       The Protocol versus Document Points of View in Computer Protocols

Status of this Memo

Copyright Notice

Abstract

   This document contrasts two points of view: the "document" point of
   view, where digital objects of interest are like pieces of paper
   written and viewed by people, and the "protocol" point of view where
   objects of interest are composite dynamic network messages. Although
   each point of view has a place, adherence to a document point of view
   can be damaging to protocol design. By understanding both points of
   view, conflicts between them may be clarified and reduced.

Table of Contents

1. Introduction

   This document contrasts: the "document" point of view, where digital
   objects of interest are thought of as pieces of paper written and
   viewed by people, and the "protocol" point of view, where objects of
   interest are composite dynamic network messages. Those accustomed to
   one point of view frequently have great difficulty appreciating the
   other: Even after they understand it, they almost always start by
   considering things from their accustomed point of view, assume that
   most of the universe of interest is best viewed from their
   perspective, and commonly slip back into thinking about things
   entirely from that point of view. Although each point of view has a
   place, adherence to a document point of view can be damaging to
   protocol design. By understanding both points of view, conflicts
   between them may be clarified and reduced.

   Much of the IETF's traditional work has concerned low level binary
   protocol constructs. These are almost always viewed from the
   protocol point of view. But as higher level application constructs
   and syntaxes are involved in the IETF and other standards processes,
   difficulties can arise due to participants who have the document
   point of view. These two different points of view defined and
   explored in section 2 below.

   Section 3 gives some examples. Section 4 tries to synthesize the
   views and give general design advice in areas that can reasonably be
   viewed either way.

2. Points of View

   The following subsections contrast the document and protocol points
   of view. Each viewpoint is EXAGGERATED for effect.

   The document point of view is indicated in paragraphs headed "DOCUM",
   and the protocol point of view is indicated in paragraphs headed
   "PROTO".

2.1. The Basic Points of View

    DOCUM: What is important are complete (digital) documents, analogous
       to pieces of paper, viewed by people. A major concern is to be
       able to present such documents as directly as possible to a court
       or other third party. Because what is presented to the person is
       all that is important, anything that can effect this, such as a
       "style sheet" [CSS], MUST be considered part of the document.
       Sometimes it is forgotten that the "document" originates in a
       computer, may travel over, be processed in, and be stored in
       computer systems, and is viewed on a computer, and that such
       operations may involve transcoding, enveloping, or data
       reconstruction.

    PROTO: What is important are bits on the wire generated and consumed
       by well-defined computer protocol processes. No person ever sees
       the full messages as such; it is only viewed as a whole by geeks
       when debugging, and even then they only see some translated
       visible form. If one actually ever has to demonstrate something
       about such a message in a court or to a third party, there isn't
       any way to avoid having computer experts interpret it. Sometimes
       it is forgotten that pieces of such messages may end up being
       included in or influencing data displayed to a person.

2.2. Questions of Meaning

    The document and protocol points of view have radically different
    concepts of the "meaning" of data. The document oriented tend to
    consider "meaning" to a human reader extremely important, but this is
    something the protocol oriented rarely think about at all.

    This difference in point of view extends beyond the core meaning to
    the meaning of addenda to data. Both core and addenda meaning are
    discussed below.

2.2.1. Core Meaning

    DOCUM: The "meaning" of a document is a deep and interesting human
       question related to volition. It is probably necessary for the
       document to include or reference human language policy and/or
       warranty/disclaimer information. At an absolute minimum, some
       sort of semantic labelling is required. The assumed situation is
       always a person interpreting the whole "document" without other
       context. Thus it is reasonable to consult attorneys during
       message design, to require that human-readable statements be
       "within the four corners" of the document, etc.

