Application of the ASN.1 specification technique to the Bluetooth Service Discover Protocol

Prof John Larmouth
Salford University
Salford M5 4WT
England
+44 161 928 1605
j.larmouth@salford.ac.uk

Olivier Dubuisson
France Telecom R&D
DTL/MSV - 22307 Lannion
France
+33 2 96 05 38 50
Olivier.Dubuisson@francetelecom.com

Paul Thorpe
OSS Nokalva, Inc
1 Executive Drive, Suite 450
Somerset NJ 08873 USA
+1 732 302 9669
thorpe@oss.com

ABSTRACT
This paper describes the work undertaken, and evaluates the results produced, in a project that applied ASN.1 and its newly developed Encoding Control Notation to the Bluetooth Service Discovery Protocol. The use of ASN.1+ECN was shown to be fully capable of specifying the bits-on-the-line required by the approved Bluetooth specification, and identified some areas where the current Bluetooth specification lacks precision - noticeably in areas concerned with "extensibility".

General Terms
Languages.

Keywords
ASN.1, Bluetooth, ECN, Protocol Specification Techniques.

1. INTRODUCTION TO ASN.1
ASN.1 is a notation for the specification of the syntax of messages used in (binary) protocol exchanges.

It is a language-independent notation for the definition of data structures that was first developed in the early 1980s, and widely used as various versions up to the current 1997 version [1]. A 2002 version is in preparation.

Many tools exist to assist in the implementation of protocols defined using ASN.1, with mappings of ASN.1 definitions to C, C++ and Java data structures and automatic generation of encodings.

Defining protocols using ASN.1 can in general reduce the time-to-market for an implementation produced using ASN.1 tools, and can also reduce the incidence of interworking problems due to encoding-related bugs or ambiguities in the specification.

ASN.1 definition of protocols can also make it easier to use notations such as:

- Specification and Description Language (SDL) [2] for definition of the procedural aspects of a protocol; and
- Tree and Tabular Combined Notation (TTCN) [3] for definition of test sequences for a protocol.

Whilst many protocols have been and are being defined using ASN.1, there are nonetheless a significant number of protocols where the standardisers have chosen (for whatever reason) to use a different notation for protocol definition. Bluetooth is one of the latter.

Hitherto, the use of ASN.1 tools by implementers and of SDL and TTCN has been difficult if the base standard was not specified with ASN.1. The development of the Encoding Control Notation (ECN) addition to ASN.1 has changed that, and ASN.1 tools, and ASN.1-based SDL and TTCN specifications can now support any binary-based protocol (including Bluetooth).

2. THE ASN.1 ENCODING CONTROL NOTATION
Many exercises have been undertaken in the past to provide ASN.1 definitions for protocols defined in other ways. However, the application of standard ASN.1 Encoding Rules inevitably results in bits-on-the-line which are not those originally specified, making the ASN.1 specification of little use except as a means of clarifying the real semantic content of the messages.

The ASN.1 Encoding Control Notation (ECN) [4] allows the specification of encodings for data-structures defined using ASN.1 in a sufficiently flexible way that **any** protocol can be re-defined using ASN.1+ECN, with no change to the specified bits-on-the-line.

This work has several benefits.

First, it enables legacy protocols that were defined using old notations such as pictures of bits and bytes or by tables to be re-defined before extension into a new (and usually more complex) version that requires a more sophisticated protocol definition mechanism.

Second, it enables implementers of "modern" protocols (such as Bluetooth) that chose not to use ASN.1 in their definition, to use ASN.1 as part of their implementation strategy, defining ASN.1+ECN (or obtaining such definitions from an ASN.1 tool vendor), and then using ASN.1 tools for the implementation.

