4.4.13.2 Нотация ASN.1

Семенов Ю.А. (ИТЭФ-МФТИ)
Yu. Semenov (ITEP-MIPT)

Одной из наиболее сложных систем сегодня являются открытые системы связи **OSI** (Open System Interconnection). OSI представляет собой достаточно формализованную стандартную архитектуру управления межкомпьютерными коммуникациями. Для описания этой системы была разработана абстрактный синтаксис нотаций ASN.1 (Abstract Syntax Notation; см. A Layman's Guide to a Subset of ASN.1, BER, and DER. Burton S. Kaliski Jr., RSA Data Security, Inc. Redwood City, CA, 1991). ASN.1 является формальным языком, который обладает двумя основными чертами.

ASN.1 представляет собой язык описания типов данных и их значений. В общем виде такое описание имеет вид:

data type value name | data type identifier ::= data value или {data type identifier (data value)}

Коме того, ASN.1 является языком программирования. Этот язык служит для описания MIB и может использоваться для модификации существующей или создания новой базы данных MIB для SNMP. Список наиболее распространенных ключевых слов ASN.1 для описания MIB SNMP включает в себя: *BEGIN, IDENTIFIER, END, OCTETS, STRING, SEQUENCE, INTEGER, STRING, OBJECT,OF, NULL, DEFINITIONS DESCRIPTION*

Макросы OBJECT-TYPE

Объекты MIB создаются макросами ASN.1. Макрос OBJECT-TYPE имеет формат:

data type value name OBJECT-TYPE

SYNTAX data type identifier

UNITS

ACCESS

STATUS

DESCRIPTION

REFERENCE

INDEX

DEFVAL

::= {data value}

- SYNTAX - определяет тип данных (простой, структурированный и т.д)
  ACCESS - задает уровень доступа к объекту (read only, read & write)
- STATUS - определяет статус описания объекта (текущий, устаревший и пр.)
- DESCRIPTION - опсывает роль или функцию управляемого объекта и способы его применения
- REFERENCE - опционное описание, характеризующее наследование объекта (указывает на родительский объект)
- INDEX - работает в случае, когда объект содержит список или таблицу и позволяет указать позицию в списке или таблице
- DEFVAL - опционная характеристика, указывающая значение объекта по умолчанию

Используемая в документах нотация легко читаема и понимаема, а в компактном кодовом представлении информация может использоваться коммуникационными протоколами. Неотъемлемой частью ASN.1 являются базовые правила кодирования **BER** (Basic Encoding Rules), которые позволяют определить большое разнообразие типов данных. BER описывает то, как представить или закодировать любую величину в рамках стандарта ASN.1. Все величины здесь представляются в виде последовательности 8-битных октетов. Восьмой бит октета всегда считается самым старшим. BER позволяет закодировать величину более чем одним способом. Имеется также поднабор правил кодирования **DER** (Distinguished Encoding Rules, описаны в документе X.509), которые определяют однозначные способы кодирования величин ASN.1.

Базовые правила обозначений метасинтаксиса ASN.1

Ниже приведены базовые правила обозначений метасинтаксиса ASN.1.

n	(полужирный курсив) обозначает переменную
[ ]	(квадратные скобки, напечатанные полужирным шрифтом) указывают на то, что терм является опционным.
{ }	(фигурные скобки, напечатанные полужирным шрифтом) группируют родственные термы.
|	(вертикальная черта, напечатанная полужирным шрифтом) выделяет альтернативные значения.
...	**(**многоточие, напечатанное полужирным шрифтом) обозначает повторения.
=	(знак равенства, напечатанный полужирным шрифтом) описывает терм как субтерм.

ASN.1 имеет четыре разновидности типов: простые типы, не имеющие компонент, структурные типы, имеющие компоненты, помеченные (tagged) типы, которые получаются из других типов, а также прочие типы, которые включают в себя типы CHOICE и ANY. Типам и значениям могут присваиваться имена с помощью оператора (**::=**). Эти имена в дальнейшем могут использоваться для определения других типов и величин.

Все типы ASN.1 кроме CHOICE и ANY имеют метки, которые состоят из класса и неотрицательного кода метки. Типы ASN.1 тождественны, если их числовые метки совпадают. Существует четыре класса меток.

universal	для типов, значения которых является неизменным для всех приложений. Эти типы

	определены в документе X.208.
application	для типов со значением, специфическим для приложений, таких как служба каталогов X.500. Типы двух разных приложений могут иметь одну и ту же метку и разные значения.
private	для типов, которые являются специфическими для данного предприятия.
content-specific	для типов со значением, специфическим для данного структурного типа.

