X.690

X.690 is an ITU-T standard specifying several ASN.1 encoding formats:

- Basic Encoding Rules (BER)
- Canonical Encoding Rules (CER)
- Distinguished Encoding Rules (DER)

The Basic Encoding Rules were the original rules laid out by the ASN.1 standard for encoding abstract information into a concrete data stream. The rules, collectively referred to as a *transfer syntax* in ASN.1 parlance, specify the exact octet sequences which are used to encode a given data item. The syntax defines such elements as: the representations for basic data types, the structure of length information, and the means for defining complex or compound types based on more primitive types. The BER syntax, along with two subsets of BER (the Canonical Encoding Rules and the Distinguished Encoding Rules), are defined by the ITU-T's X.690 standards document, which is part of the ASN.1 document series.

BER encoding

The format for Basic Encoding Rules specifies a self-describing and self-delimiting format for encoding ASN.1 data structures. Each data element is encoded as a type identifier, a length description, the actual data elements, and, where necessary, an end-of-content marker. These types of encodings are commonly called type-length-value or TLV encodings. This format allows a receiver to decode the ASN.1 information from an incomplete stream, without requiring any pre-knowledge of the size, content, or semantic meaning of the data.[1]

Encoding structure

The encoding of data does generally consist of four components which appear in the following order:

Identifier octets *Type*	Length octets *Length*	Contents octets *Value*	End-of-contents octets

The End-of-contents octets are optional and only used if the indefinite length form is used. The Contents octet may also be omitted if there is no content to encode like in the NULL type.

Identifier octets

Types

Data (especially members of sequences and sets and choices) can be tagged with a unique tag number (shown in ASN.1 within square brackets []) to distinguish that data from other members. Such tags can be implicit (where they are encoded as the TLV tag of the value instead of using the base type as the TLV tag) or explicit (where the tag is used in a constructed TLV that wraps the base type TLV). The default tagging style is explicit, unless implicit is set at ASN.1 module-level. Such tags have a default class of context-specific, but that can be overridden by using a class name in front of the tag.

The encoding of a choice value is the same as the encoding of a value of the chosen type. The encoding may be primitive or constructed, depending on the chosen type. The tag used in the identifier octets is the tag of the chosen type, as specified in the ASN.1 definition of the chosen type..

The following tags are native to ASN.1:

Types, universal class

Name	Value encodings	Tag number	
		Decimal	Hexadecimal
End-of-Content (EOC)	Primitive	0	0
BOOLEAN	Primitive	1	1
INTEGER	Primitive	2	2
BIT STRING	Both	3	3
OCTET STRING	Both	4	4
NULL	Primitive	5	5
OBJECT IDENTIFIER	Primitive	6	6
Object Descriptor	Both	7	7
EXTERNAL	Constructed	8	8
REAL (float)	Primitive	9	9
ENUMERATED	Primitive	10	A
EMBEDDED PDV	Constructed	11	B
UTF8String	Both	12	C
RELATIVE-OID	Primitive	13	D
Reserved		14	E
Reserved		15	F
SEQUENCE and SEQUENCE OF	Constructed	16	10
SET and SET OF	Constructed	17	11
NumericString	Both	18	12
PrintableString	Both	19	13
T61String	Both	20	14
VideotexString	Both	21	15
IA5String	Both	22	16
UTCTime	Both	23	17
GeneralizedTime	Both	24	18
GraphicString	Both	25	19
VisibleString	Both	26	1A
GeneralString	Both	27	1B
UniversalString	Both	28	1C
CHARACTER STRING	Both	29	1D
BMPString	Both	30	1E

Encoding

The identifier octets encode the element type as an ASN.1 tag, consisting of the class and number, and whether the contents octets represent a constructed or primitive value. Note that some types can have values with either primitive or constructed encodings. It is encoded as 1 or more octets.

Octet 1								Octet 2 onwards							
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
Tag class		P/C	Tag number (0–30)					N/A							
			31					More	Tag number						

In the initial octet, bit 6 encodes whether the type is primitive or constructed, bit 7–8 encode the class of the type, and bits 1–5 encode the tag number. The following values are possible:

Class	Value	Description
Universal	0	The type is native to ASN.1
Application	1	The type is only valid for one specific application
Context-specific	2	Meaning of this type depends on the context (such as within a sequence, set or choice)
Private	3	Defined in private specifications

P/C	Value	Description
Primitive (P)	0	The contents octets directly encode the element value.
Constructed (C)	1	The contents octets contain 0, 1, or more element encodings.