   PROTO: The "meaning" of a protocol message should be clear and
      unambiguous from the protocol specification. It is frequently
      defined in terms of the state machines of the sender and recipient
      processes and may have only the most remote connection with human
      volition. Such processes have additional context, and the message
      is usually only meaningful with that additional context. Adding
      any human-readable text that is not functionally required is
      silly. Consulting attorneys during design is a bad idea that
      complicates the protocol and could tie a design effort in knots.

2.2.2. Adjunct Meaning

   Adjunct items can be added or are logical addenda to a message.

   DOCUM: From a document point of view, at the top level is a person
      looking at a document. So adjunct items such as digital
      signatures, person's names, dates, etc., must be carefully labeled
      as to meaning. Thus a digital signature needs to include, in more
      or less human-readable form, what that signature means (is the
      signer a witness, author, guarantor, authorizer, or what?).
      Similarly, a person's name needs to be accompanied by that
      person's role, such as editor, author, subject, or contributor.
      As another example, a date needs to be accompanied by the
      significance of the date, such as date of creation, modification,
      distribution, or some other event.
         Given the unrestrained scope of what can be documented, there
      is a risk of trying to enumerate and standardize all possible
      "semantic tags" for each kind of adjunct data during in the design
      process. This can be a difficult, complex, and essentially
      infinite task (i.e., a rat hole).

   PROTO: From a protocol point of view, the semantics of the message
      and every adjunct in it are defined in the protocol specification.
      Thus, if there is a slot for a digital signature, person's name, a
      date, or whatever, the party who is to enter that data, the party
      or parties who are to read it, and its meaning are all pre-
      defined. Even if there are several possible meanings, the
      specific meaning that applies can be specified by a separate
      enumerated type field. There is no reason for such a field to be
      directly human readable. Only the "meanings" directly relevant to
      the particular protocol need be considered. Another way to look
      at this is that the "meaning" of each adjunct, instead of being
      pushed into and coupled with the adjunct itself, as the document
      point of view encourages, is commonly promoted to the level of the
      protocol specification, resulting in simpler adjuncts.

2.3. Processing Models

   The document oriented and protocol oriented have very different views
   on what is likely to happen to an object.

2.3.1. Amount of Processing

   DOCUM: The model is of a quasi-static object like a piece of paper.
      About all one does to pieces of paper is transfer them as a whole,
      from one storage area to another, or add signatures, date stamps,
      or similar adjuncts. (Possibly one might want an extract from a
      document or to combine multiple documents into a summary, but this
      isn't the common case.)

   PROTO: The standard model of a protocol message is as an ephemeral
      composite, multi-level object created by a source process and
      consumed by a destination process. Such a message is constructed
      from information contained in previously received messages,
      locally stored information, local calculations, etc. Quite
      complex processing is normal.

2.3.2. Granularity of Processing

   DOCUM: The document view is generally of uniform processing or
      evaluation of the object being specified. There may be an
      allowance for attachments or addenda, but, if so, they would
      probably be simple, one level, self documenting attachments or
      addenda. (Separate processing of an attachment or addenda is
      possible but not usual.)

   PROTO: Processing is complex and almost always affects different
      pieces of the message differently. Some pieces may be intended
      for use only by the destination process and may be extensively
      processed there. Others may be present so that the destination
      process can, at some point, do minimal processing and forward them
      in other messages to yet more processes. The object's structure
      can be quite rich and have multilevel or recursive aspects.
      Because messages are processed in a local context, elements of the
      message may include items like a signature that covers multiple
      data elements, some of which are in the message, some received in
      previous messages, and some locally calculated.

2.3.3. Extensibility of Processing

   DOCUM: The document oriented don't usually think of extensibility as
      a major problem. They assume that their design, perhaps with some
      simple version scheme, will meet all requirements. Or, coming
      from an SGML/DTD world of closed systems, they may assume that
      knowledge of new versions or extensions can be easily and
      synchronously distributed to all participating sites.