Ниже приведена таблица 4.4.13.2.1 типов и их меток универсального класса.

Типы и их метки

Таблица 4.4.13.2.1. Типы и их метки

Тип	Комментарий	Цифровая метка (шестнадцатеричное)
INTEGER	Любое целое число	02
BIT STRING	Произвольная строка бит	03
OCTET STRING	Произвольная последовательность октетов	04
NULL	0	05
OBJECT IDENTIFIER	Последовательность целых компонент, идентифицирующих объект	06
SEQUENCE and SEQUENCE OF		10
SET and SET OF		11
PrintableString	Последовательность печатных символов	13
IA5String	Произвольная строка символов IA5 (ASCII)	16
UTCTime	Универсальное время (по Гринвичу; GMT)	17

ASN.1 типы и значения выражаются в нотации, близкой к используемой в языках программирования. Множественные пробелы и разрывы строк рассматриваются как один пробел. Комментарии выделяются парами дефисов или парой дефисов и переводом строки. Идентификаторы (имена значений и полей) и имена типов состоят из букв, цифр и пробелов. Идентификаторы начинаются со строчной буквы, а имена типов - с прописной.

В SMI (Structure of Management Information) не используется полный набор типов объектов, предусмотренный в ASN.1, разрешены только следующие типы примитивов: INTEGER, OCTET STRING, OBJECT IDENTIFIER и NULL.

Стандарт ASN.1 определяет форму представления информации и имен. Для строчных типов может быть введено ограничение на максимальный размер. В ASN.1 определено четыре структурированных типов:

Структурированные типы строк

SEQUENCE	упорядоченный набор из одного или более типов.
SEQUENCE OF	упорядоченный набор из нуля или более представителей данного типа.
SET	неупорядоченный набор из одного или более типов.
SET OF	неупорядоченный набор из нуля или более представителей данного типа.

Структурированные типы могут иметь опционные компоненты, в том числе со значениями по умолчанию.

Существуют типы помеченные явно и неявно. Неявно помеченные типы получаются из других типов путем изменения метки. Для неявной пометки используется ключевое слово IMPLICIT. Явно помеченные типы получаются из других типов путем добавления внешней метки. Помеченный явно тип - это структурированный тип, состоящий из одного компонента основного типа. Для явной пометки используется ключевое слово EXPLICIT. Пометка (тэгирование) весьма удобна для различия типов в пределах одного приложения.

Тип CHOICE обозначает объединение одного или более альтернатив. Тип ANY служит для обозначения произвольной величины для произвольного типа.

Правила BER

Правила BER определяют один или более способов представить любую величину в виде строки октетов. Существует три метода кодирования величин (в рамках BER): примитивный с известной длиной; конструктивный при известной длине и конструктивный при неизвестной длине. Выбор метода зависит от типа величины и от того, известна ли длина преобразуемой величины. Для простых не строчных типов используется примитивный метод кодирования. В каждом методе BER-кодирование имеет три или четыре части:

Identifier octets	Определяет класс и числовую метку значения, а также указывает, является ли метод примитивным или конструктивным.
Length octets	Для методов кодирования с известной длиной определяет число октетов содержимого.
Contents octets	Для примитивных методов с заданной длиной дает конкретное выражение значения.
End-of-contents octets	Для конструктивных методов с неопределенной длиной указывает на конец содержимого.

Примитивный метод кодирования с заданной длиной

Этот метод применим для простых типов и типов полученных из простых типов путем неявной пометки. Здесь необходимо, чтобы длина величины была известна заранее. *Октеты идентификатора* имеют два формата: для числовых меток от 0 до 31, и для числовых меток более 31. В первом случае биты 7 и 8 определяют класс (см. таблицу 4.4.13.2.2), бит 6 равен нулю, указывая на то, что метод кодирования *primitive*. Остальные биты используются для записи кода числовой метки. Во втором случае используется два или более октетов. В первом октете кодировка аналогична первому варианту за исключением того, что биты 1-5 содержат единицы.

Коды классов

Таблица 4.4.13.2.2. Коды классов

Класс	**Бит 8**	**Бит 7**
Универсальный	0	0
Прикладной	0	1
Контекстно-ориентированный	1	0
Частный	1	1

Для октетов длины примитивного метода имеется два формата: короткий (один октет для длин 0-127) и длинный (2-127 октетов). Для короткой формы 8-ой бит октета всегда равен нулю. Для длинной формы восьмой бит первого октета всегда равен 1, биты 1-7 содержат код числа дополнительных октетов длины. Старшая цифра записывается первой.

