ASN.1
Abstract Syntax Notation One

Christophe Borelly

IUT Béziers Dépt. R&T

http://www.borelly.net/

Christophe@Borelly.net

Généralités

ASN.1 est une notation symbolique flexible permettant de représenter des types abstraits et des valeurs.

Parmi les différentes normes issues du CCITT

- X.200 : Open Systems Interconnection (1984)
- X.208 : Abstract Syntax Notation One (1988)
- X.209 : Basic Encoding Rules (1988)

BER (Basic Encoding Rules)

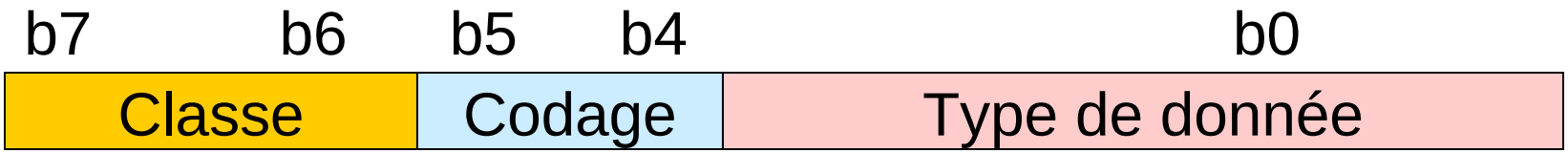
BER défini trois façon de coder un objet ASN.1 :

- Primitive, construite de longueur définie et construite de longueur indéfinie

Ces codages comportent 3 ou 4 parties :

- La partie fin est utilisée uniquement dans le cas du codage de longueur indéfinie

Codage de l'identifiant



Les bits 7 et 6 définissent la classe utilisée :

Universal (00) : le type est universel.

Application (01) : le type peut varier suivant l'application utilisée.

Private (10) : Signification du type spécifique à une entreprise donnée.

Context-specific (11) : Signification du type spécifique à un type structuré donné.

Le bit 5 défini le codage : **Primitif** (0) , **Construit** (1)

Les types simple ASN.1
(bits 4 à 0 de l'identifiant)

Type	Tag
INTEGER	0x02
BIT STRING	0x03
OCTET STRING	0x04
NULL	0x05
OBJECT IDENTIFIER	0x06
SEQUENCE (OF)	0x10
SET (OF)	0x11
PrintableString	0x13
T61String	0x14
IA5String	0x16
UTCTime	0x17

Codage primitif - Low-tag

Codes courts (Low-tag) de 0 à 0x30 (un octet seulement).

Le Tag correspond aux 5 bits (4 à 0) de l'identifiant.

Exemple pour coder un entier (Universal 00, Primitive 0, INTEGER 0x02) : 0x02

Pour coder une heure (Universal 00, Primitive 0, UTCTime 0x17) : 0x17

Codage primitif - High-tag

Codes longs (High-tag) supérieurs à 0x30 (deux ou plusieurs octets).

Le premier octet est obtenu en remplaçant la partie tag (5 bits de poids faible) par des 1. La suite est composée des octets représentant le tag en base 128 (du poids fort au poids faible) avec le minimum de digits. Le bit de poids fort étant mis à 1 pour tous les octets sauf le dernier.

Exemple pour coder (Universal 00, Primitive 0, Tag 134 = 128 + 6) : 0x1F 0x81 0x06

Codage de la longueur

Forme courte avec le bit 7 à 0 (valeur de 0 à 127).

Forme longue (de 2 à 127 octets) :

- Le premier octet à le bit 7 à 1 et le reste représente le nombre de digits de la valeur à coder.
- La suite est le codage base 256 de la valeur (digit de poids fort en premier).

Exemple : 326 = 256 + 70 : 0x01 0x46 : 2 digits

=> 0x82 0x01 0x46

Codage construit de longueur indéfinie

Le codage de longueur indéfinie n'est utilisé qu'en codage construit (bit 5 de l'identifiant à 1)

- La partie longueur est fixée à 0x80.
- La partie fin est fixée à 0x00 0x00.

Codage de la valeur

Contient :

- Simplement la valeur
- La concaténation des composants de la valeur
- Le codage du type sou jacent dans le cas d’un type dérivé d’un type implicite.

DER

Il y a plusieurs façons d'encoder un objet avec BER.

Le système DER (Distinguished Encoding Rules, section 8.7 de X.509) propose un codage unique à partir des règle de BER (utile pour la signature d'un objet ASN.1 par exemple).

INTEGER 0x02

Codage base 256 en complément à 2, digit de poids fort en premier.

