The Blocks eXtensible eXchange Protocol Framework
draft-mrose-blocks-protocol-03.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted. (If this document becomes part of an IETF working group activity, then it will be brought into full compliance with Section 10 of RFC2026.)

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 15, 2000.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This memo describes a generic application protocol framework for connection-oriented, asynchronous request/response interactions. The framework permits multiplexing of independent request/response streams over a single transport connection, supporting both textual and binary messages.

To subscribe to the Blocks discussion list, send e-mail[11]; there is also a developers’ site[12].
Table of Contents

1. Introduction ................................................. 4
2. The Blocks eXtensible eXchange Protocol .................... 5
  2.1 Roles ..................................................... 5
  2.2 Messages and Frames ....................................... 6
    2.2.1 Message Syntax ...................................... 7
    2.2.1.1 Frame Header .................................. 8
    2.2.1.2 Frame Payload .................................. 10
    2.2.1.3 Frame Trailer .................................. 10
    2.2.2 Frame Semantics ..................................... 11
  2.3 Channel Management ........................................ 12
    2.3.1 Message Semantics ................................... 12
      2.3.1.1 The Start Message ............................. 12
      2.3.1.2 The Greeting Message ........................ 15
      2.3.1.3 The Error Message ............................. 16
  2.4 Session Establishment and Release ......................... 17
  2.5 Flow Control ............................................. 19
    2.5.1 Channel Creation .................................... 19
    2.5.2 Sending REQ or RSP Messages ........................ 20
    2.5.3 Receiving REQ or RSP Messages ........................ 20
    2.5.4 Processing SEQ Messages ............................ 21
    2.5.5 Use of Flow Control ................................ 22
  2.6 Parallelism ................................................. 23
    2.6.1 Within a single channel ............................. 23
    2.6.2 Between different channels ........................... 23
    2.6.3 Pre-emptive responses ................................ 23
    2.6.4 Interference ......................................... 23
  2.7 Peer-to-Peer Behavior ...................................... 24
  3. Transport Security .......................................... 25
    3.1 The TLS Transport Security Profile .................... 28
      3.1.1 Profile Identification and Initialization ........ 28
      3.1.2 Request and Response Messages .................... 29
      3.1.3 Message Semantics ................................ 30
        3.1.3.1 The Ready Message ............................ 30
        3.1.3.2 The Proceed Message .......................... 30
  4. User Authentication .......................................... 31
    4.1 The SASL Family of Profiles ............................ 32
      4.1.1 Profile Identification and Initialization .......... 33
      4.1.2 Request and Response Messages ........................ 35
      4.1.3 Message Semantics ................................ 36
  5. Profile Registration Template ............................... 37
    5.1 Initial Profile Registrations ........................... 38
      5.1.1 BXXP Channel Management .......................... 38
      5.1.2 BXXP Channel Management DTD ........................ 39
      5.1.3 TLS Transport Security Profile Registration ....... 41
      5.1.4 TLS Transport Security Profile DTD ................ 42
      5.1.5 SASL Family of Profiles Registration ................ 43
      5.1.6 SASL Family of Profiles DTD ........................ 44
1. Introduction

This memo describes a generic application protocol framework for connection-oriented, asynchronous request/response interactions over TCP[1]. Consult [2] for a description of the design principles for the framework.

At the core of the BXXP framework is a framing mechanism that allows for peer-to-peer exchanges of requests and responses. The framing mechanism permits multiplexing multiple, simultaneous, and independent exchanges over a single transport connection with flow control and segmentation. Requests and responses are either textual (structured using XML[3]) or arbitrary (structured using MIME[4]).

Frames are exchanged in the context of a "channel". Each channel has an associated "profile" that defines the syntax and semantics of the messages exchanged. Implicit in the operation of BXXP is the notion of channel management. In addition to defining BXXP’s channel management profile, this document defines:

- the TLS[5] transport security profile; and,

Other profiles, such as those used for data exchange, are defined by an application protocol designer. A registration template is provided for this purpose.
2. The Blocks eXtensible eXchange Protocol

BXXP is a message-oriented protocol. Arbitrary octets are encapsulated within a frame and tagged as either a request or a response. All interactions occur in the context of a channel -- a binding to a well-defined aspect of the application, such as transport security, user authentication, or data exchange.

During the creation of a channel, the requestor supplies one or more proposed profiles for that channel. If the responder creates the channel, it selects one of the profiles and returns it in a response; otherwise, it may indicate that none of the profiles are acceptable, and decline creation of the channel.

There are no other management capabilities for channels other than creation, as channel usage falls into one of two categories:

initial tuning: these are used by profiles that perform initialization once the BXXP session is established (e.g., negotiating the use of transport security); although several request/response exchanges may be required to perform the initialization, these channels become inactive early in the BXXP session and remain so for the duration.

continuous: these are used by profiles that support data exchange; typically, these channels are created after the initial tuning channels have gone quiet.

2.1 Roles

Although BXXP is a peer-to-peer protocol, it is convenient to label each peer in the context of the role it is performing at a given time:

- When a BXXP session is established, we designate the peer that awaits new connections as acting in the listening role, and the other peer, which establishes a connection to the listener, as acting in the initiating role. In the examples which follow, these are referred to as "I:" and "L:,", respectively.

- We designate a BXXP peer making a request as a client (or requestor); similarly, we designate the other BXXP peer as a server (or responder). In the examples which follow, these are referred to as "C:" and "S:,", respectively.

Typically, a BXXP peer acting in the server role is also acting in a listening role. However, because BXXP is peer-to-peer in nature, no such requirement exists.
2.2 Messages and Frames

In BXXP, there are three kinds of messages: requests, responses, and sequence updates.

Each request or response conveys data, which is segmented as the payload of one or more frames. Each frame consists of a header, the payload, and a trailer. The header and trailer are each represented using printable ASCII characters and are terminated with a CRLF pair. Between the header and the trailer is the payload, consisting of zero or more octets.

