ASN.1 is getting sexy again!
or
ASN.1, XML, and Fast Web Services.
Whoops - did I mean that?
So - How Sexy?

- XML is **sexy**!
- Multiple layers of XML support in ASN.1
- Web Services is even more **sexy**!
- Fast Web Services are coming soon!
ASN.1 lineage

- ASN.1 was born around 1982
- First ASN.1 Standard (CCITT X.409) in 1984
- Borne from X.400 (Mother with an early child, and the e-mail standard the world *should* have had!)
- Fathered by X.500 (Certified insane at birth, but totally secure)
- Grand-parents (OSI) died prematurely and are not discussed in polite conversation today
- Married XML, and begat Fast Web Services
A shorter history of protocol specification
The Montagues and the Capulets

(Contending Philosophies)

(With apologies to William Shakespeare)
Understanding of protocol specification techniques

- 1.5 billion seconds ago ..... Computers started to communicate
- Major advances every 150 million seconds
- There was a need for
  - A means of syntax (data structure) specification
  - Procedure (sequence) specification
  - Test suite specification
  - Validation
- And tools to support rapid implementation!
The Montagues and Capulets

- A long and on-going civil dispute
- Montagues => Binary-based specification
- Capulets => Character-based specification

With apologies to William Shakespeare and to those from a non-UK culture!
The stone-age Montagues

Diagrams of bits and bytes - e.g. IPv4
(The earliest approach, simple and clear, but focusing totally on the bits-on-the-line.)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Version | IHL | Type of Service | Total Length |
| Identification | Flags | Fragment Offset |
| Time to Live | Protocol | Header Checksum |
| Source Address |
| Destination Address |
| Options | Padding |

Tool support not possible - but see ECN discussion.
Extensibility support crude - based on reserved fields.
The stone-age Capulets

- Simple “command lines” – in ASCII!
- Three character mnemonics and error codes (eg “200 OK”)
- Simple comma-separated parameters
- Good for simple dialogues
- Extensibility by adding commands in V2, with unknown commands ignored by V1 systems
The Bronze Age Montagues invent TLV and Tabular Notation

- Each PDU and each parameter has an ID (or Type), a Length, and a Value
- Tables list each parameter: Tabular Notation

<table>
<thead>
<tr>
<th>Parameter ID</th>
<th>Length</th>
<th>Optionality</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1 octet</td>
<td>Mandatory</td>
<td>See para 14.2</td>
</tr>
<tr>
<td>Priority</td>
<td>1 octet</td>
<td>Optional (default 0)</td>
<td>See para 14.3</td>
</tr>
<tr>
<td>Called address</td>
<td>Variable</td>
<td>Mandatory</td>
<td>See para 14.4</td>
</tr>
<tr>
<td>Calling address</td>
<td>Variable</td>
<td>Mandatory</td>
<td>See para 14.5</td>
</tr>
<tr>
<td>Additional information</td>
<td>Variable</td>
<td>Optional</td>
<td>See para 14.6</td>
</tr>
</tbody>
</table>

And Yuck – we are still going this route in 2003!
Tabular Notation and TLV was a breakthrough – should have been patented!

- Extensibility was EXCELLENT.
- Version 1 systems just skipped (using TLV) anything they did not know.
- Tool-support, however, not possible.
- **But it was verbose!**

  *But not as verbose as the character-based encoding used by the Capulets!*
The Capulets’ main concern was with precise specification of correct syntax.

This was the dawning of Backus Naur Form (BNF).

This potentially allowed more complex information to be specified in a “command”.

But it never really made it to the modern era of automatic mapping to Java, C++ etc.
150 Million seconds after the Bronze Age

Recognition of:
- Separation of abstract and transfer syntax
  (This is jargon for “content definition” and “encoding” or “syntax”)
- Encoding rules

- ASN.1 specs define a de facto API (message content)
- Tools emerge to support the transformation of ASN.1 to an API, and the encoding of data across that API

Profits for all! ASN.1 gets really Sexy!
ASN.1 deployment

- Wide use in a large range of industries:
  - Keeping the lights burning
  - Portable phones – we need them
  - Birthday presents on time
  - Traffic lights
  - Aircraft fly safely
  - Multimedia standards

- Many other industrial sectors
  - Most recently, biometrics
Without ASN.1:

- The lights go out!
- Portable phones don’t work!
- Parcels get lost!
- Traffic lights fail!
- Aircraft fall from the sky!
- Your impending marriage suffers as NetMeeting fails!