Конструктивный метод с заданной длиной

Этот метод используется для простых строчных и структурированных типов, типов, производных от простых строчных типов, и некоторых других. Здесь *октеты идентификатора* и октеты длины имеют формат, идентичный используемому примитивным методом, за исключением того, что бит 6 первого октета идентификатора равен 1.

Конструктивный метод кодирования с незаданной длиной

Метод используется для простых строчных типов, структурированных типов и типов, полученных из простых и структурированных типов с помощью неявной пометки. Октеты идентификатора идентичны предшествующему. Октет длины содержит код 80. Два октета конца содержательной части содержат 00 00.

Нотация типов, помеченных неявно, имеет вид:

[**[**class**]** number] IMPLICIT *Type*

class = UNIVERSAL | APPLICATION | PRIVITE

где *Type* - тип, class - опционное имя класса и number - цифровая метка (неотрицательное целое число).

Если имя класса отсутствует, тогда метка является контекстно-ориентированной. Такие метки могут появляться только в структурных компонентах или в типе CHOICE. Например:

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

Здесь исходным (порождающим) типом является Attributes, класс отсутствует (т.е. контекстно-ориентированный), а числовая метка равна нулю. Кодирование компоненты *attributes* величины PrivateKeyInfo осуществляется следующим образом.

Октеты идентификатора равны 80, если значение порождающей величины Attributes имеет конструктивное BER-кодирование. Октеты длины и содержимого строго соответствуют октетам

порождающей величины Attributes.

Непосредственная (явная) пометка используется для опционных компонент SEQUENCE с порождающим типом ANY и для компонент *version* типа *Certificate* (X.509 и RFC-1114). Нотация типов, помеченных явно, имеет формат.

[**[**class**]** number] EXPLICIT ***Type***

class = UNIVERSAL | APPLICATION | PRIVATE

где ***Type*** - тип, ***class*** - опционное имя класса, а ***number*** - числовая метка в пределах класса (неотрицательное целое число). Пример:

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

Тип ContentInfo имеет опционную компоненту content с явной контекстно-ориентированной меткой. Здесь порождающим типом является ANY DEFINED BY contentType, класс отсутствует, а числовая метка в пределах класса равна 0.

Другим примером может являться тип Certificate [X.509], имеющий компоненту с явной контекстно-ориентированной меткой (ключевое слово EXPLICIT опущено).

Certificate **::= …**
Version [0] Version DEFAULT v1988,
…

BER-кодирование величин, помеченных явно, является всегда конструктивным. Октеты содержимого идентичны соответствующим октетам порождающей величины. Например, BER-кодирование компоненты content величины ContentInfo имеет следующий вид.

Октеты идентификатора равны нулю, Октеты длины представляют длину BER-кодирования порождающей величины ANY DEFINED BY contentType.

Тип ANY

Тип ANY обозначает произвольную величину произвольного типа, где произвольный тип возможно определен при регистрации идентификатора объекта или является целочисленным индексом. Нотация типа ANY имеет формат:

ANY [DEINED BY ***identifier***]

где ***identifier*** - опционный идентификатор. Форма ANY DEINED BY ***identifier*** может появиться только в компоненте типа SEQUNCE или SET, для которой ***identifier*** определяет какую-то другую компоненту и эта компонента имеет тип INTEGER или OBJECT IDENTIFIER. В этой форме истинный тип задается величиной этой другой компоненты. Например, тип AlgorithmIdentifier [X.509] имеет компоненту типа ANY:

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

Здесь истинный тип компоненты parameter зависит от величины компоненты algorithm. Истинный тип будет определен при регистрации объекта величины идентификатора длякомпоненты algorithm.

Битовые строки

Тип BIT STRING обозначает произвольные битовые последовательности произвольной длины (включая ноль). Тип BIT STRING используется для цифровых сигнатур типа ExtendedCertificate или

Certificate [X.509]. Нотация BIT STRING имеет формат.

BIT STRING

Например, тип SubjectPublicKeyInfo имеет компоненту типа BIT STRING:

SubjectPublicKeyInfo **::=** SEQUENCE {
Algorithm AlgorithmIdentifier,
PublicKey BIT STRING }

BER-кодирование величины BIT STRING может быть примитивным или конструктивным. При примитивном кодировании первый октет содержимого несет в себе длину битовой строки в октетах. В последующих октетах записывается сама битовая последовательность. Процедура кодирования может включать в себя дополнение битовой строки до целого числа октетов нулями (если это необходимо). Строка делится на октеты.

При конструктивном кодировании октеты содержимого представляют собой соединение последовательности субстрок, только последняя из которых содержит код длины, выраженный в октетах. Например, при BER-кодировании значения BIT STRING “0111 1101 1001 1111 11” может быть представлена в одном из следующих видов, в зависимости от выбора схемы дополнения до целого числа октетов, от формата октетов длины и от метода кодирования примитивный/ конструктивный).