For example, here is a request message whose data is contained in a single frame that contains a payload of 94 octets spread over 3 lines (each line of the data is terminated with a CRLF pair):

C: REQ . 1 14 94 0
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/sasl/OTP' /> 
C: </start>
C: END

Note that the header is two lines long (the second line is blank signifying a lack of explicit MIME typing information).

The sequence update message is used to flow control request and response messages, and is represented using printable ASCII characters terminated by a CRLF pair.

For example, here is a sequence update message:

C: SEQ 1 0 65535

Note that the sequence update message doesn’t have a header, payload, or trailer -- it’s simply a single line.
2.2.1 Message Syntax

The ABNF for a message is:

```plaintext
message = frame / seq
frame = header payload trailer
header = req / rsp
req = "REQ" SP more SP serial SP seqno SP size SP channel CR LF [[mime] CR LF]
rsp = "RSP" SP more SP serial SP seqno SP size SP status CR LF [[mime] CR LF]
more = "." / "**"

; use of 0 for <serial> is reserved for the initial greeting
serial = 0..2147483647
seqno = 0..4294967295
size = 0..2147483647

; use of 0 for <channel> is reserved for BXXP channel management
channel = 0..255

; defaults are:
;
;     Content-Type: text/xml
;     Content-Transfer-Encoding: binary
;
mime = <MIME Content {entity-headers} from RFC 2045>
status = "+" / "-"
payload = *OCTET
trailer = "END" CR LF
seq = "SEQ" SP channel SP ackno SP window CR LF
ackno = seqno
window = size
```
2.2.1.1 Frame Header

The frame header consists of a three-character keyword (one of: "REQ" or "RSP"), followed by a continuation indicator, a serial number, a sequence number, a payload size, and one additional parameter. A single space character (decimal code 32, " ") separates each component. The header is terminated with a CRLF pair.

The "REQ" keyword indicates that this frame is part of a request message. Following the "REQ" keyword, the continuation indicator, the serial number, the sequence number, and the payload size is the channel number for the request.

The "RSP" keyword indicates that this frame is part of a response message. Following the "RSP" keyword, the continuation indicator, the serial number, the sequence number, and the payload size is the status indicator for the response.

The continuation indicator (one of: decimal code 42, "*", or decimal code 46, ".") specifies whether this is the final frame of the message:

intermediate ("*"): at least one other frame follows for the message; or,

complete ("."): this frame completes the data for the message.

The serial number must be a non-negative integer (in the range 0..2147483647) and have a different value than all other outstanding request messages (regardless of channel number).

The sequence number must be a non-negative integer (in the range 0..4294967295) and specifies the sequence number of the first octet in the payload, for the associated channel.

The payload size must be a non-negative integer (in the range 0..2147483647) and specifies the exact number of octets in the payload. (This does not include the trailer.)

The status indicator (one of: decimal code 43, "+", or decimal code 45, "-"), specifies whether the request corresponding to this response was performed:

positive ("+"): the request was performed and the response’s data contains the corresponding results; or,

negative ("-"): the request could not be performed (either for transient or permanent reasons) and the response’s data contains the corresponding error information.
There are several rules for identifying poorly-formed frames:

- if the header doesn’t start with "REQ" or "RSP";
- if the header starts with "REQ" or "RSP", but any of the continuation indicator, serial number, sequence number, or payload size cannot be determined or is invalid;
- if the header starts with "REQ", but the channel number cannot be determined or is invalid;
- if the header starts with "RSP", but the status indicator cannot be determined or is invalid;
- if the header starts with "RSP", but the serial number does not refer to an outstanding request message;
- if the value of the sequence number doesn’t correspond to the expected value for the associated channel (c.f., Section 2.5.3);
- if the header starts with "REQ" and refers to a message for which at least one other "REQ" frame has been received, and if the continuation indicator of the immediately-previous received frame for this message is intermediate ("*"), and if the channel numbers aren’t identical; or,
- if the header starts with "RSP" and refers to a message for which at least one other "RSP" frame has been received, and if the status indicator of this frame and the immediately-previous received frame for this message are not identical.

If a frame is poorly-formed, then the connection is closed without generating a response, and it is recommended that a diagnostic entry be logged.

The final frame in a message has a continuation indicator of complete ("."), whilst all earlier frames (if any) have a continuation indicator of intermediate ("*"). Note that any of these frames may have an empty payload, e.g.,

```
S: RSP * 1 284 25 +
S:
S: ...
S: ...
S: ...
S: END
S: RSP . 1 309 0 +
S:
S: END
```
2.2.1.2 Frame Payload

The data conveyed with a message is structured according to the rules of MIME. Accordingly, the header of the first frame for a message may include "entity-headers" (c.f., MIME[4]'s Section 3). If none, or only some, of the entity-headers are present:

- the default "Content-Type" is "text/xml"; and,
- the default "Content-Transfer-Encoding" is "binary".

Hence, in the absence of typing information, a message's data is a well-formed XML[3] document.

Note that the "entity-headers" (and the empty line that follows) are part of the of the header, not the payload. Thus, they do not contribute to the size of the payload.

2.2.1.3 Frame Trailer

The frame trailer consists of "END" followed by a CRLF pair.

When receiving a frame, if the characters immediately following the payload don’t correspond to a trailer, then the connection is closed without generating a response, and it is recommended that a diagnostic entry be logged.
2.2.2 Frame Semantics

The semantics of the payload of each frame is channel-specific. Accordingly, the profile associated with a channel must define:

- the initialization messages, if any, exchanged during channel creation;
- the set of request and response messages may be carried in the payload of the channel; and,
- the semantics of these messages.

A profile registration template (Section 5) organizes this information.

Note that if a profile uses XML to structure its messages, then only XML’s baseline facilities (as described in the XML 1.0 specification[3]) are allowed. Additional XML features (e.g., namespaces) are made available only by being referenced and allowed in a given profile’s specification.