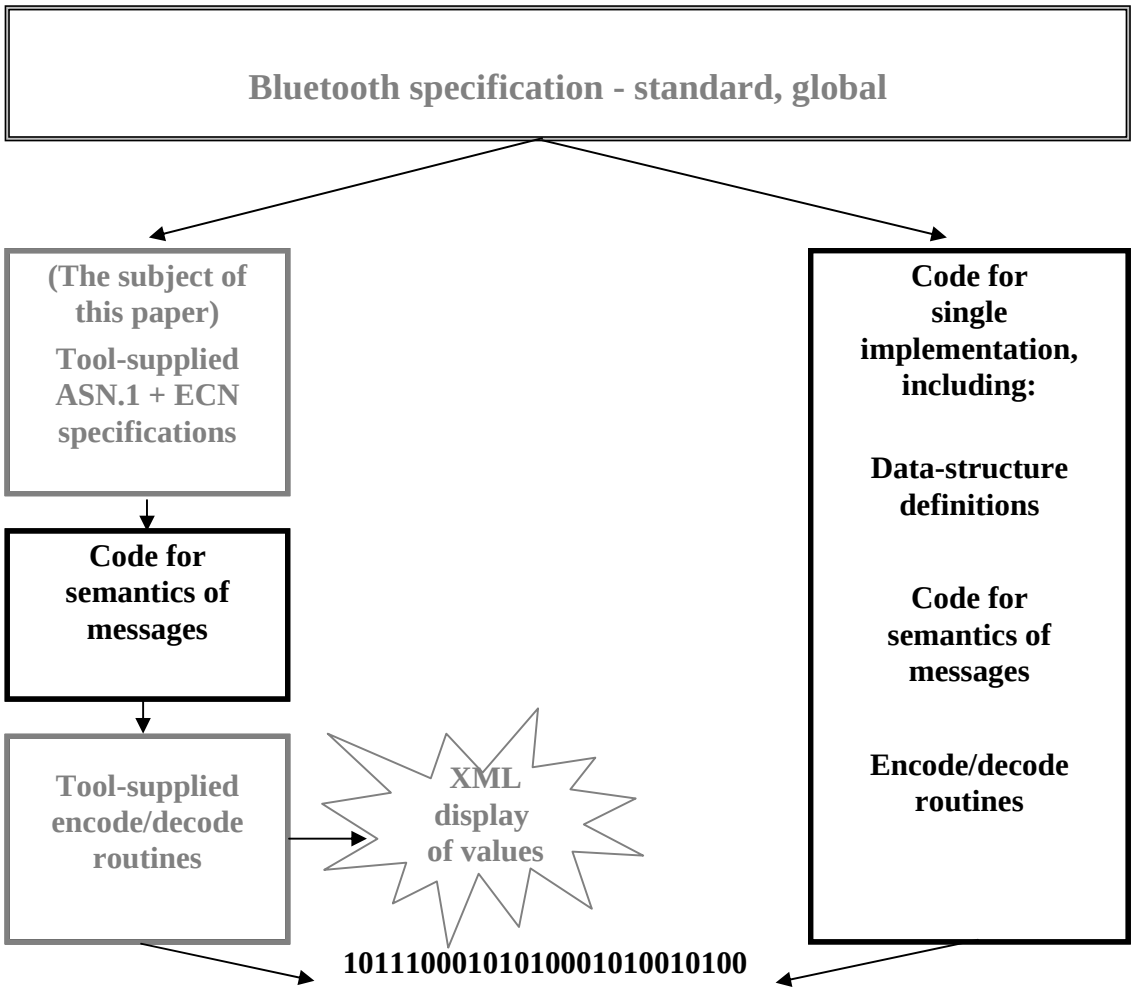

Figure 1: Specifications and code needed

Third, it makes the ASN.1 XML Value Notation and XML Encoding Rules [5] available, at no cost, for the display in a browser of the protocol messages being sent or received by an implementation. This can be a powerful debugging aid.

This paper is concerned primarily with the second of these facilities, and describes the application of ASN.1+ECN to the definition of the Bluetooth Service Discovery Protocol [6].

Figure 1 illustrates the specifications and code needed for (on the right) a conventional implementation of Bluetooth, and (on the left) implementation using an ASN.1+ECN specification and ASN.1 tools. Only the parts that are not grayed out need to be generated for each specific implementation. The figure also shows the ability of ASN.1 tools to display (for debugging purposes) sent and received messages in XML form using the ASN.1 XML Value Notation and Encoding Rules. The main subject of this poster-paper is the specification needed for the box at the top of the left-hand column, a specification that could be provided by the vendor of an ECN tool to Bluetooth implementers.

3. THE BLUETOOTH SERVICE DISCOVERY PROTOCOL

This protocol can reasonably be described as the heart of the Bluetooth protocol suite. It enables any Bluetooth-enabled system to discover the presence in the local environment of other Bluetooth-enabled systems, either in general, of a specific type (offering a specific service), or of a specific type with a given name.The protocol uses a mixture of TLV-encoded types and fixed-length fields.

One of the key features of this protocol is its open-ended nature. It allows suppliers of Bluetooth-enabled equipment to define new types of service, and the parameters associated with that service.