On second thoughts – it might be a better world?
The emergence of ECN

- Ambitious
- Use ASN.1 with a formal encoding notation to define any (binary) protocol
- ETSI-funded
- Took off slowly
- But very much still of interest
- Not the subject for today, but gives transition from ad hoc binary to XML encoding
300 Million seconds later, the Capulets develop XML

- Focus still on what is correct syntax, not content
  (This is still bad. What is syntax variation and what is a difference in the message? Covert channels.)
- Came out of SGML and HTML
- The “X” does not mean eXtensibility”
- Essentially a TLV style of encoding, but with human readable “<Start>…. </End>” wrappers
- Rapidly gained popularity! Idiots can understand it! Oh dear!
And finally, after another Million seconds

- ASN.1 develops XML Encoding Rules
- “Coloring” added to allow control of (for example) attributes v elements

Romeo and Juliet marry!
EXAMPLES

(If you can’t understand the examples at first glance, something is wrong!)
A simple invoice

Invoice ::= SEQUENCE {
  number INTEGER,
  name UTF8String,
  details SEQUENCE OF
    line-item LineItemPair,
  charge REAL,
  authenticator BIT STRING
}

LineItemPair ::= SEQUENCE {
  part-no INTEGER,
  quantity INTEGER
}
LinItemPair in XSD!

\[
\text{LinItemPair ::= SEQUENCE} \{ \\
\text{part-no INTEGER,} \\
\text{quantity INTEGER} \}
\]

How sexy is that – half the size!
An example Invoice (1)

<Invoice>
  <number>32950</number>
  <name>funny-name with &lt;</name>
  <details>
    <line-item>
      <part-no>296</part-no>
      <quantity>2</quantity>
    </line-item>
  </details>
</Invoice>
An example Invoice (2)

Continuation

<line-item>
  <part-no>4793</part-no>
  <quantity>74</quantity>
</line-item>

</details>

<charge>397.65</charge>

<authenticator>
  EFF8 E976 5403 629F
</authenticator>

</Invoice>
A base-ball card defined

BBCard ::= SEQUENCE {
  name     IA5String (SIZE (1..60)),
  team     IA5String (SIZE (1..60)),
  age      INTEGER (1..100),
  position IA5String (SIZE (1..60)),
  handedness ENUMERATED {
    left-handed (0),
    right-handed (1),
    ambidextrous (2) },
  batting-average REAL }

---

ASN.1 consortium
A base-ball card value in XML syntax

<BBCard>

  <name>Jorge Posada</name>
  <team>New York Yankees</team>
  <age>29</age>
  <position>C</position>
  <handedness><right-handed/></handedness>
  <batting-average>0.277</batting-average>

</BBCard>
"Coloring" for different XML syntax

BBCard ::= SEQUENCE {
  name [ATTRIBUTE] IA5String (SIZE (1..60)),
  team [ATTRIBUTE] IA5String (SIZE (1..60)),
  age INTEGER (1..100),
  position IA5String (SIZE (1..60)),
  handedness [TEXT] ENUMERATED {
    left-handed (0),
    right-handed (1),
    ambidextrous (2) },
  batting-average REAL }

ASN.1 consortium
The new XML syntax

<BBCard
    name = "Jorge Posada"
    team = "New York Yankees">
    <age>29</age>
    <position>C</position>
    <handedness>right-handed</handedness>
    <batting-average>0.277</batting-average>
</BBCard>
The C data-structure for the base-ball card

typedef struct BBCard {
    char name [61] ;
    char team [61] ;
    short age ;
    char position [61] ;
    enum {
        left_handed = 0,
        right_handed = 1,
        ambidextrous = 2,
    } handedness ;
    float batting_average ;
} BBCard ;
A personnel-record defined (1)

PersonnelRecord ::= SEQUENCE {
    name            Name,
    title           VisibleString,
    number          EmployeeNumber,
    dateOfHire      Date,
    nameOfSpouse    Name,
    children        SEQUENCE OF
        child      ChildInformation
        DEFAULT {} }
A personnel-record defined (2)