In particular this limitation allows use of only the five predefined general entities references ("&amp;", "&lt;", "&gt;", "&apos;", and "&quot;") and numeric entity references in the messages exchanged.

Finally, because the profile registration template defines the messages exchanged over a channel, the XML documents exchanged in each message needn’t have either a "XML" declaration (e.g., <xml version="1.0" ?>) or a "DOCTYPE" declaration (e.g., <!DOCTYPE ...>). Of course, all other XML 1.0 instructions (e.g., CDATA blocks, processing instructions, and so on) are allowed.
2.3 Channel Management

When a BXXP session starts, only channel number 0 is defined, which is used for channel management. Section 6.1 contains the profile registration for BXXP channel management.

2.3.1 Message Semantics

2.3.1.1 The Start Message

When a BXXP peer wants to create a channel, it sends a "start" element as data on channel 0, e.g.,

I: REQ . 1 14 94 0
I:
I: <start number='1'>
I:  <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
I:  </start>
I: END

The "start" element has a "number" attribute, an optional "serverName" attribute, and one or more "profile" elements:

- the "number" attribute indicates the channel number (in the range 1..255) used to identify the channel in future messages;
- the "serverName" attribute, an arbitrary string, indicates the desired server name for this BXXP session; and,
- each "profile" element contained within the "start" element identifies a profile, and, optionally, contains an arbitrary XML element exchanged during channel creation as its content.

To avoid conflict in assigning channel numbers when requesting the creation of a channel, BXXP peers acting in the initiating role use only positive integers that are odd-numbered; similarly, BXXP peers acting in the listening role use only positive integers that are even-numbered.

The "serverName" attribute for the first successful "start" element received by a BXXP peer is memorable. (If the attribute isn’t present or it’s value is empty, then the sending BXXP peer is requesting a configuration-specific default value.) The BXXP peer decides whether to operate as the indicated "serverName"; if not, an "error" element is returned as data in a negative "RSP" message.

When a BXXP peer receives a "start" element as data on channel 0, it examines each of the proposed profiles, and decides whether to use one of them to create the channel. If so, the appropriate "profile"
element is returned as data in a positive "RSP" message; otherwise, an "error" element is returned as data in a negative "RSP" message.

When creating the channel, the value of the "serverName" attribute from the first successful "start" element is consulted to provide configuration information, e.g., the desired server-side certificate when starting the TLS transport security profile (Section 3.1).

For example, a successful channel creation might look like this:

```
I: REQ . 1 14 171 0
I:
I: <start number='1'>
I:   <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
I:   <profile
I:      uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
I: </start>
I: END
L: RSP . 1 284 61 +
L:
L: <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L: END
```

Similarly, an unsuccessful channel creation might look like this:

```
I: REQ . 1 14 94 0
I:
I: <start number='2'>
I:   <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
I: </start>
I: END
L: RSP . 1 284 89 -
L:
L: <error code='501'>number attribute
L: in &lt;start&gt; element must be odd-valued</error>
L: END
```
Finally, here’s an example in which an initialization element is exchanged during channel creation:

```
C: REQ . 1 14 120 0
C:
C: <start number='1'>
C: <profile uri='http://xml.resource.org/profiles/TLS'>
C: <ready />
C: </profile>
C: </start>
C: END
S: RSP . 1 84 83 +
S:
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S: <proceed />
S: </profile>
S: END
```
2.3.1.2 The Greeting Message

When a BXXP session is established, each BXXP peer signifies its availability by immediately sending a positive "RSP" message with a serial number of zero that contains a "greeting" element, e.g.,

L: <wait for incoming connection>
I: <open connection>
L: RSP . 0 0 84 +
L:
L: <greeting>
L: <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END

Note that this example implies that the BXXP peer in the initiating role waits until the BXXP peer in the listening role sends its greeting -- this is an artifact of the presentation; in fact, both BXXP peers send their response messages independently.

The "greeting" element has two optional attributes ("features" and "localize") and zero or more "profile" elements, one for each profile supported by the BXXP peer acting in a server role:

- the "features" attribute, if present, contains one or more feature tokens, each indicating an optional feature of the channel management profile supported by the BXXP peer;

- the "localize" attribute, if present, contains one or more language tokens, each identifying a desirable language tag to be used by the remote BXXP peer when generating textual diagnostics for the "error" element (the tokens are ordered from most to least desirable); and,

- each "profile" element contained within the "greeting" element identifies a profile, and unlike the "profile" elements that occur within the "start" element, the content of each "profile" element may not contain an optional initialization element.

At present, there are no optional features defined for the channel management profile.

Each token in the value of the "localize" attribute is defined according to [7]. If not present, the default is "i-default".
2.3.1.3 The Error Message

When a BXXP peer declines the creation of a channel, it returns an "error" element as data in a negative "RSP" message, e.g.,

I: REQ . 1 14 89 0
I:
I: <start number='2'>
I:  <profile uri='http://xml.resource.org/profiles/FOO' />
I: </start>
I: END
L: RSP . 1 284 67 -
L:
L: <error code='550'>all requested profiles are
L: unsupported</error>
L: END

The "error" element has a "code" attribute, an optional "xml:lang" attribute, and an optional textual diagnostic as its content:

- the "code" attribute is a three digit reply code meaningful to programs (c.f., Section 7);

- the "xml:lang" attribute identifies the language that the element's content is written in (the value is suggested, but not mandated, by the "localize" attribute of the "greeting" element sent by the remote BXXP peer); and,

- the textual diagnostic (which may be multiline) is meaningful to implementors, perhaps administrators, and possibly even users.

Note that if the textual diagnostic is present, then the "xml:lang" attribute is absent only if the language indicated as the remote BXXP's first choice is used.