03 04 06 7D 9F C0	DER-кодирование
03 04 06 7D 9F E0	Дополнение кодом “100000”
03 81 04 06 7D 9F C0	Длинная форма представления октетов длины
23 09 03 03 00 7D 9F 03 02 06 C0	Конструктивное кодирование “01111101 1001 1111” +”11”

Первый октет первых трех представлений содержат код типа (для BIT STRING = 03). Второй октет в первых двух представлениях - код октетов длины (ведь далее следует 4 октета). Третий октет первой и второй строк содержит число добавленных нулей до числа бит, кратного восьми. Во втором байте третей строки записана 8, что указывает на длинную форму представления октетов длины. Последующая 1 говорит о том, что использован один дополнительный октет длины. Код длины в четвертом примере записан также во втором октете (далее следует 9 октетов). В этом варианте битовая строка разбита на две субстроки, первая из которых имеет длину в два октета. DER-кодирование предполагает всегда примитивный метод представления с дополнением строки нулями до длины, кратной целому числу октетов.

Тип CHOICE

Этот тип служит для объединения одной или более альтернатив. Нотация типа CHOICE имеет формат.

CHOICE {
[***identifier***$_1$] ***Type***$_1$,
…,
[***identifier***$_n$] ***Type***$_n$ }

где ***identifier***$_1$, **…,** ***identifier***$_n$ являются опционными идентификаторами альтернатив, а типы ***Type***$_1$, …, ***Type***$_n$ - альтернативы. Идентификаторы нужны для документирования и не играют какой-либо роли при кодировании. Типы должны иметь определенные метки. Рассмотрим пример типа ExtendedCertificateOrCertificate, который относится к типу CHOICE.

ExtendedCertificateOrCertificate **::=** CHOICE {
certificate Certificate, -- X.509 certificate

extendedCertificate [0] IMPLICIT ExtendedCertificate }

Здесь идентификаторами для альтернатив являются certificate и extendedCertificate, а сами альтернативы представлены типами Certificate и [0] IMPLICIT ExtendedCertificate. BER-кодирование для типа CHOICE сводится к кодированию альтернатив. При этом октеты идентификатора для рассмотренного примера содержат код 30, если выбранная альтернатива certificate, и A0 - в случае ExtendedCertificate.

Строки IA5

Тип IA5String представляет любые последовательности IA5-символов (международный алфавит 5 - эквивалентно ASCII). Длина строки может быть любой, включая нуль. Этот тип используется для адресов электронной почты и неструктурированных имен. Нотация типа IA5String имеет простой формат.

IA5String

BER-кодирование величины IA5String может быть примитивным или структурированным. При примитивном кодировании октеты содержимого представляют собой символы IA5 в ASCII-кодов. При конструктивном кодировании октеты содержимого представляют собой соединение ряда IA5-субстрок. Рассмотрим примеры представления значения IA5-строки “test1@rsa.com”.

12 0D 74 65 73 74 31 40 72 73 61 2E 63 6F 6D DER-кодирование	
Длинная форма октетов длины	
32 13	Конструктивное
12 05 74 65 73 74 31	кодирование: “test1”
12 01 40	+ “@” +
12 07 72 73 61 2E 63 6F 6D	“rsa.com”

DER-кодирование

DER-кодирование является всегда примитивным, октеты содержимого идентичны случаю BER-кодирования.

Целое

Тип INTEGER представляет любые целые числа (положительные, отрицательные или 0). Тип INTEGER используется для номеров версий, криптографических параметров (показателей, модулей) и типов RSAPublicKey, RSAPrivatKey, DHParameter PBEParameter. Нотация типа INTEGER имеет формат:

INTEGER **[**{***identifier***$_1$(***value$_1$***)** … ***identifier***$_n$(***value$_n$***) }]**

где ***identifier***$_1$ … ***identifier***$_n$ являются опционными идентификаторами, а ***value$_1$***… ***value$_n$*** опционные целые значения. Например, Version RFC-1114 относится к целому типу со значением:

Version ::= INTEGER { 1988(0) }

Идентификатору v1988 поставлено в соответствие значение 0. Тип *Certificate* RFC-1114 использует идентификатор *v1988*для присвоения значения по умолчанию компоненту *version*:

Certificate
version Version DEFAULT v1988,
…

BER-кодирование значения INTEGER является всегда примитивным. Октеты содержимого представляют значение целого по модулю 256 в форме дополнения по модулю 2. Старшая цифра

является первой. Значение нуль кодируется одним октетом 00. Примеры BER-кодирования (совпадающего в данном случае с DER-кодированием) представлены в таблице 4.4.13.2.3.