   PROTO: Those who are protocol oriented assume that protocols will
      always need to be extended and that it will not be possible to
      update all implementations as such extensions are deployed and/or
      retired. This is a difficult problem but those from the protocol
      point of view try to provide the tools needed. For example, they
      specify carefully defined versioning and extension/feature
      labelling, including the ability to negotiate versions and
      features where possible and at least a specification of how
      parties running different levels should interact, providing
      length/delimiting information for all data so that it can be
      skipped if not understood, and providing destination labelling so
      that a process can tell that it should ignore data except for
      passing it through to a later player.

2.4. Security and Canonicalization

   Security is a subtle area. Sometime problems can be solved in a way
   that is effective across many applications. Those solutions are
   typically incorporated into standard security syntaxes such as those
   for ASN.1 [RFC3852] and XML [RFC3275, XMLENC]. But there are almost
   always application specific questions, particularly the question of
   exactly what information needs to be authenticated or encrypted.

   Questions of exactly what needs to be secured and how to do so
   robustly are deeply entwined with canonicalization. They are also
   somewhat different for authentication and encryption, as discussed
   below.

2.4.1. Canonicalization

   Canonicalization is the transformation of the "significant"
   information in a message into a "standard" form, discarding
   "insignificant" information, for example, encoding into a standard
   character set or changing line endings into a standard encoding and
   discarding the information about the original character set or line
   ending encodings. Obviously, what is "significant" and what is
   "insignificant" varies with the application or protocol and can be
   tricky to determine. However, it is common that for each particular
   syntax, such as ASCII [ASCII], ASN.1 [ASN.1], or XML [XML], a

standard canonicalization (or canonicalizations) is specified or
developed through practice. This leads to the design of applications
that assume one of such standard canonicalizations, thus reducing the
need for per-application canonicalization. (See also [RFC3076,
RFC3741].)

DOCUM: From the document point of view, canonicalization is suspect
   if not outright evil. After all, if you have a piece of paper
   with writing on it, any modification to "standardize" its format
   can be an unauthorized change in the original message as created
   by the "author", who is always visualized as a person. Digital
   signatures are like authenticating signatures or seals or time
   stamps on the bottom of the "piece of paper". They do not justify
   and should not depend on changes in the message appearing above
   them. Similarly, encryption is just putting the "piece of paper"
   in a vault that only certain people can open and does not justify
   any standardization or canonicalization of the message.

PROTO: From the protocol point of view, canonicalization is simply a
   necessity. It is just a question of exactly what canonicalization
   or canonicalizations to apply to a pattern of bits that are
   calculated, processed, stored, communicated, and finally parsed
   and acted on. Most of these bits have never been seen and never
   will be seen by a person. In fact, many of the parts of the
   message will be artifacts of encoding, protocol structure, and
   computer representation rather than anything intended for a person
   to see.
      Perhaps in theory, the "original", idiosyncratic form of any
   digitally signed part could be conveyed unchanged through the
   computer process, storage, and communications channels that
   implement the protocol and could be usefully signed in that form.
   But in practical systems of any complexity, this is unreasonably
   difficult, at least for most parts of messages. And if it were
   possible, it would be virtually useless, because to authenticate
   messages you would still have to determine their equivalence with
   the preserved original form.
      Thus, signed data must be canonicalized as part of signing and
   verification to compensate for insignificant changes made in
   processing, storage, and communication. Even if, miraculously, an
   initial system design avoids all cases of signed message
   reconstruction based on processed data or re-encoding based on
   character set or line ending or capitalization or numeric
   representation or time zones or whatever, later protocol revisions
   and extensions are certain to require such reconstruction and/or
   re-encoding eventually. If such "insignificant" changes are not
   ameliorated by canonicalization, signatures won't work, as
   discussed in more detail in 2.4.3 below.