ChildInformation ::= SEQUENCE {
    name Name,
    dateOfBirth Date
}
Name ::= SEQUENCE {
    givenName VisibleString,
    initial VisibleString,
    familyName VisibleString
}
EmployeeNumber ::= INTEGER
Date ::= VisibleString -- YYYYMMDD
An example personnel-record (1)

```
<PersonnelRecord>
  <name>
    <givenName>John</givenName>
    <initial>P</initial>
    <familyName>Smith</familyName>
  </name>
  <title>Director</title>
  <number>51</number>
  <dateOfHire>19710917</dateOfHire>
  <nameOfSpouse>
    <givenName>Mary</givenName>
    <initial>T</initial>
    <familyName>Smith</familyName>
  </nameOfSpouse>
</PersonnelRecord>
```

Cont
An example personnel-record (2)

```xml
<children>
  <child>
    <name>
      <givenName>Ralph</givenName>
      <initial>T</initial>
      <familyName>Smith</familyName>
    </name>
    <dateOfBirth>19571111</dateOfBirth>
  </child>
  <child>
    <name>
      <givenName>Susan</givenName>
      <initial>B</initial>
      <familyName>Jones</familyName>
    </name>
    <dateOfBirth>19590717</dateOfBirth>
  </child>
</children>
</PersonnelRecord>
```
A count of octets for the personnel-record value

- With white-space omitted, 653 octets
- Fully human-readable with white-space, can be double that! *(Who cares?)*
- BER 136 octets
- PER 94 octets *(An unfriendly example?)*
- ZIP compression .....
- But does size matter anyway? *(Or transaction processing speed?)*
ASN.1 support for XML - Basic and simple

- XML Encoding Rules
- A simple, fixed, encoding in XML for any type defined using ASN.1
- No use of attributes
- No use of Lists
- No use of xsi:type or xsi:nil
- No support for namespaces
- Simple and easy
Add XML Encoding Instructions

- Do not confuse with ECN – a similar but different concept
- Allows use of XML ATTRIBUTES
- Allows use of LIST for SEQUENCE OF
- Either a prefixed encoding instruction (like a TAG), or an encoding control section in the module
- Provides anything a right-minded person might want!
Causes the ASN.1 XML encodings to be the same as XSD encodings for the same type.

For example, the BOOLEAN type encodes as
- `true` or `false`

And not as
- `<true/>` or `<false/>`
Now add more encoding instructions

- Full support for anything you can do with XSD
- Yuck!
- Mapping from XSD to ASN.1 (X.694)
- Reverse not provided – politics!
ASN.1 is an XML Schema notation

- MoU (ISO, IEC, ITU-T, UN/ECE and others) MG recommendation:
  E-business standards should use both XSD and ASN.1 as XML Schema notations
- OASIS UBL uses both XSD and ASN.1
- OASIS XCBF uses only ASN.1 as the Schema notation
Web Services

- Machine-to-machine using Web protocols (SOAP wrappers)
- Flexible publishing of services (parameters etc) with WSDL
- Mapping into Java etc code
- XML encoded transfers
And now the REALLY sexy stuff

- *Fast* Web Services = ASN.1 and PER!
- Being promoted by SUN
- Progressing as ITU-T X.695 | ISO 8825-6
- Linked to binary encoding of XML data
- ASN.1 SOAP, ASN.1 encoding of the XML Infoset
- A new lease of life for ASN.1? Watch this space!
Binary XML – marriages again!

- Schema driven
- Schema-less
- Importance of the XML Infoset
- Simple compression
- Many options
- XSD -> ASN.1 -> PER
PER extensions

- Several aims – not yet mature
- Simplified ECN for common cases
- Better support for Binary XML:
  - Support for message fragments
  - Support for partial messages
  - Variable and partial compression
  - Namespace support
  - Added element support
- A further marriage – XML text goes into PER binary encodings
Syntax for specification

- Relax NG compact syntax – ASN.1-like
- Compact syntax for XSD under discussion
- XML syntax for ASN.1 under discussion
- UML class diagram syntax for ASN.1 – a UML profile – beginning *(how sexy is UML?)*
- So what are the differences? Why does the spec language matter?
  - A variety of encoding rules
  - Mappings to C, C++, Java
  - Efficient processing of messages
  - The language you love or hate!
In conclusion

- ASN.1 made major break-throughs in each of the last few decades
- The abstract syntax concept
- The Information Object concept (not discussed in this presentation)
- Embracing XML encodings
- ECN and then Encoding Instructions
- And now Fast Web Services
- How much more sexy (and useful!) can you get?
Whoops - not that again!

Finito!

ASN.1 consortium