.

ASN.1 & XML: ASN.1 can be used as your XML data schema definition.

INFORMATION OBJECTS & OPEN TYPES: allow complex restrictions for values to match entries in a reference set

ASN.1 provides an advanced way to define Item using Information Objects and Open Types as shown in the example here.

INFORMATION OBJECT CLASS: use of upper/lower case after & is semantically significant.

INFORMATION OBJECTS SET: each entry in Catalog becomes a "restriction" on what values Item can have.

Value of incorrectItem does not satisfy the constraints on its type defined in Catalog.

```
MyShop-Module1 { <oid> } -- oid - object identifier is optional
DEFINITIONS
AUTOMATIC TAGS ::=
```

```
BEGIN
```

```
IMPORTS Item, Address FROM MyItems-Module2;
```

```
PurchaseOrder ::= SEQUENCE {
  dateOfOrder    DATE,
  name           UTF8String (SIZE(3..50)) DEFAULT "N/A",
  address        Address,           -- imported
  phone          Phone OPTIONAL,   -- defined below
  items          ListOfItems       -- defined below via imported Item
}
```

```
-- types that are referenced by PurchaseOrder, but can also be used elsewhere
ListOfItems ::= SEQUENCE (SIZE (1..100)) OF Item -- an "array"
Phone ::= VisibleString (PATTERN "\d#3-\d#3-\d#4")
```

```
-- examples of my values --
```

```
myName UTF8String ::= "Иван Грозный"
myPhone Phone ::= "333-444-5555"
myFavorite Item ::= {id color:green, quantity crate, unitPrice 1.99}
myAddr1 Address ::= {street "1st Ave", city "Somerset", state "NJ", zip "08873"}
myAddr2 ::= <Address> <street>2nd Ave</street> <city>Somerset</city>
<state>NJ</state> <zip>08873</zip> </Address>
```

```
END
```

```
MyItems-Module2 DEFINITIONS AUTOMATIC TAGS ::=
```

```
BEGIN
```

```
-- By default, all defined types (Item and Address) are exported
```

```
Item ::= SEQUENCE {
  id CHOICE { -- id alternatives - code, url or color
    code INTEGER (0..MAX),
    url VisibleString,
    color ENUMERATED { black, blue, ... , -- extended
      green, red}
  } DEFAULT code:9999,
  quantity INTEGER {single(1), dozen(12), crate(36)},
  options BIT STRING DEFAULT '101100011'B,
  unitPrice REAL ( 1.00 .. 9999.00 ),
  ... , -- extension allowed below this line
  [[ isTaxable BOOLEAN DEFAULT FALSE ]], -- added to Item in v.2
  [[ voltage INTEGER (110 | 220) OPTIONAL ]], -- added in v.3
}
```

```
Address ::= SEQUENCE {
  street VisibleString (SIZE (5 .. 50)),
  city VisibleString (ALL EXCEPT "Springfield"),
  state VisibleString (SIZE(2) ^ FROM ("A".."Z")),
  zip NumericString (SIZE(5 | 9))
}
```

```
END
```

```
PRODUCT ::= CLASS { -- Information Object Class
  &id INTEGER (1..MAX) UNIQUE,
  -- Open Type -- &Feature, -- starts upper case, type varies per item
  &price REAL
} WITH SYNTAX { &id, &Feature, &price }
```

```
Catalog PRODUCT ::= { -- Information Object Set
  -- id Feature type price --
  { 101, INTEGER (110 | 220), 20.00 } | -- Charger
  { 104, NULL, 99.00 } | -- Glass Egg
  { 105, Event, 9.99 } -- $9.99 Basket
}
```

```
Event ::= ENUMERATED {christmas, easter}
```

```
Item ::= SEQUENCE {
  ident PRODUCT.&id ( {Catalog} ),
  feat PRODUCT.&Feature ( {Catalog}{@ident} ),
  unitPrice PRODUCT.&price ( {Catalog}{@ident} )
}
```

```
correctItem Item ::= {id 105, feat Event:easter, unitPrice 9.99}
incorrectItem Item ::= {id 101, feat Event:easter, unitPrice 99.00}
```