Таблица 4.4.13.2.3. Примеры BER-кодирования

Значение целого	BER-код
0	02 01 00
127	02 02 00 7F
128	02 02 00 80
256	02 02 01 00
-128	02 01 80
-129	02 02 FF 7F

NULL

Тип NULL обозначает нулевую величину и предназначен для использования в качестве параметра алгоритмов. Нотация для типа NULL имеет формат:

NULL

Кодирование для типа NULL является всегда примитивным, октеты содержимого пусты. Например, BER-представление значения NULL может иметь одну из приведенных ниже форм (зависит от используемого представления октетов длины).

05 00
05 81 00

DER-кодирование типа NULL является также примитивным и совпадает с первой строкой приведенного выше примера.

Объектные идентификаторы

Тип OBJECT IDENTIFIER служит для обозначения дентификаторов, которые представляют собой последовательность целочисленных компонент, которые идентифицируют такие объекты, как алгоритм или атрибут имени каталога. Значение OBJECT IDENTIFIER может содержать любое число неотрицательных компонент. Этот тип не относится в числу строчных. Значения OBJECT IDENTIFIER присваиваются при регистрации.

Тип OBJECT IDENTIFIER используется для идентификации содержимого ContentInfo, алгоритмов в X.509 (AlgorithmIdentifier) и атрибутов Attribute и AttributeValueAssertion (X.501). Нотация OBJECT IDENTIFIER имеет формат.

OBJECT IDENTIFIER

Нотация величины OBJECT IDENTIFIER имеет вид:

{ [***identifier***] $component_1 \ldots component_n$}
$\boldsymbol{component_i = identifier_i \mid identifier_i\ (value_i) \mid value_i}$

где $\boldsymbol{identifier, identifier_1, \ldots identifier_n}$ являются идентификаторами, а $\boldsymbol{value_1 \ldots, value_n}$ - опционные целые числа. Идентификаторы без целых значений могут встретиться только для объектов, описанных в X.208.

Например, нижеприведенные величины объектных идентификаторов присвоены RSA DATA Security, Inc.

{ iso(1) member-body(2) 840 113549 }
{ 1 2 840 113549 }

В таблице 4.4.13.2.4 представлены некоторые объектные идентификаторы и их значения.

Таблица 4.4.13.2.4. Некоторые объектные идентификаторы и их значения

Величина объектного идентификатора	Назначение
{ 1 2 }	Члены ISO
{ 1 2 840 }	US (ANSI)
{ 1 2 840 113549}	RSA Data Security, Inc.
{ 1 2 840 113549 }	RSA Data Security, Inc. PKCS (Public Key Cryptography Standard)
{ 2 5 }	Служба каталогов (X.500)
{ 2 5 8 }	Служба каталогов - алгоритмы

BER-кодирование OBJECT IDENTIFIER является всегда примитивным. Октеты содержимого представляют собой объединение n-1 строки октетов, где n число компонент объектного идентификатора. Каждая октетная строка несет в себе целое число по модулю 128 (старшая часть первая). 8-ой бит каждого октета, кроме последнего, равен 1. Пусть $value_1$, …, $value_n$ целые значения компонентов объектного идентификатора. Тогда n-1 субидентификаторов, из которых формируется октетная строка, будут иметь следующий вид.

1. Первый субидентификатор равен $40value_1 + value_2$. (значение $value_1$ лежит в пределах 0-2 включительно, а $value_2$ в интервале 0-39, когда $value_1$ равна 0 или 1.
2. i-ый субидентификатор равен $value_{i+1}$ *; 2 J iJ n-1.*

Например, субидентификаторы объектного идентификатора RSA Data Security, Inc. равны 42 = 40r 1 + 2, 840, 113549 и 1. В шестнадцатеричном представлении BER-код этого объектного идентификатора имеет вид:

06 07 2A 86 48 86 F7 0D 01

DER-кодирование в данном случае совпадает с BER.

Строки октетов

Тип OCTET STRING служит для представления произвольных последовательностей октетов. Значение OCTET STRING может иметь любую длину, включая нуль. OCTET STRING используется для представления сообщений, включая зашифрованные, а также для типа PBEParameter. Нотация типа OCTET STRING имеет формат.