Exemple pour 128

- DER : 0x02 0x02 0x00 0x80

Exemple pour -128

- DER : 0x02 0x01 0x80

Exemple pour -129

- DER : 0x02 0x02 0xFF 0x7F

Notation INTEGER

```
INTEGER [{id1(val1)… idX(valX)}]
```

Définition du type Version pour lequel v1988 est un identifiant de la valeur 0.

```
Version ::= INTEGER { v1988(0) }
Certificate ::= ...
   version Version DEFAULT v1988,
...
```

BIT STRING 0x03 (1)

Le premier octet de valeur contient le nombre de bits non utilisés dans le dernier octet (6 dans l'exemple).

On rajoute des bits de bourrage pour obtenir un multiple de 8 bits (Avec DER, on utilise des 0).

Exemple pour la clé 0110 1110 0101 1101 11

- DER : 0x03 0x04 0x06 0x6E 0x5D 0xC0

BIT STRING 0x03 (2)

Encodage construit avec deux BIT STRING : "0110 1110 0101 1101" + "11"

0x23 0x09

- 0x03 0x03 0x00 0x6E 0x5D
- 0x03 0x02 0x06 0xC0

OCTET STRING 0x04

Le codage de la suite d'octets 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF est le suivant :

0x04 0x08 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF

Le codage construit des suites 0x01 0x23 0x45 0x67 et 0x89 0xAB 0xCD 0xEF est le suivant :

0x24 0x0C

- 0x04 0x04 0x01 0x23 0x45 0x67
- 0x04 0x04 0x89 0xAB 0xCD 0xEF

Notation
OCTET STRING 0x04

```
OCTET STRING [SIZE ({size |
  size1..size2})]
```

Exemple de notation :

```
PBEParameter ::= SEQUENCE {
 salt OCTET STRING SIZE(8),
 iterationCount INTEGER }
```

NULL 0x05

Il n’y a pas de valeur, la longueur est donc de 0 octet.

Suivant le codage de la longueur, on peut avoir :

- 0x05 0x00
- 0x05 0x81 0x00

OBJECT IDENTIFIER 0x06

Un OID permet de représenter un objet par une suite de nombres entiers positifs val1, val2, ..., valX (par exemple dans la RFC 1213 qui définie les OID de la MIB, on a sysUpTime 1.3.6.1.2.1.1.3).

Le premier octet du codage contient 40 * val1 + val2.

Cette valeur est non ambiguë car :

- val1 ne peut prendre que les valeurs 0, 1 et 2
- val2 est limitée de 0 à 39 quand val1 est égal à 0 ou 1
- Le nombre de digits X est au moins égal à 2

Les octets suivants, si il y en a, représentent les valeurs val3, ..., valX en base 128 (digit de poids fort en premier), avec le bit 7 fixé a 1 pour tous les octets sauf pour le dernier digit.

OBJECT IDENTIFIER 0x06

Le codage de l'OID :

- RSA Data Security { 1 2 840 113549 }

40 * 1 + 2 = 42 : 0x2A

840 = 6 * 128 + 72 : 0x86 0x48

113549 = 6 * 1282 + 119 * 128 + 13 : 0x86 0xF7 0x0D

Soit le codage :

- 0x06 0x06 0x2A 0x86 0x48 0x86 0xF7 0x0D

Notation OBJECT IDENTIFIER

```
{ [id] comp1 ... compX }
compX = idX | idX(valX) | valX
```

Définition de l'OID mgmt = 1.3.6.1.2 et de mib-2 = 1.3.6.1.2.1.

```
iso OBJECT IDENTIFIER ::= { 1 }
mgmt OBJECT IDENTIFIER ::= { iso org(3)
  dod(6) internet(1) mgmt(2) }
mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }
```

Notation SEQUENCE

```
SEQUENCE {
   [id1] Type1 [{ OPTIONAL | DEFAULT val1}],
   ...,
   [idX] TypeX [{ OPTIONAL | DEFAULT valX}]
}
```

SEQUENCE indique une suite ordonnée de 1 ou plusieurs éléments.

```
Validity ::= SEQUENCE {
   start UTCTime,
   end UTCTime }
```

SEQUENCE 0x10

Le codage est de type construit et résulte de la concaténation du codage des éléments dans l'ordre de la liste.

Si un élément de type OPTIONAL ou DEFAULT est absent de la liste de valeurs, il n'est pas encodé.

Si la valeur d'un élément correspond à la valeur DEFAULT, il peut ou ne pas être encodé (non encodé en DER).

Notation SEQUENCE OF 0x10

`SEQUENCE OF Type`

SEQUENCE OF indique une suite de 0 ou plusieurs éléments identiques.

`RDNSequence ::= SEQUENCE OF RelativeDistinguishedName`

Codage identique à SEQUENCE

Notation SET 0x11

```
SET {
  [id1] Type1 [{ OPTIONAL | DEFAULT val1}],
  ...,
  [idX] TypeX [{ OPTIONAL | DEFAULT valX}]
}
```

SET indique une suite non ordonnée de 1 ou plusieurs éléments.