In addition, a BXXP peer returns an "error" element whenever:

- it receives a "REQ" message containing an unexpected element; or,

- a BXXP session is established, the BXXP peer is acting in the listening role, and that BXXP peer is unavailable (in this case, the BXXP acting in the listening role does not send a "greeting" element).

In the latter case, both BXXP peers close the connection, and it is recommended that a diagnostic entry be logged by both BXXP peers.
2.4 Session Establishment and Release

When a BXXP session is established, each BXXP peer signifies its availability by immediately sending a positive "RSP" message with a serial number of zero that contains a "greeting" element, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: RSP . 0 0 84 +
L:
L: <greeting>
L:    <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END
```

which, for the BXXP peer acting in the listening role, indicates that it is available.

Alternatively, if the BXXP peer acting in the listening role is unavailable, it returns a negative response, e.g.,

```
L: <wait for incoming connection>
I: <open connection>
L: RSP . 0 0 22 -
L:
L: <error code='421' />
L: END
I: <close connection>
L: <close connection>
L: <wait for next connection>
```

and the "greeting" element sent by the BXXP peer acting in the initiating role is ignored. It is recommended that a diagnostic entry be logged by both BXXP peers.
When a BXXP peer wants to release the BXXP session, it sends a "REQ" message on channel 0 with no data. The other BXXP peer may accept the request (by sending a positive "RSP" message), e.g.,

```
C: REQ . 1 14 0 0
C:
C: END
S: RSP . 1 284 0 +
S:
S: END
C: <close connection>
S: <close connection>
L: <wait for next connection>
```

If the other BXXP peer sends a negative "RSP" message, then the connection should remain open, if possible.
2.5 Flow Control

Although the underlying transport service imposes flow control on a per-connection basis, if multiple channels are simultaneously in use on a connection, BXXP must provide a mechanism to avoid starvation and deadlock. To achieve this, BXXP re-introduces mechanisms used by the TCP: sequence numbers and window-based flow control. Briefly, each channel has a sliding window that indicates the number of payload octets that a peer may transmit before receiving further permission.

Every payload octet sent in each direction on a channel has an associated sequence number. Numbering of payload octets within a frame is such that the first payload octet is the lowest numbered, and the following payload octets are numbered consecutively.

The actual sequence number space is finite, though very large, ranging from 0..4294967295 (2**32 - 1). Since the space is finite, all arithmetic dealing with sequence numbers is performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 - 1 to 0 again.

2.5.1 Channel Creation

When a channel is created, the sequence number associated with the first payload octet of the first frame is 0, and the initial window size for that channel is 4096 octets. After channel creation, a BXXP peer may update the window size by sending a "SEQ" message (Section 2.5.4).

If a BXXP peer is requested to create a channel and it is unable to allocate at least 4096 octets for that channel, it must decline creation of the channel (Section 2.3.1.3). Similarly, during establishment of the BXXP session, if the BXXP peer acting in the listening role is unable to allocate at least 4096 octets for channel 0, then it must return a negative response (Section 2.4) instead of a greeting.
2.5.2 Sending REQ or RSP Messages

Before a message is sent, the sending BXXP peer must ensure that the size of the payload is within the window advertised by the receiving BXXP peer. If not, it has three choices:

- if the window would allow for at least one payload octet to be sent, the BXXP peer may segment the message and start by sending a smaller frame (up to the size of the remaining window);
- the BXXP peer may delay sending the message until the window becomes larger; or,
- the BXXP peer may signal to its application that it is unable to send the message, allowing the application to try again at a later time (or perhaps signalling its application when a larger window is available.)

The choice is implementation-dependent, although it is recommended that the application using BXXP be given a mechanism for influencing the decision.

2.5.3 Receiving REQ or RSP Messages

When a frame is received, the sum of its sequence number and payload size, modulo 4294967296 (2**32), gives the expected sequence number associated with the first payload octet of the next frame received. Accordingly, when receiving a frame if the sequence number isn’t the expected value for this channel, then the BXXP peers have lost synchronization, then the connection is closed without generating a response, and it is recommended that a diagnostic entry be logged.
2.5.4 Processing SEQ Messages

As an application accepts responsibility for incoming frames, its BXXP peer should send "SEQ" messages to advertise a new window.

The "SEQ" message has three parameters:

- a channel number;
- an acknowledgement number, that indicates the value of the next sequence number that the sender is expecting to receive on this channel; and,
- a window size, that indicates the number of payload octets beginning with the one indicated by the acknowledgement number that the sender is expecting to receive on this channel.

A single space character (decimal code 32, " ") separates each component. The "SEQ" message is terminated with a CRLF pair.

When a "SEQ" message is received, if any of the channel number, acknowledgement number, or window size cannot be determined or is invalid, then the connection is closed without generating a response, and it is recommended that a diagnostic entry be logged.
2.5.5 Use of Flow Control

The key to successful use of flow control within BXXP is to balance performance and fairness:

- large messages should be segmented into multiple frames (e.g., the BXXP segment size should be no larger than TCP’s negotiated maximum segment size minus some small constant);
- frames for different channels with traffic ready to send should be sent in a round-robin fashion; and,
- a "SEQ" message should be sent each time a "REQ" or "RSP" message is received (if the transport service presents multiple messages to a BXXP peer simultaneously, then a single consolidating "SEQ" message may be sent).

In order to avoid pathological interactions with the transport service, it is important that a BXXP peer advertise windows based on available buffer space, to allow data to be read from the transport service as soon as available. Further, "SEQ" messages for a channel should have higher priority than "REQ" or "RSP" messages for that channel.

Finally, implementations may wish to provide queue management facilities to the application using BXXP, e.g., channel priorities. In particular, implementations should not allow a given channel to monopolize the underlying transport window (e.g., slow readers should get small windows).
2.6 Parallelism

2.6.1 Within a single channel

A BXXP peer acting in the client role may send multiple "REQ" messages for the same channel without waiting to receive the corresponding "RSP" messages. A BXXP peer acting in the server role must process all "REQ" messages for a given channel in the same order as they are received. As a consequence, that BXXP peer must generate the corresponding "RSP" messages in the same order as the "REQ" messages are received.

2.6.2 Between different channels

A BXXP peer acting in the client role may send multiple "REQ" messages for different channels without waiting to receive the corresponding "RSP" messages. A BXXP peer acting in the server role may process "REQ" messages received for different channels in parallel. As a consequence, although the "RSP" messages for a given channel are generating according to the order in which the corresponding "REQ" messages are received, there is no ordering constraint between "RSP" messages for different channels.

2.6.3 Pre-emptive responses

A BXXP peer acting in the server role may send a negative response to a request before it receives the final "REQ" frame of a request. If it does so, that BXXP peer is obliged to ignore any subsequent "REQ" frames for that request, up to and including the final "REQ" frame.

If a BXXP peer acting in the client role receives a negative "RSP" frame before it sends the final "REQ" frame for a request, then it is required to send a "REQ" frame with a continuation status of complete (".") and having a zero-length payload.

2.6.4 Interference

If the processing of a particular frame has sequencing impacts on other frames (either intra-channel or inter-channel), then the corresponding profile should define this behavior, e.g., a profile whose messages alter the underlying transport service.
Peer-to-Peer Behavior

BXXP is a peer-to-peer protocol -- as such both peers must be prepared to receive both "REQ" and "RSP" frames. Accordingly, an initiating BXXP peer capable of acting only in the client role must behave gracefully if it receives a "REQ" message. Accordingly, all profiles must provide an appropriate error message for responding to unwanted requests.

As a consequence of the peer-to-peer nature of BXXP, serial numbers are unidirectionally-significant. That is, the serial numbers in "REQ" messages sent by a BXXP peer acting in the initiating role are unrelated to the serial numbers in "REQ" messages sent by a BXXP peer acting in the listening role.

For example, these two frames:

```
I: REQ . 1 14 94 0
I:
I: <start number='1'>
I:   <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
I: </start>
I: END
L: REQ . 1 284 89 0
L:
L: <start number='2'>
L:   <profile uri='http://xml.resource.org/profiles/SEP' />
L: </start>
L: END
```

have no fundamental relationship to each other.
3. Transport Security

When a BXXP session starts, plaintext transfer, without privacy, is provided. Accordingly, transport security in BXXP is achieved using an initial tuning profile.

This document defines one profile:

- the TLS transport security profile, based on TLS version one[5].

Other profiles may be defined and deployed on a bilateral basis.

When a channel associated with transport security begins the underlying negotiation process, all channels (including channel 0), are closed on the BXXP session. Upon completion of the negotiation process, regardless of its outcome, a new greeting is issued by both BXXP peers.
A BXXP peer may choose to issue different greetings based on whether privacy is in use, e.g.,

L: <wait for incoming connection>
I: <open connection>
L: RSP . 0 0 84 +
L:
L: <greeting>
L:  <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END
I: REQ . 1 14 120 0
I:
I: <start number='1'>
I:  <profile uri='http://xml.resource.org/profiles/TLS'>
I:   <ready />
I:  </profile>
I: </start>
I: END
L: RSP . 1 84 83 +
L:
L: <profile uri='http://xml.resource.org/profiles/TLS'>
L:  <profile uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
L:  <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L:  <profile uri='http://xml.resource.org/profiles/SEP' />
L: </profile>
L: END

... successful transport security negotiation ...

L: RSP . 0 0 224 +
L:
L: <greeting>
L:  <profile
L:       uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
L:  <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L:  <profile uri='http://xml.resource.org/profiles/SEP' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END
Of course, not all BXXP peers need be as single-minded:

L: <wait for incoming connection>
I: <open connection>
L: RSP . 0 0 284 +
L:
L: <greeting>
L:   <profile
L:     uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
L:   <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L:   <profile uri='http://xml.resource.org/profiles/SEP' />
L:   <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END
I: REQ . 1 14 120 0
I:
I: <start number='1'>
I:   <profile uri='http://xml.resource.org/profiles/TLS'>
I:     <ready />
I:   </profile>
I: </start>
I: END
L: RSP . 1 284 83 +
L:
L: <profile uri='http://xml.resource.org/profiles/TLS'>
L:   <proceed />
L: </profile>
L: END

... failed transport security negotiation ...

L: RSP . 0 0 284 +
L:
L: <greeting>
L:   <profile
L:     uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
L:   <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
L:   <profile uri='http://xml.resource.org/profiles/SEP' />
L:   <profile uri='http://xml.resource.org/profiles/TLS' />
L: </greeting>
L: END
I: RSP . 0 0 14 +
I:
I: <greeting />
I: END
3.1 The TLS Transport Security Profile

Section 6.3 contains the registration for this profile.

3.1.1 Profile Identification and Initialization

The TLS transport security profile is identified as:

http://xml.resource.org/profiles/TLS

in the BXXP "profile" element during channel creation.

During channel creation, the corresponding "profile" element in the BXXP "start" element may contain a "ready" element. If channel creation is successful, then before sending the corresponding "RSP" message, the BXXP peer processes the "ready" element and includes the resulting response in the "RSP" message, e.g.,

C: REQ . 1 14 120 0
C:
C: <start number='1'>
C:    <profile uri='http://xml.resource.org/profiles/TLS'>
C:        <ready />
C:    </profile>
C: </start>
C: END
S: RSP . 1 84 83 +
S:
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S:     <proceed />
S: </profile>
S: END
Note that it is possible for the channel to be created, but for the encapsulated operation to fail, e.g.,

C: REQ . 1 14 135 0
C: <start number='1'>
C:   <profile uri='http://xml.resource.org/profiles/TLS'>
C:     <ready version='oops' />  
C:   </profile>
C: </start>
C: END
S: RSP . 1 84 156 +
S: <profile uri='http://xml.resource.org/profiles/TLS'>
S:   <error code='501'>version attribute poorly formed in &lt;ready&gt; element</error>
S: </profile>
S: END

In this case, a positive "RSP" message is returned (as channel creation succeeded), but the encapsulated response contains an indication as to why the operation failed.

3.1.2 Request and Response Messages

Section 6.4 defines the messages that are used in the TLS transport security profile:

- "REQ" messages carry only the "ready" element as data;
- positive "RSP" messages carry only the "proceed" element as data; and,
- negative "RSP" messages carry only the "error" element as data.
3.1.3 Message Semantics

3.1.3.1 The Ready Message

The "ready" element has an optional "version" attribute and no content:

- the "version" element defines the earliest version of TLS acceptable for use.

When a BXXP peer sends the "ready" element, it no longer sends any traffic on any channel until a corresponding "RSP" message is received; similarly, before processing a "ready" element, the receiving BXXP peer waits until any pending "RSP" messages have been generated and sent.

3.1.3.2 The Proceed Message

The "proceed" element has no attributes and no content. It is sent in response to the "ready" element. When a BXXP peer receives the "ready" element, it begins the underlying negotiation process for transport security.
