Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-intserv-data-encoding-01.txt

Integrated Services WG
J. Wroclawski
MIT LCS

November, 1995
Expires: 6/96

Standard Data Encoding for Integrated Services Objects

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

This draft is a product of the Integrated Services Working Group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at int-serv@isi.edu and/or the author(s).

Abstract

A number of data objects must be transmitted between end-nodes and network elements to support resource reservation in an integrated services internet. This document provides some background requirements and specifies preferred transmission encodings for these objects, which include RSpecs, TSpecs, characterization paramters, and composition function intermediate parameters.

Introduction

Support for end-to-end resource reservation and controlled qualities of service requires that several types of information be communicated between end-nodes and intermediate network elements. This information may include characterizations of the traffic to be sent over that path, requests for a specific level of service, characterizations of the chosen data path, and other related data.

The specifics of this communication depend heavily on the mechanisms being used to set up and reserve resources for QoS-controlled paths. The same integrated services building blocks may be used to support a range of different setup protocols, reservation styles, and management functions. Data and message formats designed for use with these building blocks must be able to support this varied use.

This note provides some background information about the requirements of the integrated services data transport formats. It then describes a general set of encoding rules which, together with the information contained in a particular service specification document, specify the preferred wire encoding format for the data associated with that service. Finally, it gives some usage examples for these rules.

Background and Requirements

Current IETF integrated services documents discuss four types of data objects:

- Traffic Specifications, or TSpecs, describe a flow of data traffic. This may be the traffic generated by a data source, the traffic expected by a data receiver, or the traffic subject to a resource reservation at an intermediate network element.

- Request Specifications, or RSpecs, describe a request for a specific QoS control service. This may be request being placed by an application or end-node, or it may be a request delivered to a particular intermediate network element.

- Characterization Parameters are specific parameters made available by the service module at an end-node or intermediate network element for the purpose of characterizing the end-to-end QoS delivered over a data path. The parameters made available by a service module are a function of the service that module implements.

- Composition variables are the intermediate variables and end results of composition functions; the functions which process per-hop characterization parameters to compute end-to-end path characterizations.

> NOTE: Traffic Specifications, Request Specifications and Characterization Parameters are defined further in [the template document]. The phrase "Composition Variables", or equivalent, is not currently defined in current version of that document, but will be included inthe next version.
>
> NOTE: Additional types of data objects may be defined in the future.

These objects may be used in a variety of different ways, depending on the situation. Some examples are listed below:

- A routing protocol might query network elements for certain characterization parameters, such as link bandwidth or delay. This protocol would make no use of RSpecs or TSpecs, and may or may not use composition variables.

- A management station setting up a quasi-permanant flow might transmit RSpecs (specifying the requested service) and TSpecs (specifying the level of traffic for which the service is requested) directly from a central point to each of a number of network elements.

- A one-pass receiver based resource reservation protocol such as RSVP might do the following:

  o Transmit TSpecs describing each sender's traffic from the sender to intermediate network elements.

  o Compute end-to-end characterizations in the sender-to-receiver direction, by collecting characterization parameters at the sending end-node and intermediate elements, executing composition functions at the intermediate nodes, and transmitting composition variables from senders towards receivers.

  o Transmit an RSpec (describing each receiver's desired reservation request) and a TSpec (describing the traffic parameters for which each receiver desires this level of

service) from each receiver to intermediate network elements.

o Compute an appropriate RSpec and TSpec for each network element, based on the protocol's rules, and attempt to reserve resources based on that RSpec and TSpec.

- A two-pass sender-based setup protocol such as ST-II might do the following:

o On pass one, transmit each sender's RSpec (describing the sender's service request) and TSpec (describing the sender's traffic generation pattern) into the network.

o On pass one, compute end-to-end characterizations in the sender-to-receiver direction, by collecting characterization parameters at the sending end-node and intermediate elements, executing composition functions at the intermediate nodes, and transmitting composition variables from senders towards receivers.

o Between passes, compute a final RSpec and TSpec for the resource reservation request, based on information collected in pass one.

o On pass two, carry the final RSpec and TSpec from the receiver towards the sender, placing reservations using these parameters.

These differing examples show that a single protocol message may contain one, some, or all of these data objects. For this reason, the format described below is self-parsing at the object level.

The examples also show that a single protocol message may be required to carry objects relating to several services at the same time. For example, a one-pass reservation protocol such as RSVP may transport composition variables for several services from sender to receiver to more fully characterize the path. For this reason, the format described below has a two-level structure which can carry objects from multiple services simultaneously.

Message Format

This section describes the preferred wire format for protocol messages used to transmit integrated services data objects. It

provides rules constructing the overall data format for carrying integrated messages. The data contained *within a specific object* is unique to the service defining the object, and is described in the service definition. To fully interpret a message, the recipient must understand both the overall format presented here and the data objects defined by the specific service. However, the format is self-parsing at the service level, so that objects belonging to an unknown service may be ignored entirely.