OBJECT IDENTIFIER VALUES	VALUES																																																																																																
<p>oid1 OBJECT IDENTIFIER ::= { iso standard 2345 modules (0) basic-types (1) }</p> <p>oid2 OBJECT IDENTIFIER ::= { joint-iso-itu-t ds(5) }</p> <p>oid3 OBJECT IDENTIFIER ::= { oid2 modules(0) }</p> <p>oid4 OBJECT IDENTIFIER ::= { oid3 basic-types(1) }</p> <p>oid5 OBJECT IDENTIFIER ::= { 2 5 0 1 } -- equals oid4</p> <p>Object identifier value Meaning</p> <p>{ 1 2 } ISO member bodies</p> <p>{ 1 2 840 } US (ANSI)</p> <p>{ 1 2 840 113549 } RSA Data Security, Inc.</p> <p>{ 1 2 840 113549 1 } RSA Data Security, Inc. PKCS</p> <p>{ 2 5 } directory services (X.500)</p> <p>{ 2 5 8 } directory services-algorithms</p> <p style="text-align: center;">TYPES</p> <table border="1"> <thead> <tr> <th>Basic Types</th> <th>Tag</th> <th>Other Types</th> <th>Tag</th> </tr> </thead> <tbody> <tr> <td>BOOLEAN</td> <td>dec/hex [01/01]</td> <td>ObjectDescriptor</td> <td>dec/hex [07/07]</td> </tr> <tr> <td>INTEGER</td> <td>[02/02]</td> <td></td> <td></td> </tr> <tr> <td>BIT STRING</td> <td>[03/03]</td> <td>EXTERNAL</td> <td>[08/08]</td> </tr> <tr> <td>OCTET STRING</td> <td>[04/04]</td> <td>EMBEDDED PDV</td> <td>[11/0B]</td> </tr> <tr> <td>NULL</td> <td>[05/05]</td> <td></td> <td></td> </tr> <tr> <td>OBJECT IDENTIFIER</td> <td>[06/06]</td> <td>OID-IRI</td> <td>[35/ *]</td> </tr> <tr> <td>RELATIVE-OID</td> <td>[13/0b]</td> <td>RELATIVE-OID-IRI</td> <td>[36/ *]</td> </tr> <tr> <td>REAL</td> <td>[09/09]</td> <td></td> <td></td> </tr> <tr> <td>ENUMERATED</td> <td>[10/0A]</td> <td>SET</td> <td>[17/11]</td> </tr> <tr> <td></td> <td></td> <td>SET OF</td> <td>[17/11]</td> </tr> <tr> <td>SEQUENCE</td> <td>[16/10]</td> <td></td> <td></td> </tr> <tr> <td>SEQUENCE OF CHOICE</td> <td>[16/10]</td> <td>UTCTime</td> <td>[23/17]</td> </tr> <tr> <td></td> <td>----</td> <td>GeneralizedTime</td> <td>[24/18]</td> </tr> <tr> <td>UTF8String</td> <td>[12/0C]</td> <td>PrintableString</td> <td>[19/13]</td> </tr> <tr> <td>NumericString</td> <td>[18/12]</td> <td>T61String</td> <td>[20/14]</td> </tr> <tr> <td>IA5String</td> <td>[22/16]</td> <td>VideotexString</td> <td>[21/15]</td> </tr> <tr> <td>VisibleString</td> <td>[26/1A]</td> <td>GraphicString</td> <td>[25/19]</td> </tr> <tr> <td></td> <td></td> <td>GeneralString</td> <td>[27/1B]</td> </tr> <tr> <td>DATE</td> <td>[31/ *]</td> <td>UniversalString</td> <td>[28/1C]</td> </tr> <tr> <td>TIME-OF-DAY</td> <td>[32/ *]</td> <td>CHARACTER STRING</td> <td>[29/1D]</td> </tr> <tr> <td>DATE-TIME</td> <td>[33/ *]</td> <td>BMPString</td> <td>[30/1E]</td> </tr> <tr> <td>DURATION</td> <td>[34/ *]</td> <td>ISO646String</td> <td>[26/1A]</td> </tr> <tr> <td></td> <td></td> <td>TeletexString</td> <td>[20/14]</td> </tr> </tbody> </table> <p>*occupies two octets</p>	Basic Types	Tag	Other Types	Tag	BOOLEAN	dec/hex [01/01]	ObjectDescriptor	dec/hex [07/07]	INTEGER	[02/02]			BIT STRING	[03/03]	EXTERNAL	[08/08]	OCTET STRING	[04/04]	EMBEDDED PDV	[11/0B]	NULL	[05/05]			OBJECT IDENTIFIER	[06/06]	OID-IRI	[35/ *]	RELATIVE-OID	[13/0b]	RELATIVE-OID-IRI	[36/ *]	REAL	[09/09]			ENUMERATED	[10/0A]	SET	[17/11]			SET OF	[17/11]	SEQUENCE	[16/10]			SEQUENCE OF CHOICE	[16/10]	UTCTime	[23/17]		----	GeneralizedTime	[24/18]	UTF8String	[12/0C]	PrintableString	[19/13]	NumericString	[18/12]	T61String	[20/14]	IA5String	[22/16]	VideotexString	[21/15]	VisibleString	[26/1A]	GraphicString	[25/19]			GeneralString	[27/1B]	DATE	[31/ *]	UniversalString	[28/1C]	TIME-OF-DAY	[32/ *]	CHARACTER STRING	[29/1D]	DATE-TIME	[33/ *]	BMPString	[30/1E]	DURATION	[34/ *]	ISO646String	[26/1A]			TeletexString	[20/14]	<p>Values are usually specified in ASN.1 modules only for indicating defaults, or ranges for constraining items (such as the maximum length of a name).