OCTET STRING **[**SIZE ({***size*** **|** $size_1$**..**$size_2$})**]**

где ***size, size***$_1$ ***и size***$_2$ опционные ограничения размера. В форме OCTET STRING SIZE(***size***) строка октетов должна иметь октеты ***size***. В формате OCTET STRING SIZE($size_1$ **..** $size_2$) строка должна

содержать число октетов между ***size*$_1$** и ***size*$_2$.** Например, тип PBEParameter имеет компоненту типа OCTET STRING:

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

Здесь размер компоненты salt всегда равен 8 октетам. BER-кодирование типа OCTET STRING может быть примитивным или конструктивным. При примитивном кодировании октеты содержимого несут в себе октеты строки с первого по последний. При конструктивном кодировании содержимое октетов представляет собой последовательное объединение субстрок значения OCTET STRING. Например, BER-код значения OCTET STRING 01 23 45 67 89 AB CD EF может иметь один из следующих видов, в зависимости от формата октетов длины и вида кодирования (примитивное/конструктивное).

04 08 01 23 45 67 89 AB CD EF	DER-кодирование
04 81 08 01 23 45 67 89 AB CD EF	Длинный формат октетов длины
24 0C	Конструктивное кодирование
04 04 01 23 45 67	“01 23 45 67” + “89 AB CD EF”
04 04 89 AB CD EF	

Строки печатных символов

Тип PrintableString предназначен для описания произвольных последовательностей печатных символов из набора:

A, B,…,Z
a,b,…,z
0,1,…,9
(пробел) ‘ () +, - . / : = ?

Этот тип используется для представления атрибутов имен (X.520). Нотация типа PrintableString имеет вид:

PrintableString

BER-кодирование значения PrintableString может быть примитивным или конструктивным. При примитивном кодировании печатных символов байты содержимого несут в себе строки октетов печатных ASCII-кодов. При конструктивном кодировании содержимое октетов представляет собой последовательное объединение субстрок. Например, BER-код значения PrintableString “Test User 1” может быть представлено одним из ниже приведенных способов.

13 0B 54 65 73 74 20 55 73 65 72 20 31	DER-кодирование
13 81 0B 54 65 73 74 20 55 73 65 72 20 31	Длинная форма октетов длины
33 0F	Конструктивная форма,
13 05 54 65 73 74 20	“Test” + “User 1”
13 06 55 73 65 72 20 31	

Тип SEQUENCE

Тип SEQUENCE обозначает упорядоченную последовательность одного или более типов. Нотация типа SEQUENCE имеет вид:

SEQUENCE {
[*identifier*$_1$] ***Type*$_1$** **[**{OPTIONAL | DEFAULT ***value*$_1$**}**]**,

…,
[*identifier$_n$*] *Type$_n$* [{OPTIONAL | DEFAULT ***value$_n$***}**]**,

где ***identifier$_1$* , …, *identifier$_n$*** являются опционными идентификаторами компонентов, ***Type$_1$* , …, *Type$_n$*** - типы компонентов, а ***value$_1$* ,…, *value$_n$*** опционные значения компонентов по умолчанию.
Квалификатор OPTIONAL указывает на то, что значения компонентов являются опционными. Квалификатор DEFAULT говорит о том, что величина компонента является опционной и ей присваивается определенное значение, если компонент отсутствует. Например, тип Validity [X.509] относится к типу SEQUENCE и имеет два компонента.

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

Здесь start и end являются идентификаторами компонент, а типом компонент служит UTCTime. BER-кодирование значения SEQUENCE является всегда конструктивным. Октеты содержимого представляют последовательное объединение BER-кодов значений компонентов последовательности.

Тип SEQUENCE OF

Тип SEQUENCE OF обозначает упорядоченную последовательность из нуля или более компонентов данного типа, используется для имен в X.501. Нотация SEQUENCE OF имеет вид:

SEQUENCE OF ***Type***

Так например, тип RNDSequence состоит из нуля или более компонент типа RelativeDistinguishedName.

RNDSequence **::=** SEQUENCE OF RelativeDistinguishedName

BER-кодирование SEQUENCE OF является всегда конструктивным. Октеты содержимого представляют собой объединение BER-кодов значений элементов последовательности в порядке их расположения. DER-кодирование имеет тот же формат.

Тип SET

Тип SET представляет собой неупорядоченное объединение из одного или более типов. Нотация типа SET имеет вид.

SET {
[*identifier$_1$*] *Type$_1$* *Type$_1$* [{OPTIONAL | DEFAULT ***value$_1$***}**]**,
…,
[*identifier$_n$*] *Type$_n$* [{OPTIONAL | DEFAULT ***value$_n$***}**]**,

где ***identifier$_1$* , …, *identifier$_n$*** являются опционными идентификаторами компонентов, ***Type$_1$* , …, *Type$_n$*** - типы компонентов, а ***value$_1$* ,…, *value$_n$*** опционные значения компонентов по умолчанию.
Квалификатор OPTIONAL указывает на то, что значения компонентов являются опционными. Квалификатор DEFAULT говорит о том, что величина компонента является опционной и ей присваивается определенное значение, если компонент отсутствует.