All fields described below are drawn and transmitted in big-endian "network byte order".

The overall representation consists of a 32-bit header specifying the version number and total length of the message, followed by any number of service-specific data blocks. The overall message must be aligned to a 32-bit boundary within the protocol's data packet. The message length is measured in 32-bit words *not including the word containing the header*.

Each service-specific data block begins with a 16-bit header containing the service number and length field, followed by the service-specific parameters. The length field specifies the number of 32-bit words used to hold data specific to this service as a count of 32-bit words *not including the word containing the header*. Note that the alignment rules described below ensure that the 16-bit service header will always be aligned to a 32-bit boundary.

Each parameter within a service-specific data block is identified by a 16-bit parameter-id header containing the parameter identifier (parameter number) and a flag field. The data field(s) of the parameter follow. The parameter's ID number, as well as the meaning, data format, and location of each data field within the parameter, are given by the specification which defines the parameter. The placement and alignment of a parameter's data fields within the message is specified by the alignment rules below.

Within a service-specific data block, parameter headers and parameter data fields are aligned to their natural boundaries (16 bit fields aligned to a 16 bit boundary, etc.) by pre-padding with zero bytes if required. Note that this implies that parameter headers are aligned only to the nearest 16-bit boundary. Within a service-specific data block, all parameter headers after the first may appear in either the high-order or low-order 16 bits of a 32-bit word, depending on where the data field(s) of the previous parameter end.

If the last field of the last parameter in a service-specific data block does not end on a 32 bit boundary the data block is padded to the next 32-bit boundary with zero bytes.

Message Header Field

```
    31    28 27                   16 15                             0
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | V     |    Unused            |    OVERALL LENGTH           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
V                - Flowspec version; currently 0
OVERALL LENGTH   - Overall length in 32-bit words not including header
```

Service-Specific Data Header Field

```
    15             8 7             0
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  SVC_NUMBER   |  LENGTH       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
SVC_NUMBER       - Service ID number (defined in service specification)
LENGTH           - Service-specific data length in 32-bit words,
                   not including header.
```

Parameter-ID Header

```
    15             8 7             0
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  PARAM_NUM    |   PARAM_FLAGS |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                     I
                     n
                     v
```

```
PARAM_NUM        - Parameter ID number (defined in service specification)
PARAM_FLAGS      - Flags. Currently:
                   INV (Invalid) (bit 7) - This parameter may be invalid.
```

Examples

Shown below is an example message carrying both a TSpec (parameter 1) and RSpec (parameter 2) for service number 2. This message might be contained within a FLOWSPEC object in the RSVP protocol.

```
        31           24 23            16 15              8 7              0
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | 0 (a) |    Unused             |            5 (b)              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     2 (c)     |    4 (d)      |    1 (e)      |     0 (f)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | 16-bit TSpec field            |      zero padding             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                    32-bit TSpec field                         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | 16-bit TSPec field            | 16-bit TSpec field            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     2 (g)     |    0 (h)      |   16-bit RSpec field          |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

(a) - Version number (0)
(b) - Overall length (6 words not including header)
(c) - Service header, service number 2
(d) - Length of per-service data, 5 words not including header
(e) - Parameter ID, parameter 1 (TSpec)
(f) - Parameter 1 flags (none set)
(g) - Parameter ID, parameter 2
(h) - Parameter 2 flags (none set)

Shown below is an example message carrying characterization information for two different services simultaneously. This message might be contained within an ADSPEC object in the RSVP protocol.

```
        31           24 23            16 15              8 7              0
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | 0 (a) |    Unused             |            7 (b)              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     10 (c)    |    3 (d)      |    17 (e)     |     0 (f)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |        32-bit characterization parameter value                |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     18 (g)    |1(h)           |       zero padding            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |        32-bit characterization parameter value                |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     11 (i)    |    2 (j)      |    17 (k)     |     0 (l)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   16-bit parameter 17 value   |    18 (m)     |     0 (n)     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   16-bit parameter 18 value   |         zero padding          |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

(a) - Version number (0)
(b) - Overall length, 7 words not including header
(c) - Service header, indicates service number 10
(d) - Length of per-service data, 3 words not including header
(e) - Parameter ID, indicates characterization parameter 17
(f) - Parameter 17 flags (none set)
(g) - Parameter ID, indicates characterization parameter 18
(h) - Parameter 18 flags. INVALID flag set.
(i) - Service header, indicates service number 11
(j) - Length of per-service data, 2 words not including header
(k) - Parameter ID, indicates characterization parameter 17
(l) - Parameter 17 flags (none set)
(m) - Parameter ID, indicates characterization parameter 18
(n) - Parameter 18 flags (none set)

Security Considerations

Security considerations are not discussed in this memo.

Authors' Address:

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139
jtw@lcs.mit.edu
617-253-7885
617-253-2673 (FAX)