2.4.2. Digital Authentication

   DOCUM: The document-oriented view on authentication tends to be a
      "digital signature" and "forms" point of view. (The "forms" point
      of view is a subset of the document point of view that believes
      that a principal activity is presenting forms to human beings so
      that they can fill out and sign portions of those forms [XForms]).
      Since the worry is always about human third parties and viewing
      the document in isolation, those who are document oriented always
      want "digital signature" (asymmetric key) authentication, with its
      characteristics of "non-repudiability", etc. As a result, they
      reject secret key based message authentication codes, which
      provide the verifier with the capability of forging an
      authentication code, as useless. (See any standard reference on
      the subject for the usual meaning of these terms.)
         From their point of view, you have a piece of paper or form
      which a person signs. Sometimes a signature covers only part of a
      form, but that's usually because a signature can only cover data
      that is already there. And normally at least one signature covers
      the "whole" document/form. Thus the document oriented want to be
      able to insert digital signatures into documents without changing
      the document type and even "inside" the data being signed, which
      requires a mechanism to skip the signature so that it does not try
      to sign itself.

   PROTO: From a protocol point of view, the right kind of
      authentication to use, whether "digital signature" or symmetric
      keyed authentication code (or biometric or whatever), is just
      another engineering decision affected by questions of efficiency,
      desired security model, etc. Furthermore, the concept of signing
      a "whole" message seems very peculiar (unless it is a copy being
      saved for archival purposes, in which case you might be signing a
      whole archive at once anyway). Typical messages are made up of
      various pieces with various destinations, sources, and security
      requirements. Furthermore, there are common fields that it is
      rarely useful to sign because they change as the message is
      communicated and processed. Examples include hop counts, routing
      history, and local forwarding tags.

2.4.3. Canonicalization and Digital Authentication

   For authenticating protocol system messages of practical complexity,
   you are faced with the choice of doing

   (1) "too little canonicalization" and having brittle authentication,
       useless due to verification failures caused by surface
       representation changes without significance,

   (2) the sometimes difficult and tricky work of selecting or designing
       an appropriate canonicalization or canonicalizations to be used
       as part of authentication generation and verification, producing
       robust and useful authentication, or

   (3) "too much canonicalization" and having insecure authentication,
       useless because it still verifies even when significant changes
       are made in the signed data.

   The only useful option above is number 2.

2.4.4. Encryption

   In terms of processing, transmission, and storage, encryption turns
   out to be much easier to get working than signatures. Why? Because
   the output of encryption is essentially random bits. It is clear
   from the beginning that those bits need to be transferred to the
   destination in some absolutely clean way that does not change even
   one bit. Because the encrypted bits are meaningless to a human
   being, there is no temptation among the document oriented to try to
   make them more "readable". So appropriate techniques of encoding at
   the source, such as Base64 [RFC2045], and decoding at the
   destination, are always incorporated to protect or "armor" the
   encrypted data.

   Although the application of canonicalization is more obvious with
   digital signatures, it may also apply to encryption, particularly
   encryption of parts of a message. Sometimes elements of the
   environment where the plain text data is found may affect its
   interpretation. For example, interpretation can be affected by the
   character encoding or bindings of dummy symbols. When the data is
   decrypted, it may be into an environment with a different character
   encoding or dummy symbol bindings. With a plain text message part,
   it is usually clear which of these environmental elements need to be
   incorporated in or conveyed with the message. But an encrypted
   message part is opaque. Thus some canonical representation that
   incorporates such environmental factors may be needed.

   DOCUM: Encryption of the entire document is usually what is
      considered. Because signatures are always thought of as human
      assent, people with a document point of view tend to vehemently
      assert that encrypted data should never be signed unless the plain
      text of it is known.

   PROTO: Messages are complex composite multi-level structures, some
      pieces of which are forwarded multiple hops. Thus the design
      question is what fields should be encrypted by what techniques to
      what destination or destinations and with what canonicalization.

It sometimes makes perfect sense to sign encrypted data you don't
understand; for example, the signature could just be for integrity
protection or for use as a time stamp, as specified in the
protocol.

