[JET-PROTO]:
JET Relay Protocol

Revision History

Revision summary			
Author	**Date**	**Revision history**	**Comments**
Marc-André Moreau	10/25/2018	0.1	Initial draft
Marc-André Moreau	07/14/2019	0.2	Major update

Contents

1 Introduction 3
1.1 Glossary 3
1.2 References 3
1.2.1 Normative References 3
1.3 Overview 3
1.4 Prerequisites/Preconditions 3
1.5 Applicability Statement 3

2 Messages 4
2.1 Transport 4
2.2 Message Syntax 4
2.2.1 Protocol Messages 4
2.2.1.1 JET_PACKET 4
2.2.2 Protocol Examples 4
2.2.2.1 TCP Server Accept 6
2.2.2.2 TCP Client Connect 7
2.2.2.3 Connection Sequence 8

1 Introduction

This document specifies the JET Relay Protocol.

1.1 Glossary

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt.

1.3 Overview

The JET protocol bears some similarities with the SOCKS proxy protocol and the routing token packets often used in remote desktop connections for load balancing and session selection. However, all of these protocols make connections in a forward manner: the client connects to the proxy, then the proxy connects to the server and then relays the traffic. The JET protocol is designed to relay TCP traffic between a TCP client and server using only outgoing TCP connections.

1.4 Prerequisites/Preconditions

The JET protocol requires a TCP transport.

1.5 Applicability Statement

The JET protocol is suitable for simple, efficient relaying of TCP protocols.

2 Messages

2.1 Transport

The JET protocol is designed to provide a simple, efficient way to relay TCP traffic between two nodes that can only perform outgoing TCP connections to the same server, using a rendezvous connection style. Alternatively, a JET packet can be sent between a client and server as a way to discover a direct route.

2.2 Message Syntax

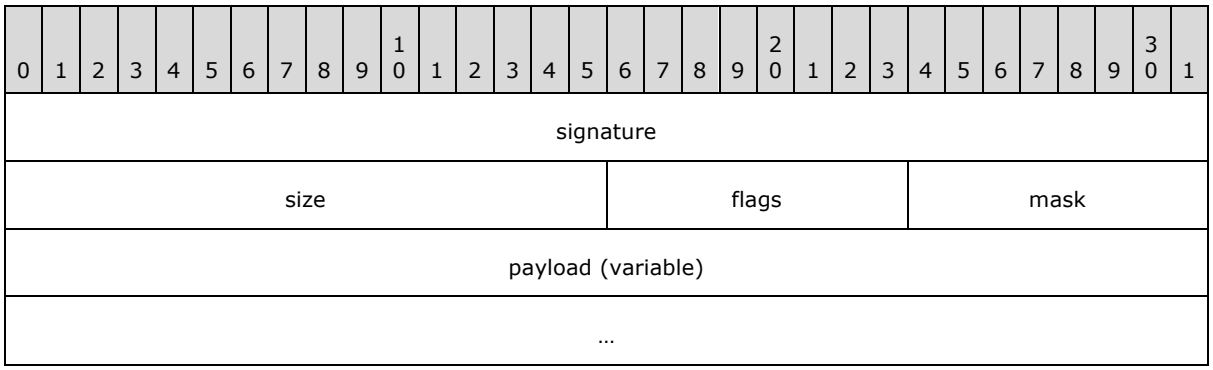
The following sections specify the JET protocol message syntax. All fields defined in this document use big endian byte ordering and are byte-aligned to their sizes (a field of 4 bytes starts at an offset that is a multiple of 4).

2.2.1 Protocol Messages

All JET binary protocol messages sent over TCP or TLS are contained within a JET_PACKET structure. The JET WebSocket protocol makes use of HTTP requests and the WebSocket handshake request path to achieve the same goal.

2.2.1.1 JET_PACKET

The JET_PACKET structure contains a routing token used to associate two indirect TCP connections.



signature (4 bytes): The packet signature. MUST be set to { 0x4A, 0x45, 0x54, 0x00 } ("JET").

size (2 bytes): The total size of the packet, including headers. The minimum size is 8.

flags (1 byte): This field is reserved for future use and MUST be set to zero.

mask (1 byte): This field contains a one-byte mask that MUST be applied to the payload.

payload (variable): This field contains a masked HTTP request or response using the value from the mask field.