BER-кодирование для типа SET является всегда конструктивным. Октеты содержимого представляют последовательное объединение BER-кодов значений компонентов набора.

Тип SET OF

Тип SET OF представляет неупорядоченный набор, состоящий из нуля или более компонентов заданного типа и предназначенный для атрибутов PKCS (Public-Key Cryptography Standard) и имен X.501. Нотация типа SET OF имеет вид:

SET OF ***Type***

где ***Type -*** тип. Так тип RelativeDistinguishedName состоит из нуля или более компонентов типа AttributeValueAssertion.

RelativeDistinguishedName **::=** SET OF AttributeValueAssertion

BER-кодирование типа SET OF является всегда конструктивным. Октеты содержимого представляют собой объединение BER-кодов величин компонентов в порядке их исходного расположения. DER-кодирование также является всегда конструктивным, октеты содержимого представляются как и в случае BER-кодировки. Но компоненты лексикографически упорядочиваются.

Тип UTCTime

Тип UTCTime служит для обозначения универсального местного времени с привязкой по Гринвичу (GMT). Значение UTCTime характеризует местное время с точностью минут или секунд и временной сдвиг по отношению к GMT. Оно может иметь следующие формы:

YYMMDDhhmmZ
YYMMDDhhmm+hh`mm`
YYMMDDhhmm-hh`mm`
YYMMDDhhmmssZ
YYMMDDhhmmss+ hh`mm`
YYMMDDhhmmss- hh`mm`

где

YY	младшие две цифры года
MM	код месяца (01 - 12)
DD	код дня (01 - 31)
hh	код часа (00 - 23)
mm	код минут (00 - 59)
ss	код секунд (00 - 59)
Z	означает местное время по Гринвичу, + указывает на то, что местное время отстает от GMT, а - указывает на то, что местное время опережает GMT.
hh`	абсолютное значение смещения по отношению к GMT в часах
mm`	абсолютное смещение по отношению к GMT в минутах.

UTCTime используется для определения типа Validity [X.509]. Нотация типа UTCTime имеет вид.

UTCTime

BER-кодирование значения UTCTime может быть примитивным или конструктивным. При примитивном кодировании октеты представляют собой символы строки в виде ASCII-кодов. При конструктивном кодировании октеты образуют объединение BER-кодов последовательных субстрок. В качестве примера рассмотрим варианты представления времени 4:45:40 после полудня 6 мая 1991 года (Pacific Daylight Time) в виде величины UTCTime.

“910506164540-0700”
“910506234540Z”

Это представление эквивалентно следующим BER-кодам:

17 0D 39 31 30 35 30 36 32 33 34 35 34 30 5A
17 11 39 31 30 35 30 36 31 36 34 35 34 30 2D 30 37 30 30

DER-кодирование типа UTCTime всегда является примитивным.

Ниже приводится пример ASN.1 нотации имен типа X.501.

Name **::=** CHOICE { RNDSequence }

RNDSequence **::=** SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName **::=** SET OF AttributeValueAssertion
AttributeValueAssertion **::=** SEQUENCE { AttributeType, AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY

Тип Name идентифицирует объект в каталоге X.500. Name имеет тип CHOICE с одной альтернативой RNDSequence. Тип RNDSequence указывает проход по дереву каталогов X.500, начиная с корневой части. RNDSequence имеет тип SEQUENCE OF, состоящий из нуля или более компонентов RelativeDistinguishedName. Тип RelativeDistinguishedName присваивает уникальное имя объекту на дереве каталога. RelativeDistinguishedName имеет тип SET OF, состоящий из нуля или более компонентов AttributeValueAssertion. Тип AttributeValueAssertion присваивает значение атрибуту имени (например, определяющее принадлежность к стране). AttributeValueAssertion имеет тип SEQUENCE, состоящий из двух компонентов AttributeType и AttributeValue. AttributeType идентифицирует атрибут с помощью объектного идентификатора. AttributeValue присваивает значение атрибуту. Ниже предлагается пример DER-кодирования типа Name. В качестве имени использовано RSA Data Security, Inc. NOTARY (отдел Internet Privacy Enhanced Mail).

(root)
|
countryName = ”US”
|
organizationName = ”RSA Data Security, Inc.”
|
organizationalUnitName = ”NOTARY”

Каждый уровень соответствует одному значению RelativeDistinguishedName, в выбранном случае они имеют только одно значение AttributeValueAssertion. Значение AttributeType помещается до знака равенства, а AttributeValue (строка печатных символов с заданным типом атрибута) - после знака равенства.