Encoding of such information is always of a Type-Length-Value (TLV) form, so a receiver who does not understand some

particular service type and parameters can easily skip such material and ignore that service.

For parameters defined by equipment suppliers, the types (and hence encoding) of those parameters are defined using a combination of a small number of construction mechanisms and a small number of primitive types. This maps neatly into ASN.1 definition, once ASN.1 types and constructors (that correspond to the Bluetooth "Data Elements") and their encodings have been defined.

Part of the challenge in producing an ASN.1+ECN specification was to provide full support for the addition of such new types by an equipment supplier, using normal ASN.1 notation without the need for additional ECN specification. This challenge was met by use of the ASN.1 constrained Open Type and an object class definition, with rules on what ASN.1 types and constructors can be used by equipment suppliers in the definition of their own types.

4. THE ABSTRACT SYNTAX CONCEPT

Most encodings of protocols have fields that carry the application semantics, together with other fields that provide length delimitation, or identification of alternative selection, or of the presence or absence of optional elements of the encoding. These are described in the ECN work as length, choice, and optionality **determinants**. (There are other determinants, related to the presence of version 2 material within what is otherwise a version 1 encoding, but that goes beyond the scope of this paper.)

There are a variety of mechanisms used in practice for length, choice, and optionality determination. For example, for length determination we can have an explicit length field counting in bits, or one counting in octets with a supplementary count of unused bits, or we can have a special terminator (for example, a null-terminated character string). TLV-style encodings frequently use the "T" part as a means of choice and optionality determination, but explicit encoding of choice indexes or of presence bits also occurs.

ECN supports the specification of all the above mechanisms - and several others - and proved to have sufficient power in this area to cover all the mechanisms used in Bluetooth.

It is often possible to write an ASN.1 type definition for a protocol in which every field, including all determinants, is included in the ASN.1 specification. This is, however, counter to the ASN.1 philosophy, and removes the two main advantages of a normal ASN.1 specification:

- Simplicity through information hiding - encoding features such as determinants are not visible in the ASN.1 specification.
- Automatic handling (insertion on encoding and use in decoding) of determinants is possible by common subroutines without any application code (and hence without application-generated bugs).

The principle of a good ASN.1 specification for a non-ASN.1 protocol is that only those fields that carry application semantics should be included in the ASN.1 specification. Fields that are used solely to support encoding and decoding are called **auxiliary fields** and should not be present in the ASN.1. This means that when a C or C++ or Java API is generated from the ASN.1, the API contains (and the application code is concerned with) only the actual values that carry application semantics. It is sometimes quite challenging to correctly identify which fields are auxiliary fields and which fields have meaning to the application, but usually it is quite obvious.

5. BLUETOOTH DATA ELEMENTS

The Bluetooth is a byte-aligned protocol (almost all fields are a multiple of eight bits), which in many parts is similar to (but of course in detail different from) the ASN.1 Basic Encoding Rules [7] in its encodings.

In particular, a type field and a length field followed by the actual value is used for what Bluetooth calls "Data Elements". The type field carries what in ASN.1 we would describe as the "tag" of the element. This structure is used both for primitive types and for construction mechanisms. Unlike ASN.1, however, Bluetooth does not allow the user to over-ride the tags assigned to these primitive types, so tagging is not visible to the user, nor is it used in the ASN.1 part of the ASN.1+ECN specification for Bluetooth.

Bluetooth declares the following primitive types (similar to, but different from, the primitive types in ASN.1), each with its own tag:

- Nil - the null type.
- Unsigned integer - 5 lengths (1, 2, 4, 8, and 16 bytes) each with a distinct tag.
- Signed two's-complement integer - 5 lengths again.
- Boolean
- A UUID - 3 lengths (2, 4, and 16 bytes)
- A text string - uses a length of length encoding, with the length restricted to one, two, or 4 octets.
- A URL - similar to a text string.

Although not described as Data Elements, Bluetooth also has primitive types for:

- An attribute
- An attribute range
- These types (together with user-defined types) can be used to form more complex user-defined types using the following construction mechanisms:
- Alternative - ASN.1 CHOICE constructions
- Data Element Sequences - ASN.1 SEQUENCE OF constructions
- Data Element Alternatives - also ASN.1 SEQUENCE OF constructions, but with a different tag, and with different application semantics (each element of the sequence represents an alternative that the receiver can select from)
- Simple concatenation in a TLV wrapper – the ASN.1 SEQUENCE construction.