</p> <p>defaultOn BOOLEAN ::= TRUE</p> <p>maxAge INTEGER ::= 120</p> <p>bitmask BIT STRING ::= '7FFF'H</p> <p>defaultBytes OCTET STRING ::= '010F'H</p> <p>placeholder NULL ::= NULL</p> <p>defaultID OBJECT IDENTIFIER ::= {joint-iso-itu-t country(16) us(840)}</p> <p>defaultPrice REAL ::= 9.99</p> <p>Item ::= SEQUENCE {</p> <p> id CHOICE {</p> <p> -- id alternatives - code, url or color</p> <p> code INTEGER (0..MAX),</p> <p> url VisibleString,</p> <p> color ENUMERATED { black, blue, ... ,-- extended green, red}</p> <p> } DEFAULT code:9999,</p> <p> quantity INTEGER {single(1), dozen(12), crate(36)},</p> <p> options BIT STRING DEFAULT '10110001'B,</p> <p> unitPrice REAL (1.00 .. 9999.00)</p> <p> ... -- extension allowed below this line</p> <p> [[isTaxable BOOLEAN DEFAULT FALSE]], -- added to Item in v.2</p> <p> [[voltage INTEGER (110 220) OPTIONAL]]] -- added in v.3</p> <p> }</p> <p>defaultItem Item ::= { -- This is a value for the type above</p> <p> id code : 1,</p> <p> quantity single,</p> <p> options '0'B</p> <p> unitPrice 1.99</p> <p> }</p> <p>ListofNumbers ::= SEQUENCE OF INTEGER</p> <p>firstPrimeNumbers ListofNumbers ::= {1, 2, 3, 5, 7, 11, 13, 17}</p> <p>name1 UTF8String ::= "Joe" -- can also hold international characters</p> <p>phone NumericString ::= "8885551212"</p> <p>text IA5String ::= "Arbitrary text - with punctuation, no problem."</p> <p>name2 VisibleString ::= "Joe" -- US ASCII without control characters</p> <p>myDay DATE ::= "2012-01-31"</p> <p>noon TIME-OF-DAY ::= "12:00:00"</p> <p>noonMyDay DATE-TIME ::= "2012-01-31T12:00:00"</p> <p>lunchtime DURATION ::= "PT1H" -- one hour for lunch</p> <p>Here is a common use for value notation for limiting a string size, especially if the same value will be used in multiple places:</p> <p>upperSize INTEGER ::= 64</p> <p>VisibleString (SIZE (0..upperSize))</p> <p>ItemList ::= SEQUENCE (SIZE(0..upperSize)) OF Item</p>
Basic Types	Tag	Other Types	Tag																																																																																														
BOOLEAN	dec/hex [01/01]	ObjectDescriptor	dec/hex [07/07]																																																																																														
INTEGER	[02/02]																																																																																																
BIT STRING	[03/03]	EXTERNAL	[08/08]																																																																																														
OCTET STRING	[04/04]	EMBEDDED PDV	[11/0B]																																																																																														
NULL	[05/05]																																																																																																
OBJECT IDENTIFIER	[06/06]	OID-IRI	[35/ *]																																																																																														
RELATIVE-OID	[13/0b]	RELATIVE-OID-IRI	[36/ *]																																																																																														
REAL	[09/09]																																																																																																
ENUMERATED	[10/0A]	SET	[17/11]																																																																																														
		SET OF	[17/11]																																																																																														
SEQUENCE	[16/10]																																																																																																
SEQUENCE OF CHOICE	[16/10]	UTCTime	[23/17]																																																																																														
	----	GeneralizedTime	[24/18]																																																																																														
UTF8String	[12/0C]	PrintableString	[19/13]																																																																																														
NumericString	[18/12]	T61String	[20/14]																																																																																														
IA5String	[22/16]	VideotexString	[21/15]																																																																																														