4. User Authentication

When a BXXP session starts, anonymous access, without trace information, is provided. Accordingly, user authentication in BXXP is achieved using an initial tuning profile.

This document defines a family of profiles based on SASL mechanisms:

- Each mechanism in the IANA SASL registry[13] has an associated profile.

Other profiles may be defined and deployed on a bilateral basis.

Whenever a successful authentication occurs, on any channel, the authenticated identity is updated for all existing and future channels on the BXXP session; further, no additional attempts at authentication are allowed.

Note that regardless of transport security and user authentication, authorization is an internal matter for each BXXP peer. As such, each peer may choose to restrict the operations it allows based on the authentication credentials provided (i.e., unauthorized operations are rejected with error code 530).
4.1 The SASL Family of Profiles

Section 6.5 contains the registration for this profile.

Note that SASL may provide both user authentication and transport security. Once transport security is successfully negotiated for a BXXP session, then a SASL security layer may not be negotiated; similarly, once any SASL negotiation is successful, a transport security profile may not be started or otherwise used.

Section 4 of the SASL specification[6] requires the following information be supplied by a protocol definition:

- **service name**: "bxxp" will be registered with the IANA as a GSSAPI service name when this draft is published as an RFC.

- **initiation sequence**: Creating a channel using a BXXP profile corresponding to a SASL mechanism starts the exchange. An optional parameter corresponding to the "initial response" sent by the client is carried within a "blob" element during channel creation.

- **exchange sequence**: "Challenges" and "responses" are carried in the "blob" element during data exchange. The "status" attribute of the "blob" element is used both by a server indicating a successful completion of the exchange, and a client aborting the exchange. The server indicates failure of the exchange by sending an "error" element.

- **security layer negotiation**: Prior to beginning the negotiation of a security layer, any pending "RSP" messages are generated and sent; further, once negotiation begins, no traffic is sent on any other channels until the negotiation completes.

  If a security layer is successfully negotiated, it takes effect immediately following the message that concludes the server’s successful completion reply. When a security layer takes effect, all channels (including channel 0), are closed on the BXXP session, and a new greeting is issued by both BXXP peers.

- **use of the authorization identity**: This is made available to all channels for the duration of the BXXP session.
4.1.1 Profile Identification and Initialization

Each SASL mechanism registered with the IANA is identified as:

http://xml.resource.org/profiles/sasl/MECHANISM

where "MECHANISM" is the token assigned to that mechanism by the IANA.

Note that during channel creation, a BXXP peer may provide multiple profiles to the remote peer, e.g.,

C: REQ . 1 14 171 0
C:
C: <start number='1'>
C:  <profile
C:    uri='http://xml.resource.org/profiles/sasl/ANONYMOUS' />
C:  <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
C: </start>
C: END
S: RSP . 1 284 61 +
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP' />
S: END

During channel creation, the corresponding "profile" element in the BXXP "start" element may provide data in a "blob" element. Note that it is possible for the channel to be created, but for the encapsulated operation to fail, e.g.,

C: REQ . 1 14 145 0
C:
C: <start number='1'>
C:  <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
C:    <blob>AGJsb2NrbWFzdGVy</blob>
C:  </profile>
C: </start>
C: END
S: RSP . 1 284 61 +
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
S:   <error code='534'>authentication mechanism is too weak</error>
S: </profile>
S: END

In this case, a positive "RSP" message is returned (as channel creation succeeded), but the encapsulated response contains an indication as to why the operation failed.
Otherwise, the server returns a challenge (or signifies success), e.g.,

C: REQ . 1 14 145 0
C:
C: <start number='1'>
C:   <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
C:     <blob>AGJsb2NrbWFzdGVy</blob>
C:   </profile>
C: </start>
S: RSP . 1 284 144 +
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/OTP'>
S:     <blob>b3RwLXNoYTEgOTk5NyBwaXh5bWlzYXM4NTgwNSB1eHQ=</blob>
S: </profile>
S: END

If a challenge is received, then the client responds and awaits a reply, e.g.,

C: REQ . 2 0 67 1
C:
C: <blob>d29yZDpmZXJuIghhbmcgYnJvdyBib25nIgh1cmQgdG9n</blob>
C: END
S: RSP . 2 0 13 +
S:
S: <blob status='complete' />
S: END

Of course, the client could abort the authentication process by sending "<blob status='abort' />") instead.

Alternatively, the server might reject the response with an error: e.g.,

C: REQ . 2 0 67 1
C:
C: <blob>d29yZDpmZXJuIghhbmcgYnJvdyBib25nIgh1cmQgdG9n</blob>
C: END
S: RSP . 2 0 22 -
S:
S: <error code='535' />
Finally, depending on the SASL mechanism, an initialization element may be exchanged unidirectionally during channel creation, e.g.,

C: REQ . 1 14 107 0
C:
C: <start number='1'>
C:  <profile
C:     uri='http://xml.resource.org/profiles/sasl/CRAM-MD5' />