A decision was taken to use the ASN.1 SET OF construction to represent the Data Element Alternatives.

Producing ASN.1 types and constructors for all the Bluetooth primitive types and construction mechanisms, with ECN

specification of the required encodings, was the major part of the work undertaken.

The full specification of the Bluetooth SDP is available from the authors.

6. PROBLEMS ENCOUNTERED

6.1 The building blocks for defining attributes

It was necessary to derive from actual specifications of attributes the primitive types that Bluetooth designers envisaged. Issues such as "can you just say TEXT DATA ELEMENT or do you have to say SHORT TEXT DATA ELEMENT?" had to be resolved by inspection of actual definitions. In fact, it became apparent that the answer in the UNSIGNED INTEGER case was different from that for the TEXT case.

6.2 Canonical representation

There are also instances where 32-bit fields (as the V part of a TLV) are used as pairs of 16 bit fields specifying ranges. These were identified in the ASN.1+ECN as additional basic Data Elements.

6.3 Construction mechanisms involving repetition of an element

It was fortunate that Bluetooth had only two different semantics (and encodings) for a repetition of elements, so that mapping these to SEQUENCE OF and to SET OF was possible. If there had been three semantically different forms of repeated element, then it would have been necessary for equipment suppliers to provide additional ECN text to supplement their definition of new attributes in ASN.1, in order to "color" the different forms of repetition. Fortunately, this was not necessary, and all equipment-supplier-specific text is simply the ASN.1 definition of the new attributes they wish to define. The supplier-independent ECN suffices to encode all types that can be defined with the Bluetooth primitive Data Elements and construction mechanisms.

6.4 The meaning of "reserved"

There was some difficulty in determining what "extensions" were envisaged for Bluetooth version 2, and what extensions equipment suppliers could make in their version 1 specifications.

It was assumed that equipment suppliers providing attribute definitions were not permitted to use Data Elements with the reserved Type Descriptor Values.

It was further assumed that if an id was assigned in version 1 to an attribute, it would not be extended (changed) in version 2, but rather that a new attribute would be defined.

Both these provisions are necessary if good inter-working between version 1 and version 2 systems is to be possible.

However, the current specification is not very precise on what a decoder should do when it receives an attribute id that is "unknown" (an attribute defined by some other equipment manufacturer) - or if this can in fact happen. The TLV structures used for Data Elements and constructors mean that skipping to the end of such material is possible (provided reserved Type Descriptor Values are not present), and this seems the most likely intention.

6.5 Character set matters

The handling of different character sets and languages in Bluetooth does not map easily into ASN.1, and the length of such strings is always an octet count. Such strings were mapped into ASN.1 OCTET STRING types, leaving it to the application to resolve the fairly complex rules on character encoding and language determination in Bluetooth.

7. CONCLUSION

This work demonstrated that, despite some difficulties (mainly arising from occasional lack of precise text in the base specification), it was possible to produce a quite clean and neat ASN.1 and ECN specification for the Bluetooth Service Discovery Protocol.

It was particularly pleasing that the facilities for constrained open types in ASN.1, and for the use of ECN to define the encoding procedures for constructors as well as the encoding of primitive fields, was both necessary and sufficient to handle the Bluetooth provision for equipment-supplier addition of attributes.

Tools for ASN.1+ECN are still under development, but it is confidently expected that the specification that has been produced will (when fed into these tools) generate the same bits-on-the-line as are required by the primary Bluetooth specification.

8. REFERENCES

[1] ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation

[2] ITU-T Recommendation Z.100 (1999), Specification and Description Language (SDL).

[3] ITU-T Recommendation X.292 (1998) | ISO/IEC 9646-3: 1999, Data Communication Networks – OSI conformance testing methodology and framework: Tree and Tabular Combined Notation (TTCN)

[4] ITU-T Recommendation X.692 (2001) | ISO/IEC 8825-3 Information technology – ASN.1 encoding rules: Encoding Control Notation (ECN). See http://www.itu.int/itu-t/asn1/ecn

[5] ITU-T Recommendation X.693 (2001) | ISO/IEC 8825-4 Information technology – ASN.1 encoding rules: XML Encoding Rules (XER). See http://www.itu.int/itu-t/asn1/xml

[6] Specification of the Bluetooth system, Volume 1 v1.0B Dec 1999: p324 – 384 Service Discovery Protocol (SDP) http://www.bluetooth.com/developer/specification

[7] ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, Information technology – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).