Codage identique à SEQUENCE sauf que pour DER, on trie les types par ordre alphabétique.

IA5String 0x16

International Alphabet 5 (ASCII)

Encodage DER "test@rsa.com" :

- 0x16 0x0C 0x74 0x65 0x73 0x74 0x40 0x72 0x73 0x61 0x2E 0x63 0x6F 0x6D

Encodage construit avec trois IA5String : "test" + "@" + "rsa.com"

0x36 0x12

- 0x16 0x04 0x74 0x65 0x73 0x74
- 0x16 0x01 0x40
- 0x16 0x07 0x72 0x73 0x61 0x2E 0x63 0x6F 0x6D

CHOICE

```
CHOICE {
  [id1] Type1, ..., [idX] TypeX
}
```

CHOICE indique une union de 1 ou plusieurs alternatives.

ANY

ANY [DEFINED BY id]

ANY indique que la définition est relative à un autre type.

```
AlgorithmIdentifier ::= SEQUENCE {
  algorithm OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL }
```

Types IMPLICIT

```
[[class] number] IMPLICIT Type
class = UNIVERSAL | APPLICATION | PRIVATE
```

Permet de donner un numéro spécifique (Tag) à un type donné autre que ANY. Le type de codage peut être primitif ou construit.

```
PrivateKeyInfo ::= SEQUENCE {
  version Version,
  privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
  privateKey PrivateKey,
  attributes [0] IMPLICIT Attributes OPTIONAL }
```

Types EXPLICIT

```
[[class] number] EXPLICIT Type
class = UNIVERSAL | APPLICATION | PRIVATE
```

Permet de donner un numéro spécifique (Tag) à un type ANY optionnel d'une SEQUENCE. Le type est uniquement construit.

```
ContentInfo ::= SEQUENCE {
  contentType ContentType,
  content [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL
}
```

Définition du type Name (X.501)

```
Name ::= CHOICE {
  RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::=
  SET OF AttributeValueAssertion

AttributeValueAssertion ::= SEQUENCE {
   AttributeType,
   AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY
```

Définition de types (X.520)

```
attributeType OBJECT IDENTIFIER ::=
  { joint-iso-ccitt(2) ds(5) 4 }

countryName OBJECT IDENTIFIER ::=
  { attributeType 6 }

organizationName OBJECT IDENTIFIER ::=
  { attributeType 10 }

commonUnitName OBJECT IDENTIFIER ::=
  { attributeType 3 }
```

Exemple (1)

Codage de la valeur de type Name :

countryName="US",**organizationName**="Example Organization",**commonName**="Test User 1"

countryName 2.5.4.6 : 40*2+5 = 85 : 0x55

- 0x06 0x03 0x55 0x04 0x06

organizationName 2.5.4.10 :

- 0x06 0x03 0x55 0x04 0x0A

commonName 2.5.4.3 :

- 0x06 0x03 0x55 0x04 0x03

Exemple (2)

Codage de la SEQUENCE construite *AttributeValueAssertion* : countryName="US" ("US" avec un type PrintableString 0x13)

0x30 0x09

- 0x06 0x03 0x55 0x04 0x06
- 0x13 0x02 0x55 0x53

Idem pour les deux autres parties.

Exemple (3)

Il faut ajouter le codage du SET OF *RelativeDistinguishedName* puis de la SEQUENCE OF *RDNSequence* et enfin du CHOICE *Name* (qui n'est rien d'autre que la SEQUENCE OF)

0x30 0x42 SEQUENCE OF

- 0x31 0x0B SET OF
  - 0x30 0x09 SEQUENCE
    - 0x06 0x03 0x55 0x04 0x06 countryName
    - 0x13 0x02 0x55 0x53 "US"

Exemple (4)

- 0x31 0x1D SET OF
  - 0x30 0x1B SEQUENCE
    - 0x06 0x03 0x55 0x04 0x0A organizationName
    - 0x13 0x14 0x45 0x78 0x61 0x6D 0x70 0x6C 0x65 0x20 0x4F 0x72 0x67 0x61 0x6E 0x69 0x7A 0x61 0x74 0x69 0x6F 0x6E "Example Organization"
- 0x31 0x14 SET OF
  - 0x30 0x12 SEQUENCE
    - 0x06 0x03 0x55 0x04 0x03 commonName
    - 0x13 0x0B 0x54 0x65 0x73 0x74 0x20 0x55 0x73 0x65 0x72 0x20 0x31 "Test User 1"

Références

http://luca.ntop.org/Teaching/Appunti/asn1.html

- **A Layman's Guide to a Subset of ASN.1, BER, and DER**

  An RSA Laboratories Technical Note

  Burton S. Kaliski Jr.

  Revised November 1, 1993