2.5. Unique Internal Labels

It is desirable to be able to reference parts of structured messages
or objects by some sort of "label" or "id" or "tag". The idea is
that this forms a fixed "anchor" that can be used "globally", at
least within an application domain, to reference the tagged part.

DOCUM: From the document point of view, it seems logical just to
   provide for a text tag. Users or applications could easily come
   up with short readable tags. These would probably be meaningful
   to a person if humanly generated (e.g., "Susan") and at least
   fairly short and systematic if automatically generated (e.g.,
   "A123"). The ID attribute type in XML [XML] appears to have been
   thought of this way, although it can be used in other ways.

PROTO: From a protocol point of view, unique internal labels look
   very different than they do from a document point of view. Since
   this point of view assumes that pieces of different protocol
   messages will later be combined in a variety of ways, previously
   unique labels can conflict. There are really only three
   possibilities if such tags are needed, as follows:

   (1) Have a system for dynamically rewriting such tags to maintain
       uniqueness. This is usually a disaster, as it (a) invalidates
       any stored copies of the tags that are not rewritten, and it
       is usually impossible to be sure there aren't more copies
       lurking somewhere you failed to update, and (b) invalidates
       digital signatures that cover a changed tag.
   (2) Use some form of hierarchical qualified tags. Thus the total
       tag can remain unique even if a part is moved, because its
       qualification changes. This avoids the digital signature
       problems described above. But it destroys the concept of a
       globally-unique anchor embedded in and moving with the data.
       And stored tags may still be invalidated by data moves.
       Nevertheless, within the scope of a particular carefully
       designed protocol, such as IOTP [RFC2801], this can work.
   (3) Construct a lengthy globally-unique tag string. This can be
       done successfully by using a good enough random number
       generator and big enough random tags (perhaps about 24
       characters) sequentially, as in the way email messages IDs are
       created [RFC2822].

        Thus, from a protocol point of view, such tags are difficult but
        if they are needed, choice 3 works best.

3. Examples

   IETF protocols are replete with examples of the protocol viewpoint
   such as TCP [RFC793], IPSEC [RFC2411], SMTP [RFC2821], and IOTP
   [RFC2801, RFC2802].

   The eXtensible Markup Language [XML] is an example of something that
   can easily be viewed both ways and where the best results frequently
   require attention to both the document and the protocol points of
   view.

   Computerized court documents, human-to-human email, and the X.509v3
   Certificate [X509v3], particularly the X509v3 policy portion, are
   examples primarily designed from the document point of view.

4. Resolution of the Points of View

   There is some merit to each point of view. Certainly the document
   point of view has some intuitive simplicity and appeal and is OK for
   applications where it meets needs.

   The protocol point of view can come close to encompassing the
   document point of view as a limiting case. In particular, it does so
   under the following circumstances:

   1. As the complexity of messages declines to a single payload
      (perhaps with a few attachments).

   2. As the mutability of the payload declines to some standard format
      that needs little or no canonicalization.

   3. As the number of parties and amount of processing declines as
      messages are transferred.

   4. As the portion of the message intended for more or less direct
      human consumption increases.

   Under the above circumstances, the protocol point of view would be
   narrowed to something quite close to the document point of view.
   Even when the document point of view is questionable, the addition of
   a few options to a protocol will usually mollify the perceived needs
   of those looking at things from that point of view. For example,
   adding optional non-canonicalization or an optional policy statement,
   or inclusion of semantic labels, or the like.

On the other hand, the document point of view is hard to stretch to encompass the protocol case. From a strict piece of paper perspective, canonicalization is wrong; inclusion of human language policy text within every significant object and a semantic tag with every adjunct should be mandatory; and so on. Objects designed in this way are rarely suitable for protocol use, as they tend to be improperly structured to accommodate hierarchy and complexity, inefficient (due to unnecessary text and self-documenting inclusions), and insecure (due to brittle signatures).