C: </start>
C: END
S: RSP . 1 284 148 +
S:
S: <profile uri='http://xml.resource.org/profiles/sasl/CRAM-MD5'>
S:  <blob>PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9uLm1jaS5uZXQ+
S:    </blob>
S: </profile>
S: END

Note that this example implies that the "blob" element in the server's reply appears on two lines -- this is an artifact of the presentation; in fact, only one line is used.

4.1.2 Request and Response Messages

Section 6.6 defines the messages that are used for each profile in the SASL family:

o "REQ" messages carry only the "blob" element as data;

o positive "RSP" messages carry only the "blob" element as data; and,

o negative "RSP" messages carry only the "error" element as data.

Because many SASL mechanisms exchange binary data, the content of the "blob" element is always a base64-encoded string.
4.1.3 Message Semantics

The "blob" element has an optional "status" attribute, and arbitrary octets as its content:

- the "status" attribute, if present, takes one of three values:
  - abort: used by a client to indicate that it is aborting the authentication process;
  - complete: used by a server to indicate that the exchange is complete and successful; or,
  - continue: used by either a client or server, otherwise.

Finally, note that SASL’s EXTERNAL mechanism works with an "external authentication" service, which is provided by one of:

- a transport security profile, capable of providing authentication information (e.g., Section 3.1), being active on the connection;
- a network service, capable of providing strong authentication (e.g., IPSec[10]), underlying the connection; or,
- a locally-defined security service.

For authentication to succeed, two conditions must hold:

- an external authentication service must be active; and,
- if present, the authentication identity must be consistent with the credentials provided by the external authentication service (if the authentication identity is empty, then an authorization identity is automatically derived from the credentials provided by the external authentication service).
5. Profile Registration Template

When a profile is registered, the following information is supplied:

Profile Identification: specify a URI[8] that authoritatively identifies this profile.

Elements Exchanged during Channel Creation: specify the elements that may be exchanged during channel creation (note that if the profile doesn’t exchange XML elements, then initialization information may not be exchanged during channel creation).

Messages in "REQ" frames: specify the datatypes that may be present in a request.

Messages in positive "RSP" frames: specify the datatypes that may be present in a positive response.

Messages in negative "RSP" frames: specify the datatypes that may be present in negative response.

Message Syntax: specify the syntax of the datatypes exchanged by the profile.

Message Semantics: specify the semantics of the datatypes exchanged by the profile.

Note that "datatype" refers to any MIME media type, whilst "element" refers to any well-formed XML document.
6. Initial Profile Registrations

6.1 BXXP Channel Management

Profile Identification: not applicable

Elements Exchanged during Channel Creation: not applicable

Messages in "REQ" frames: "start"

Messages in positive "RSP" frames: "greeting" or "profile"

Messages in negative "RSP" frames: "error"

Message Syntax: c.f., Section 6.2

Message Semantics: c.f., Section 2.3.1
6.2 BXXP Channel Management DTD

<!--
DTD for BXXP Channel Management, as of 2000-04-04

Copyright 1999, 2000 Invisible Worlds, Inc.

This document is a DTD and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted.

Refer to this DTD as:

<!ENTITY % BXXP PUBLIC "-//Blocks//DTD BXXP//EN"
    "http://xml.resource.org/profiles/BXXP/bxxp.dtd">
%BXXP;
-->

<!--
DTD data types:

<table>
<thead>
<tr>
<th>entity</th>
<th>syntax/reference</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>a channel number</td>
<td>PINT8</td>
<td>1..255</td>
</tr>
<tr>
<td>one or more feature tokens, seperated by space</td>
<td>FTRS</td>
<td>NMTOKENS</td>
</tr>
<tr>
<td>a 3-digit reply code</td>
<td>XYZ</td>
<td>[1-5][1-9][1-9]</td>
</tr>
</tbody>
</table>
-->

<!ENTITY % PINT8   "CDATA">  
<!ENTITY % DTD     "CDATA">  
<!ENTITY % FTRS    "NMTOKENS">  
<!ENTITY % LOCS    "NMTOKENS">  
<!ENTITY % XYZ     "CDATA">
<!--
BXXP messages

<table>
<thead>
<tr>
<th>role</th>
<th>REQ</th>
<th>RSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>I and L</td>
<td></td>
<td>+: greeting</td>
</tr>
<tr>
<td>I or L</td>
<td>start</td>
<td>+: profile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-: error</td>
</tr>
</tbody>
</table>

<!ELEMENT greeting (profile)*>  
<!ATTLIST greeting
  features       %FTRS;    #IMPLIED
  localize       %LOCS;    "i-default">

<!ELEMENT start (profile)+>
<!ATTLIST start
  number         %PINT8;   #REQUIRED
  serverName     CDATA     #IMPLIED>

<!ELEMENT profile ANY>
<!ATTLIST profile
  uri            %URI;     #REQUIRED>

<!ELEMENT error (#PCDATA)+>
<!ATTLIST error
  code           %XYZ;     #REQUIRED
  xml:lang       %LANG;    #IMPLIED>
6.3 TLS Transport Security Profile Registration

Profile Identification: http://xml.resource.org/profiles/TLS

Elements Exchanged during Channel Creation: "ready"

Messages in "REQ" frames: "ready"

Messages in positive "RSP" frames: "proceed"

Messages in negative "RSP" frames: "error"

Message Syntax: c.f., Section 6.4

Message Semantics: c.f., Section 3.1.3
6.4 TLS Transport Security Profile DTD