Тип атрибута

DER-кодирование трех значений AttributeType согласно примитивному методу с заданной длиной дает следующие строки октетов.

06 03 55 04 06 countryName
06 03 55 04 0A organizationName
06 03 55 04 0B organizationalUnitName

Здесь countryName, organizationName и organizationalUnitName являются типами атрибутов X.520, которые имеют следующие определения.

AttributeType OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }
countryName OBJECT IDENTIFIER **::=** { AttributeType 6 }
organizationName OBJECT IDENTIFIER **::=** { AttributeType 10 }
organizationalUnitName OBJECT IDENTIFIER **::=** { AttributeType 11 }

Октеты идентификатора имеют формат с меткой, так как метка равна 6 для OBJECT IDENTIFIER. Биты 8 и 7 равны 0, указывая на универсальный класс, бит 6 также равен 0, что говорит о примитивном методе кодирования. Октеты длины ориентированы на короткую форму представления. Октеты содержимого представляют собой объединения трех-октетных строк, полученных из субидентификаторов: 85 = 40x2 + 5; 4 и 6, 10 или 11.

Значение атрибута

DER-кодирование трех значений AttributeValue в соответствии с примитивным методом при заданной длине дает следующие строки:

13 02 55 53	“US”
13 17 52 53 41 20	“RSA
44 61 74 61 20	Data
53 65 63 75 72 69 74 79 2C 20	Security,
49 6E 63 2E	Inc.”

Метка равна 19 (PrintableString) биты 8 и 7 равны 0, указывая на универсальный класс, бит 6 также равен 0, отмечая примитивный метод кодирования. Октеты длины имеют “короткий” формат, а октеты содержимого являются ASCII-представлением значения атрибута.

Атрибут ValueAssertion

DER-кодирование трех значений AttributeValueAssertion в соответствии с конструктивным методом дает следующие строки октетов.

30 09	
06 03 55 04 06	countryName = “US”
13 02 55 53	
30 1E	
06 03 55 04 0A	
13 17 52 53 41 20	organizationName = “RSA Data Security,
44 61 74 61 20	Inc.”
53 65 63 75 72 69 74 79 2C 20	
49 6E 63 2E	
30 0D	
06 03 55 04 0B	organizationalUnitName = “NOTARY”
13 06 4E 4F 54 41 52 59	

Октеты идентификатора используют короткий формат (low-octet form), так как для SET OF метка равна 17. Биты 8 и 7 равны 0 (универсальный класс), а бит 6 равен 1 (конструктивное кодирование). Октеты длины следуют “короткому” формату, а октеты содержимого представляют собой объединение DER-кодов компонент attributeType и attributeValue.

RelativeDistinguishedName</p>

DER-кодирование трех значений RelativeDistinguishedName, выполняемое согласно конструктивному методу, дает следующие строки октетов.

31 0B

30 09 … 55
53

31 20

30 1E … 63
2E

31 0F

30 0D … 52
59

Октеты идентификатора используют короткий формат (low-octet form), так как для SET OF метка равна 17. Биты 8 и 7 равны 0 (универсальный класс), а бит 6 равен 1 (конструктивное кодирование). Октеты длины следуют “короткому” формату, а октеты содержимого представляют собой объединение DER-кодов компонент AttributeValueAssertion.

RDNSequence

DER-кодирование значений RDNSequence, выполняемое согласно конструктивному методу при заданной длине, дает следующие строки октетов.

30 40
31 0B … 55 53
31 20 … 63 2E
31 0F … 52 59

Октеты идентификатора используют короткий формат (low-octet form), так как для SEQUENCE OF метка равна 16. Биты 8 и 7 равны 0 (универсальный класс), а бит 6 равен 1 (конструктивное кодирование). Октеты длины следуют “короткому” формату, а октеты содержимого представляют собой объединение DER-кодов трех компонент RelativeDistinguishedName в порядке их следования.

Name

DER-кодирование значений Name выполняется аналогично значениям RDNSequence и выдает следующие результаты.

30 40

31 0B

30 09

06 03 55 04 06
13 02 55 53

31 20

30 1E

06 03 55 04 0A
13 17 52 53 41 20 44 61 74 61 20 53 65 63 75 72 69

74 79 2C 20 49 6E 63 2E

31 0F

30 0D

06 03 55 04 0B
13 06 4E 4F 54 41 52 59

Previous: 4.4.13.1 Управляющая база данных MIB **UP:** 4.4.13 Протокол управления SNMP