Thus, to produce usable protocols, it is best to start with the protocol point of view and add document point of view items as necessary to achieve consensus.

5. Conclusion

I hope that this document will help explain to those of either point of view where those with the other view are coming from. It is my hope that this will decrease conflict, shed some light -- in particular on the difficulties of security design -- and lead to better protocol designs.

6. Security Considerations

This document considers the security implications of the Document and Protocol points of view, as defined in Sections 2.1 and 2.2, and warns of the security defects in the Document view. Most of these security considerations appear in Section 2.4 but they are also touched on elsewhere in Section 2 which should be read in its entirety.

Informative References

    [ASCII]   "USA Standard Code for Information Interchange", X3.4,
                   American National Standards Institute: New York, 1968.

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

                   ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998,
                   "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/index.html>.

   [CSS]   "Cascading Style Sheets, level 2 revision 1 CSS 2.1
                    Specification", B. Bos, T. Gelik, I. Hickson, H. Lie,
                    W3C Candidate Recommendation, 25 February 2004.
                    <http://www.w3.org/TR/CSS21>

   [RFC793]   Postel, J., "Transmission Control Protocol", STD 7, RFC
                    793, September 1981.

   [RFC2045]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
                    Extensions (MIME) Part One: Format of Internet Message
                    Bodies", RFC 2045, November 1996.

   [RFC2411]   Thayer, R., Doraswamy, N., and R. Glenn, "IP Security
                    Document Roadmap", RFC 2411, November 1998.

   [RFC3852]   Housley, R., "Cryptographic Message Syntax (CMS)", RFC
                    3852, July 2004.

   [RFC2801]   Burdett, D., "Internet Open Trading Protocol - IOTP
                    Version 1.0", RFC 2801, April 2000.

   [RFC2802]   Davidson, K. and Y. Kawatsura, "Digital Signatures for
                    the v1.0 Internet Open Trading Protocol (IOTP)", RFC
                    2802, April 2000.

   [RFC2821]   Klensin, J., "Simple Mail Transfer Protocol", RFC 2821,
                    April 2001.

   [RFC2822]   Resnick, P., "Internet Message Format", RFC 2822, April
                    2001.

   [RFC3076]   Boyer, J., "Canonical XML Version 1.0", RFC 3076, March
                    2001.

   [RFC3275]   Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible
                    Markup Language) XML-Signature Syntax and Processing",
                    RFC 3275, March 2002.

   [RFC3741]   Berger, L., "Generalized Multi-Protocol Label Switching
                    (GMPLS) Signaling Functional Description", RFC 3471,
                    January 2003.

   [X509v3]   "ITU-T Recommendation X.509 version 3 (1997),
                    Information Technology - Open Systems Interconnection -
                    The Directory Authentication Framework", ISO/IEC 9594-
                    8:1997.

   [XForms]   "XForms 1.0", M. Dubinko, L. Klotz, R. Merrick, T.
                   Raman, W3C Recommendation 14 October 2003.
                   <http://www.w3.org/TR/xforms/>

   [XML]   "Extensible Markup Language (XML) 1.0 Recommendation
                   (2nd Edition)". T. Bray, J. Paoli, C. M. Sperberg-
                   McQueen, E. Maler, October 2000.
                   <http://www.w3.org/TR/2000/REC-xml-20001006>

   [XMLENC]   "XML Encryption Syntax and Processing", J. Reagle, D.
                   Eastlake, December 2002.
                   <http://www.w3.org/TR/2001/RED-xmlenc-core-20021210/>

Author's Address

   Donald E. Eastlake 3rd
   Motorola Laboratories
   155 Beaver Street
   Milford, MA 01757 USA

   Phone: +1 508-786-7554 (w)
           +1 508-634-2066 (h)
   Fax: +1 508-786-7501 (w)
   EMail: Donald.Eastlake@motorola.com