<!--
DTD for the TLS Transport Security Profile, as of 2000-03-03

Copyright 1999, 2000 Invisible Worlds, Inc.

This document is a DTD and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted.

Refer to this DTD as:

<!ENTITY % TLS PUBLIC "-//Blocks//DTD TLS//EN"
        "http://xml.resource.org/profiles/TLS/tls.dtd">
%TLS;
-->

<!--
TLS messages

<table>
<thead>
<tr>
<th>role</th>
<th>REQ</th>
<th>RSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>I or L</td>
<td>ready</td>
<td>+: proceed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-: error</td>
</tr>
</tbody>
</table>
-->

<!ELEMENT ready       EMPTY>
<!ATTLIST ready
  version CDATA    "1">
<!ELEMENT proceed       EMPTY>
6.5 SASL Family of Profiles Registration

Profile Identification:
http://xml.resource.org/profiles/sasl/MECHANISM, where
"MECHANISM" is a token registered with the IANA[14]

Elements Exchanged during Channel Creation: "blob"

Messages in "REQ" frames: "blob"

Messages in positive "RSP" frames: "blob"

Messages in negative "RSP" frames: "error"

Message Syntax: c.f., Section 6.6

Message Semantics: c.f., Section 4.1.3
6.6 SASL Family of Profiles DTD

<!--
DTD for the SASL Family of Profiles, as of 2000-04-04

Copyright 1999, 2000 Invisible Worlds, Inc.

This document is a DTD and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted.

Refer to this DTD as:

<!ENTITY % SASL PUBLIC "-//Blocks//DTD SASL//EN"
  "http://xml.resource.org/profiles/sasl/sasl.dtd">
%SASL;
-->

SASL messages

<table>
<thead>
<tr>
<th>role</th>
<th>REQ</th>
<th>RSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>I or L</td>
<td>blob</td>
<td>+: blob</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-: error</td>
</tr>
</tbody>
</table>

<!ELEMENT blob (#PCDATA)*>
<!ATTLIST blob
  xml:space (default|preserve) "preserve"
  status (abort|complete|continue) "continue"
7. Reply Codes

<table>
<thead>
<tr>
<th>code</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>421</td>
<td>service not available</td>
</tr>
<tr>
<td>450</td>
<td>requested action not taken (e.g., lock already in use)</td>
</tr>
<tr>
<td>451</td>
<td>requested action aborted (e.g., local error in processing)</td>
</tr>
<tr>
<td>454</td>
<td>temporary authentication failure</td>
</tr>
<tr>
<td>500</td>
<td>general syntax error (e.g., poorly-formed XML)</td>
</tr>
<tr>
<td>501</td>
<td>syntax error in parameters (e.g., non-valid XML)</td>
</tr>
<tr>
<td>504</td>
<td>parameter not implemented</td>
</tr>
<tr>
<td>530</td>
<td>authentication required</td>
</tr>
<tr>
<td>534</td>
<td>authentication mechanism insufficient (e.g., too weak, sequence exhausted, etc.)</td>
</tr>
<tr>
<td>535</td>
<td>authentication failure</td>
</tr>
<tr>
<td>537</td>
<td>action not authorized for user</td>
</tr>
<tr>
<td>538</td>
<td>authentication mechanism requires encryption</td>
</tr>
<tr>
<td>550</td>
<td>requested action not taken (e.g., no requested profiles are acceptable)</td>
</tr>
<tr>
<td>553</td>
<td>parameter invalid</td>
</tr>
<tr>
<td>554</td>
<td>transaction failed (e.g., policy violation)</td>
</tr>
</tbody>
</table>
8. Security Considerations

The BXXP framing mechanism, per se, provides no protection against attack; however, judicious use of initial tuning profiles provides varying degrees of assurance:

1. If one of the profiles from the SASL family is used, refer to [6]’s Section 9 for a discussion of security considerations.

2. If the TLS transport security profile is used (or if a SASL security layer is negotiated), then:

   1. A man-in-the-middle may remove the security-related profiles from the BXXP greeting or generate an error response to the "ready" element of the TLS transport security profile. A BXXP peer may be configurable to refuse to proceed without an acceptable level of privacy.

   2. A man-in-the-middle may cause a down-negotiation to the weakest cipher suite available. A BXXP peer should be configurable to refuse weak cipher suites.

   3. A man-in-the-middle may modify any protocol interactions prior to a successful negotiation. Upon completing the negotiation, a BXXP peer must discard previously cached information about the BXXP session.

As different TLS ciphersuites provide varying levels of security, administrators should carefully choose which ciphersuites are provisioned.
9. IANA Considerations

The IANA maintains a list of:

- BXXP reply codes (c.f., Section 7); and,
- BXXP profiles that are defined in the RFC series.

In addition, prior to publication of this draft as an RFC, the IANA will be asked to register "bxxp" as a GSSAPI service name.
References


[11] mailto:blocks-request@invisible.net


[15] mailto:ddc@lcs.mit.edu

[16] mailto:dcrocker@brandenburg.com

[17] mailto:deering@cisco.com
Author’s Address

Marshall T. Rose
Invisible Worlds, Inc.
1179 North McDowell Boulevard
Petaluma, CA  94954-6559
US

Phone: +1 707 789 3700
EMail: mrose@invisible.net
URI:  http://invisible.net/
Appendix A. Acknowledgements

The author gratefully acknowledges the contributions of: David Clark[15], Dave Crocker[16], Steve Deering[17], Marco Gazzetta[18], Danny Goodman[19], Robert Herriot[20], Ben Laurie[21], Carl Malamud[22], Michael Mealling[23], Paul Mockapetris[24], RL 'Bob' Morgan[25], Frank Morton[26], Darren New[27], Chris Newman[28], Craig Partridge[29], Paul Vixie[30], and Daniel Woods[31]. In particular, Dave Crocker provided helpful suggestions on the nature of flow control and segmentation in the framing protocol.
Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.