Long form

Where the identifier is not universal, its tag number may be too large for the 5-bit tag field, so it is encoded in further octets.

The initial octet encodes the class and primitive/constructed as before, and bits 1–5 are 1. The tag number is encoded in the following octets, where bit 8 of each is 1 if there are more octets, and bits 1–7 encode the tag number. The tag number bits combined, big-endian, encode the tag number. The least number of following octets should be encoded; that is, bits 1–7 should not all be 0 in the first following octet.

Length octets

There are two forms of the length octets: The definite form and the indefinite form.

First length octet

Form	Bits							
	8	7	6	5	4	3	2	1
Definite, short	0	Length (0–127)						
Indefinite	1	0						
Definite, long	1	Number of following octets (1–126)						
Reserved	1	127						

Definite form

This encodes the number of content octets and is always used if the type is primitive or constructed and data are immediately available. There is a short form and a long form, which can encode different ranges of lengths. Numeric data is encoded as unsigned integers with the least significant bit always first (to the right).

The **short form** consists of a single octet in which bit 8 is 0, and bits 1–7 encode the length (which may be 0) as a number of octets.

The **long form** consist of 1 initial octet followed by 1 or more subsequent octets, containing the length. In the initial octet, bit 8 is 1, and bits 1–7 (excluding the values 0 and 127) encode the number of octets that follow.[1] The following octets encode, as big-endian, the length (which may be 0) as a number of octets.

Long form example, length 435

Octet 1								Octet 2								Octet 3							
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	1
Long form	2 length octets							435 content octets															

Indefinite form

This does not encode the length at all, but that the content octets finish at marker octets. This applies to constructed types and is typically used if the content is not immediately available at encoding time.

It consists of single octet, in which bit 8 is 1, and bits 1–7 are 0. Then, 2 end-of-contents octets must terminate the content octets.

Contents octets

The contents octets encode the element data value.[1]

Note that there may be no contents octets (hence, the element has a length of 0) if only the existence of the ASN.1 object, or its emptiness, is to be noted. For example, this is the case for an ASN.1 NULL value.

CER encoding

CER (Canonical Encoding Rules) is a restricted variant of BER for producing unequivocal transfer syntax for data structures described by ASN.1. Whereas BER gives choices as to how data values may be encoded, CER (together with DER) selects just one encoding from those allowed by the basic encoding rules, eliminating rest of the options. CER is useful when the encodings must be preserved; e.g., in security exchanges.

DER encoding

DER (Distinguished Encoding Rules) is a restricted variant of BER for producing unequivocal transfer syntax for data structures described by ASN.1. Like CER, DER encodings are valid BER encodings. DER is the same thing as BER with all but one sender's options removed.

DER is a subset of BER providing for exactly one way to encode an ASN.1 value. DER is intended for situations when a unique encoding is needed, such as in cryptography, and ensures that a data structure that needs to be digitally signed produces a unique serialized representation. DER can be considered a canonical form of BER. For example, in BER a Boolean value of true can be encoded as any of 255 non-zero byte values, while in DER there is one way to encode a boolean value of true.

The most significant DER encoding constraints are:

1. Length encoding must use the definite form
    - Additionally, the shortest possible length encoding must be used
2. Bitstring, octetstring, and restricted character strings must use the primitive encoding
3. Elements of a Set are encoded in sorted order, based on their tag value

DER is widely used for digital certificates such as X.509.

BER, CER and DER compared

The key difference between the BER format and the CER or DER formats is the flexibility provided by the Basic Encoding Rules. BER, as explained above, is the basic set of encoding rules given by ITU X.690 for the transfer of ASN.1 data structures. It gives senders clear rules for encoding data structures they want to send, but also leaves senders some encoding choices. As stated in the X.690 standard, "Alternative encodings are permitted by the basic encoding rules as a sender's option. Receivers who claim conformance to the basic encoding rules shall support all alternatives".[1]

A receiver must be prepared to accept all legal encodings in order to legitimately claim BER-compliance. By contrast, both CER and DER restrict the available length specifications to a single option. As such, CER and DER are restricted forms of BER and serve to disambiguate the BER standard.