VisibleString	[26/1A]	GraphicString	[25/19]																																																																																														
		GeneralString	[27/1B]																																																																																														
DATE	[31/ *]	UniversalString	[28/1C]																																																																																														
TIME-OF-DAY	[32/ *]	CHARACTER STRING	[29/1D]																																																																																														
DATE-TIME	[33/ *]	BMPString	[30/1E]																																																																																														
DURATION	[34/ *]	ISO646String	[26/1A]																																																																																														
		TeletexString	[20/14]																																																																																														
<p style="text-align: center;">INFORMATION OBJECTS</p> <p>Use of upper/lower case after '&' is semantically significant.</p> <p>MY-SIMPLE-CLASS ::= TYPE-IDENTIFIER</p> <p>MY-CLASS ::= CLASS {</p> <p> &id OBJECT IDENTIFIER UNIQUE,</p> <p> &simple-value ENUMERATED {high, low} DEFAULT low,</p> <p> &set-of-values INTEGER OPTIONAL,</p> <p> &any-type,</p> <p> &an-inform-object SOME-CLASS,</p> <p> &a-set-of-objects SOME-OTHER-CLASS</p> <p>} WITH SYNTAX</p> <p> { KEY &id</p> <p> [URGENCY &simple-value] -- Optional</p> <p> [VALUE-RANGE &set-of-values]</p> <p> PARAMETERS &any-type</p> <p> SYNTAX &an-inform-object</p> <p> MATCHING-RULES &a-set-of-objects</p> <p> }</p> <p>my-object MY-CLASS ::= {</p> <p> KEY { }</p> <p> URGENCY high</p> <p> VALUE-RANGE { 1..10 20..30 }</p> <p> PARAMETERS My-type</p> <p> SYNTAX defined-syntax</p> <p> MATCHING-RULES { at-start at-end exact }</p> <p>}</p> <p>My-object-set MY-CLASS ::= {</p> <p> object1 object2 object3,</p> <p> </p> <p> version2-object</p> <p>}</p> <p>Message ::= SEQUENCE {</p> <p> -- Has to be an OBJECT IDENTIFIER (KEY) from the set:</p> <p> key MY-CLASS.&id ({My-object-set}),</p> <p> -- Has to be the PARAMETERS for the object with KEY:</p> <p> parms MY-CLASS.&any-type ({My-object-set} {@key})</p> <p>}</p> <p>Variable type value fields and value set fields are out of the scope of this reference card</p>	<p style="text-align: center;">PARAMETERIZATION</p> <p>All assignments defining reference names (type, value, class definitions, object definitions, object set) can be given a dummy parameter list. Here we have two dummy parameters – normal-priority and Parameter.</p> <p>Invoke-message {INTEGER:normal-priority, Parameter} ::= SEQUENCE {</p> <p> component1 INTEGER DEFAULT normal-priority,</p> <p> component2 Parameter }</p> <p>Now we define our messages as a choice of two possibilities that differ only in the default priority and the Type that is to be used:</p> <p>Messages ::= CHOICE {</p> <p> first Invoke-message { low-priority, Type1 },</p> <p> second Invoke-message { high-priority, Type2 },</p> <p> ... }</p> <p>Messages ::= CHOICE { -- This is what the above expands to</p> <p> first SEQUENCE {</p> <p> component1 INTEGER DEFAULT low-priority,</p> <p> component2 Type1 },</p> <p> second SEQUENCE {</p> <p> component1 INTEGER DEFAULT high-priority,</p> <p> component2 Type2 },</p> <p> ... }</p> <p style="text-align: center;">ENCODINGS</p> <p>Bit-wide PER: A compact binary encoding transferring the minimum information needed to identify a value.</p> <p>Byte-wide BER: A type-length-value (TLV) style of encoding</p> <p>DER: An encoding with only one way to encode a given value, used in security work.</p> <p>CER: Another security-related encoding, rarely used.</p> <p>XML XER: Encoding ASN.1 values as XML syntax.</p> <p>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.</p> <p>ECN An encoding control notation (ECN) is available to completely determine the encoding of ASN.1 values.</p>																																																																																																