2.2.2 Protocol Examples

Here is a sample TCP connection established using the JET protocol. To avoid interference from proxies, an 8-byte binary header is used to encapsulate HTTP requests and responses. The payload is masked using the value from the mask field.

2.2.2.1 REST API

The REST API is used to create Jet associations and optionally enforce authentication. It is optional for the JET binary protocol, but it is required for the JET WebSocket protocol.

2.2.2.1.1 Jet Association Creation

The JET WebSocket protocol requires that the association be created with an HTTP request/response. While this call could be made by a backend server, it is recommended to make the call from endpoint that will act as a server, such that DNS load balancing can be implemented properly by selecting the closest Jet instance.

The usage of an API key or authentication for this API endpoint is recommended but not required.

>> Request:

```
GET /jet/create HTTP/1.1

Host: jet.wayk.net

Jet-Version: 1
```

<< Response:

```
HTTP/1.1 200 OK

Jet-Association: e6ec698c-5793-4c63-af79-bd644ccf022f

Jet-Version: 1

{
  "s": "ok",
  "v": {
    "iceServers": [
      { "url": "stun:stun.wayk.net" },
      { "url": "jet:jet101.wayk.net:8080?transport=tcp" },
      { "url": "jet:jet101.wayk.net:443?transport=tls" },
      { "url": "jet:jet101.wayk.net?transport=wss" }
    ]
  }
}
```

2.2.2.1.2 Jet Candidate Format

A JET candidate is a URL representing a possible route that can be used for a connection.

If no scheme is specified, the "tcp://" scheme should be used. The default port can only be omitted for ws:// (80) and wss:// (443) schemes, otherwise it must be included in the URL.

tcp://jet101.wayk.net:8080

A TCP candidate using the JET binary protocol. This candidate is compatible with end-to-end TLS usage, where the JET exchange occurs prior to the TLS handshake.

tls://jet101.wayk.net:4343

A TLS candidate using the JET binary protocol. This candidate is not compatible with end-to-end TLS because the TLS handshake is performed with the Jet relay, followed by the JET exchange over TLS.

ws://jet101.wayk.net:8080

A WebSocket candidate without TLS. The WebSocket handshake request path contains the JET parameters.

wss://jet101.wayk.net:4343

A WebSocket candidate with TLS. The WebSocket handshake request path contains the JET parameters.

2.2.2.2 Binary Protocol

The JET binary protocol consists of a single request/response exchange at the beginning of the connection, either over TCP or over TLS. When done over TCP, a TLS handshake can follow. When done over TLS, the TLS handshake must be completed first, with the JET exchange done over TLS.

2.2.2.2.1 TCP Server Accept

>> Request:

GET / HTTP/1.1

Host: jet.wayk.net

Connection: Keep-Alive

Jet-Method: Accept

Jet-Version: 1

<< Response:

HTTP/1.1 200 OK

Jet-Association: e6ec698c-5793-4c63-af79-bd644ccf022f

Jet-Instance: jet101.wayk.net:8080

Jet-Version: 1

2.2.2.2.2 TCP Client Connect

>> Request:

GET / HTTP/1.1

Host: jet101.wayk.net

Connection: Keep-Alive

Jet-Method: Connect

Jet-Association: e6ec698c-5793-4c63-af79-bd644ccf022f

Jet-Version: 1

<< Response:

HTTP/1.1 200 OK

Jet-Version: 1

2.2.2.3 WebSocket Protocol

The JET WebSocket protocol consists of an HTTP request/response followed by a WebSocket handshake where the JET parameters are all encoded in the HTTP request path. The usage of the request path is necessary since it is not possible to add custom HTTP headers to a WebSocket handshake from a browser implementation.

2.2.2.3.1 WebSocket Server Accept

>> Request:

GET /jet/accept/<jet-association> HTTP/1.1

Host: jet101.wayk.net

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Version: 13

Sec-WebSocket-Key: B0mBaSTOZiC2RAkp3ScC8MA/gaI=

<< Response:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: B0mBaSTOZiC2RAkp3ScC8MA/gaI=

2.2.2.3.2 WebSocket Client Connect

>> Request:

GET /jet/connect/<jet-association> HTTP/1.1

Host: jet101.wayk.net

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Version: 13

Sec-WebSocket-Key: B0mBaSTOZiC2RAkp3ScC8MA/gaI=

<< Response:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: B0mBaSTOZiC2RAkp3ScC8MA/gaI=

2.2.2.4 Connection Sequence