CER and DER differ in the set of restrictions that they place on the sender. The basic difference between CER and DER is that DER uses definitive length form and CER uses indefinite length form in some precisely defined cases. That is, DER always has leading length information, while CER uses end-of-contents octets instead of providing the length of the encoded data. Because of this, CER requires less metadata for large encoded values, while DER does it for small ones.

In order to facilitate a choice between encoding rules, the X.690 standards document provides the following guidance:

> The distinguished encoding rules is more suitable than the canonical encoding rules if the encoded value is small enough to fit into the available memory and there is a need to rapidly skip over some nested values. The canonical encoding rules is more suitable than the distinguished encoding rules if there is a need to encode values that are so large that they cannot readily fit into the available memory or it is necessary to encode and transmit a part of a value before the entire value is available. The basic encoding rules is more suitable than the canonical or distinguished encoding rules if the encoding contains a set value or set-of value and there is no need for the restrictions that the canonical and distinguished encoding rules impose.

Criticisms of BER encoding

There is a common perception of BER as being "inefficient" compared to alternative encoding rules. It has been argued by some that this perception is primarily due to poor implementations, not necessarily any inherent flaw in the encoding rules.[2] These implementations rely on the flexibility that BER provides to use encoding logic that is easier to implement, but results in a larger encoded data stream than necessary. Whether this inefficiency is reality or perception, it has led to a number of alternative encoding schemes, such as the Packed Encoding Rules, which attempt to improve on BER performance and size.

Other alternative formatting rules, which still provide the flexibility of BER but use alternative encoding schemes, are also being developed. The most popular of these are XML-based alternatives, such as the XML Encoding Rules and ASN.1 SOAP.[3] In addition, there is a standard mapping to convert an XML Schema to an ASN.1 schema, which can then be encoded using BER.[4]

Usage

Despite its perceived problems, BER is a popular format for transmitting data, particularly in systems with different native data encodings.

- The SNMP and LDAP protocols specify ASN.1 with BER as their required encoding scheme.
- The EMV standard for credit and debit cards uses BER to encode data onto the card
- The digital signature standard PKCS #7 also specifies ASN.1 with BER to encode encrypted messages and their digital signature or digital envelope.
- Many telecommunication systems, such as ISDN, toll-free call routing, and most cellular phone services use ASN.1 with BER to some degree for transmitting control messages over the network.
- GSM TAP (Transferred Account Procedures), NRTRDE (Near Real Time Roaming Data Exchange) files are encoded using BER. [1] (http://www.gsmworld.com/using/billing/potential.shtml)

By comparison, the more definite DER encoding is widely used to transfer digital certificates such as X.509.

See also

- Kerberos
- Packed Encoding Rules (PER, X.691)
- Structured Data eXchange Format (SDXF)
- Serialization

References

This article is based on material taken from the *Free On-line Dictionary of Computing* prior to 1 November 2008 and incorporated under the "relicensing" terms of the GFDL, version 1.3 or later.

1. Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf), ITU-T X6.90, 07/2002
2. Lin, Huai-An. “Estimation of the Optimal Performance of ASN.1/BER Transfer Syntax”. ACM Computer Communication Review. July 93, 45 - 58.
3. ITU-T Rec. X.892, ISO/IEC 24824-2
4. ITU-T X.694, ISO/IEC ISO/IEC 8825-5

External links

- RSA's 'A Layman's Guide to a Subset of ASN.1, BER, and DER ' (ftp://ftp.rsasecurity.com/pub/pkcs/ascii/layman.asc)
- ITU-T X.690, ISO/IEC 8825-1 (https://www.itu.int/rec/T-REC-X.690/)
- ITU-T X.892, ISO/IEC 24824-2 (https://www.itu.int/rec/T-REC-X.892/)
- ITU-T X.694, ISO/IEC ISO/IEC 8825-5 (https://www.itu.int/rec/T-REC-X.694/)
- PKCS #7 (http://www.rsasecurity.com/rsalabs/node.asp?id=2129)
- jASN1 (https://www.openmuc.org/asn1/) Java ASN.1 BER encoding/decoding library at openmuc.org, LGPL-licensed
- PHPASN1 (https://github.com/FGrosse/PHPASN1) PHP ASN.1 BER encoding/decoding library at github, GPL-licensed
- ASN1js (https://github.com/GlobalSign/ASN1.js) JavaScript ASN.1 BER encoding/decoding library at github, GPL-licensed
- Peter Gutmann's 'X.509 Style Guide' (http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt)