Independent Submission
Request for Comments: 5572
Category: Experimental
ISSN: 2070-1721

M. Blanchet
Viagenie
F. Parent
Beon Solutions

February 2010

# IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)

## Abstract

A tunnel broker with the Tunnel Setup Protocol (TSP) enables the establishment of tunnels of various inner protocols, such as IPv6 or IPv4, inside various outer protocols packets, such as IPv4, IPv6, or UDP over IPv4 for IPv4 NAT traversal. The control protocol (TSP) is used by the tunnel client to negotiate the tunnel with the broker. A mobile node implementing TSP can be connected to both IPv4 and IPv6 networks whether it is on IPv4 only, IPv4 behind a NAT, or on IPv6 only. A tunnel broker may terminate the tunnels on remote tunnel servers or on itself. This document describes the TSP within the model of the tunnel broker model.

## Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5572.

## IESG Note

The content of this RFC was at one time considered by the IETF, and therefore it may resemble a current IETF work in progress or a published IETF work.

Table of Contents

## 1. Introduction

This document first describes the TSP framework, the protocol details, and the different profiles used. It then describes the applicability of TSP in different environments, some of which were described in the v6ops scenario documents.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. Description of the TSP Framework

Tunnel Setup Protocol (TSP) is a signaling protocol to set up tunnel parameters between two tunnel endpoints. TSP is implemented as a tiny client code in the requesting tunnel endpoint. The other endpoint is the server that will set up the tunnel service. TSP uses XML [W3C.REC-xml-2004] basic messaging over TCP or UDP. The use of XML gives extensibility and easy option processing.

TSP negotiates tunnel parameters between the two tunnel endpoints. Parameters that are always negotiated are:

- Authentication of the users, using any kind of authentication mechanism (through Simple Authentication and Security Layer (SASL) [RFC4422]) including anonymous

- Tunnel encapsulation:

    * IPv6 over IPv4 tunnels [RFC4213]

    * IPv4 over IPv6 tunnels [RFC2473]

    * IPv6 over UDP-IPv4 tunnels for NAT traversal

- IP address assignment for the tunnel endpoints
- DNS registration of the IP endpoint address (AAAA)

Other tunnel parameters that may be negotiated are:

- Tunnel keep-alive
- IPv6 prefix assignment when the client is a router
- DNS delegation of the inverse tree, based on the IPv6 prefix assigned





- Routing protocols

The tunnel encapsulation can be explicitly specified by the client, or can be determined during the TSP exchange by the broker. The latter is used to detect the presence of NAT in the path and select IPv6 over UDP-IPv4 encapsulation.

The TSP connection can be established between two nodes, where each node can control a tunnel endpoint.

The nodes involved in the framework are:

1. the TSP client
2. the client tunnel endpoint
3. the TSP server
4. the server tunnel endpoint

1,3, and 4 form the tunnel broker model [RFC3053], where 3 is the tunnel broker and 4 is the tunnel server (Figure 1). The tunnel

broker may control one or many tunnel servers.

In its simplest model, one node is the client configured as a tunnel endpoint (1 and 2 on the same node), and the second node is the server configured as the other tunnel endpoint (3 and 4 on the same node). This model is shown in Figure 2:

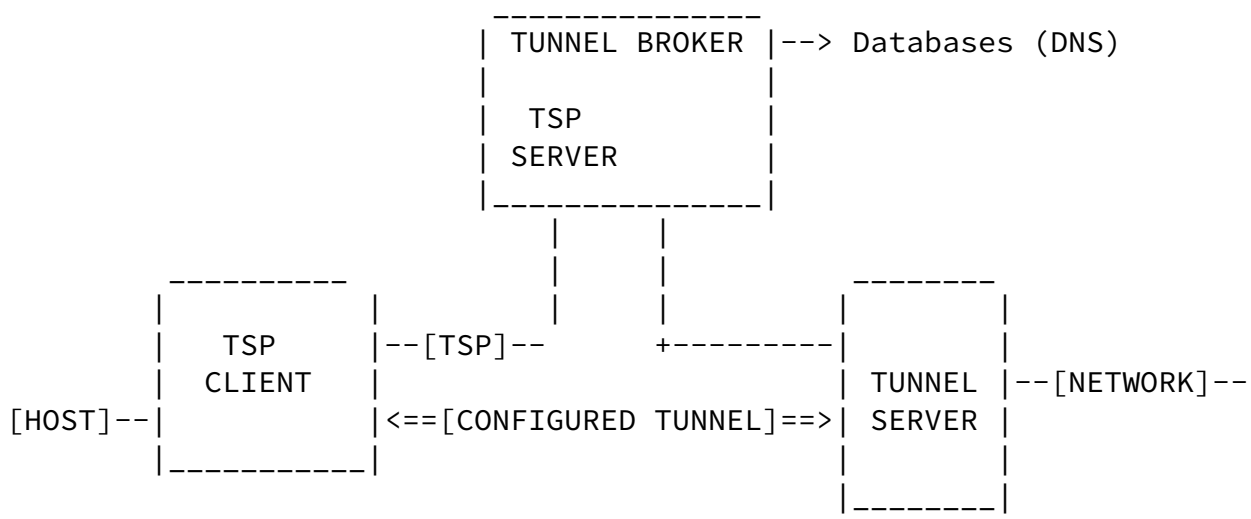


Figure 1: Tunnel Setup Protocol Used on Tunnel Broker Model





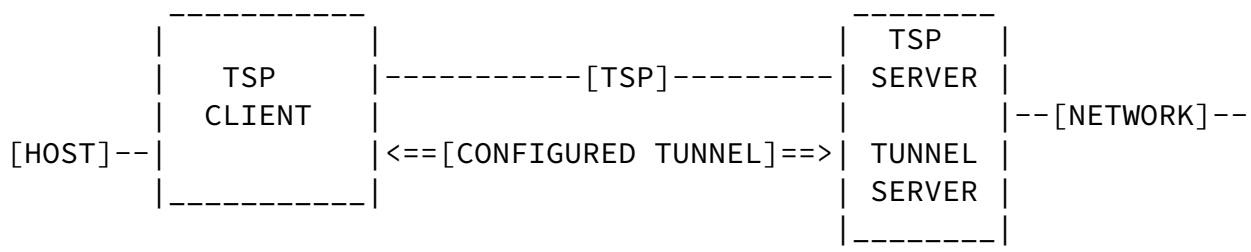


Figure 2: Tunnel Setup Protocol Used on Tunnel Server Model

From the point of view of an operating system, TSP is implemented as a client application that is able to configure network parameters of the operating system.

## 2.1. NAT Discovery

TSP is also used to discover if a NAT is in the path. In this discovery mode, the client sends a TSP message over UDP, containing its tunnel request information (such as its source IPv4 address) to the TSP server. The TSP server compares the IPv4 source address of the packet with the address in the TSP message. If they differ, one or many IPv4 NATs are in the path.

If an IPv4 NAT is discovered, then IPv6 over UDP-IPv4 tunnel encapsulation is selected. Once the TSP signaling is done, the tunnel is established over the same UDP channel used for TSP, so the same NAT address-port mapping is used for both the TSP session and the IPv6 traffic. If no IPv4 NAT is detected in the path by the TSP server, then IPv6 over IPv4 encapsulation is used.

A keep-alive mechanism is also included to keep the NAT mapping active.

The IPv4 NAT discovery builds the most effective tunnel for all cases, including in a dynamic situation where the client moves.

## 2.2. Any Encapsulation

TSP is used to negotiate IPv6 over IPv4 tunnels, IPv6 over UDP-IPv4 tunnels, and IPv4 over IPv6 tunnels. IPv4 over IPv6 tunnels is used in the Dual-Stack Transition Mechanism (DSTM) together with TSP [DSTM].

## 2.3. Mobility

When a node moves to a different IP network (i.e., change of its IPv4 address when doing IPv6 over IPv4 encapsulation), the TSP client reconnects automatically to the broker to re-establish the tunnel





(keep-alive mechanism). On the IPv6 layer, if the client uses user authentication, the same IPv6 address and prefix are kept and re-established, even if the IPv4 address or tunnel encapsulation type changes.

# 3. Advantages of TSP

- Tunnels established by TSP are static tunnels, which are more

secure than automated tunnels [RFC3964]; no third-party relay required.

- Stability of the IP address and prefix, enabling applications needing stable address to be deployed and used. For example, when tunneling IPv6, there is no dependency on the underlying IPv4 address.

- Prefix assignment supported. Can use provider address space.

- Signaling protocol flexible and extensible (XML, SASL)

- One solution to many encapsulation techniques: IPv6 in IPv4, IPv4 in IPv6, IPv6 over UDP over IPv4. Can be extended to other encapsulation types, such as IPv6 in IPv6.

- Discovery of IPv4 NAT in the path, establishing the most optimized tunneling technique depending on the discovery.

# 4. Protocol Description

## 4.1. Terminology

Tunnel Broker: In a tunnel broker model, the broker is taking charge of all communication between tunnel servers (TSs) and tunnel clients (TCs). Tunnel clients query brokers for a tunnel and the broker finds a suitable tunnel server, asks the tunnel server to set up the tunnel, and sends the tunnel information to the tunnel Client.

Tunnel Server: Tunnel servers are providing the specific tunnel service to a tunnel client. It can receive the tunnel request from a tunnel broker (as in the tunnel broker model) or directly from the tunnel client. The tunnel server is the tunnel endpoint.

Tunnel Client: The tunnel client is the entity that needs a tunnel for a particular service or connectivity. A tunnel client can be either a host or a router. The tunnel client is the other tunnel endpoint.





v6v4: IPv6-over-IPv4 tunnel encapsulation

v6udpv4: IPv6-over-UDP-over-IPv4 tunnel encapsulation

v4v6: IPv4-over-IPv6 tunnel encapsulation

## 4.2. Topology

The following diagrams describe typical TSP scenarios. The goal is to establish a tunnel between tunnel client and tunnel server.

## 4.3. Overview

The Tunnel Setup Protocol is initiated from a client node to a tunnel broker. The Tunnel Setup Protocol has three phases:

Authentication phase: The Authentication phase is when the tunnel broker/server advertises its capability to a tunnel client and when a tunnel client authenticate to the broker/server.

Command phase: The command phase is where the client requests or updates a tunnel.

Response phase: The response phase is where the tunnel client receives the request response from the tunnel broker/server, and the client accepts or rejects the tunnel offered.

For each command sent by a tunnel client, there is an expected response from the server.

After the response phase is completed, a tunnel is established as requested by the client. If requested, periodic keep-alive packets can be sent from the client to the server.

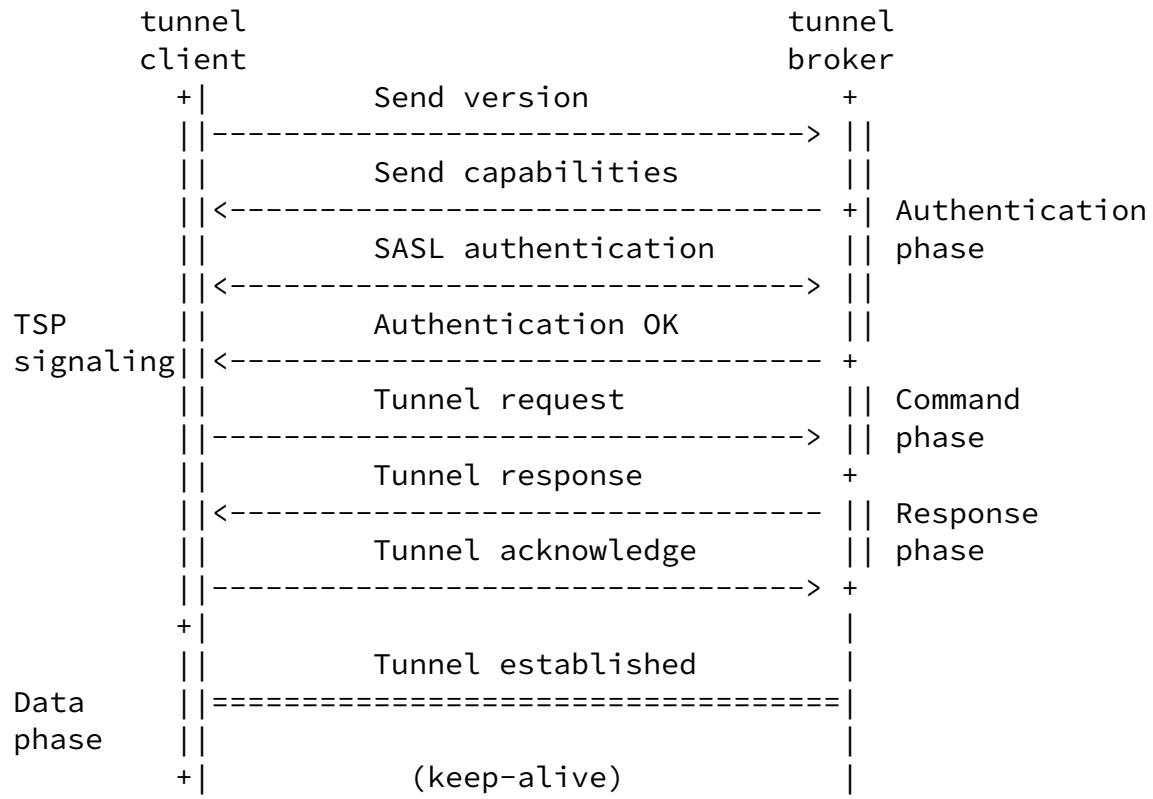


Figure 3: Tunnel Setup Protocol Exchange

## 4.4. TSP Signaling

The following sections describe in detail the TSP and the different phases in the TSP signaling.

### 4.4.1. Signaling Transport

TSP signaling can be transported over TCP or UDP, and over IPv4 or IPv6. The tunnel client selects the transport according to the tunnel encapsulation being requested. Figure 4 shows the transport used for TSP signaling with possible tunnel encapsulation requested.

TSP signaling over UDP/v4 MUST be used if a v6 over UDP over IPv4 (v6udpv4) tunnel is to be requested (e.g., for NAT traversal).

| Tunnel Encapsulation Requested | Valid Transport | Valid Address family |
|---|---|---|
| v6anyv4 | TCP UDP | IPv4 |
| v6v4 | TCP UDP | IPv4 |
| v6udpv4 | UDP | IPv4 |
| v4v6 | TCP UDP | IPv6 |

Figure 4: TSP Signaling Transport

Note that the TSP framework allows for other type of encapsulation to be defined, such as IPv6 over Generic Routing Encapsulation (GRE) or IPv6 over IPv6.

#### 4.4.1.1. TSP Signaling over TCP

TSP over TCP is sent over port number 3653 (IANA assigned). TSP data used during signaling is detailed in the next sections.

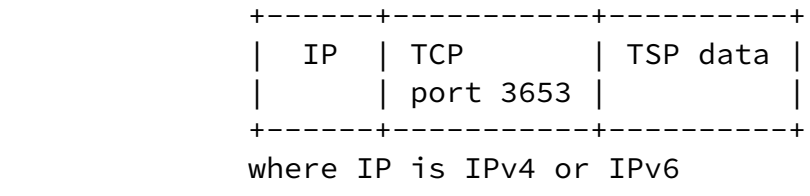


Figure 5: Tunnel Setup Protocol Packet Format (TCP)

#### 4.4.1.2. TSP Signaling over UDP/v4

While TCP provides the connection-oriented and reliable data delivery features required during the TSP signaling session, UDP does not offer any reliability. This reliability is added inside the TSP session as an extra header at the beginning of the UDP payload.

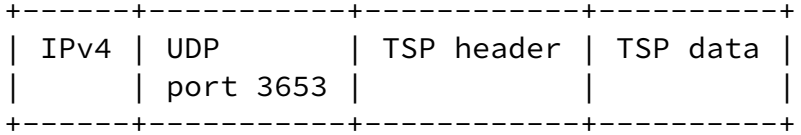

Figure 6: Tunnel Setup Protocol Packet Format (UDP)

The algorithm used to add reliability to TSP packets sent over UDP is described in Section 22.5 of [UNP].





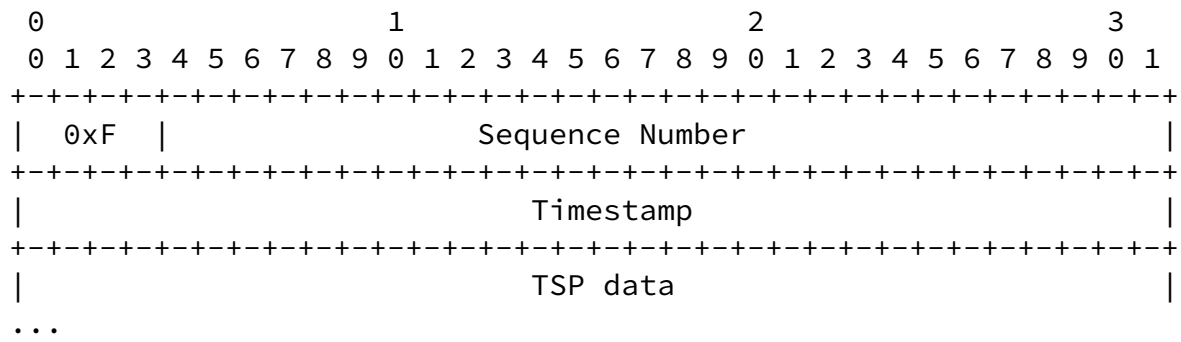


Figure 7: TSP Header for Reliable UDP

The 4-bit field (0-3) is set to 0xF. This marker is used by the tunnel broker to identify a TSP signaling packet that is sent after an IPv6 over UDP is established. This is explained in Section 4.5.2

Sequence Number: 28-bit field. Set by the tunnel client. Value is increased by one for every new packet sent to the tunnel broker. The return packet from the broker contains the unaltered sequence number.

Timestamp: 32-bit field. Set by the tunnel client. Generated from the client local-time value. The return packet from the broker contains the unaltered timestamp.

TSP data: Same as in the TCP/v4 case. Content described in later sections.

The TSP client builds its UDP packet as described above and sends it to the tunnel broker. When the tunnel broker responds, the same values for the sequence number and timestamp MUST be sent back to the

client. The TSP client can use the timestamp to determine the retransmission timeout (current time minus the packet timestamp). The client SHOULD retransmit the packet when the retransmission timeout is reached. The retransmitted packet MUST use the same sequence number as the original packet so that the server can detect duplicate packets. The client SHOULD use exponential backoff when retransmitting packets to avoid network congestion.

### 4.4.2. Authentication Phase

The authentication phase has 3 steps:

- Client's protocol version identification





- Server's capability advertisement
- Client authentication

When a TCP or UDP session is established to a tunnel broker, the tunnel client sends the current protocol version it is supporting. The version number syntax is:

```
VERSION=2.0.0 CR LF
```

Version 2.0.0 is the version number of this specification. Version 1.0.0 was defined in earlier documents.

If the server doesn't support the protocol version, it sends an error message and closes the session. The server can optionally send a server list that may support the protocol version of the client.

Example of an unsupported client version (without a server list):

```
-- Successful TCP Connection --
C:VERSION=0.1 CR LF
S:302 Unsupported client version CR LF
-- Connection closed --
```

Figure 8: Example of Unsupported Client Version

Example of a version not supported (with a server list):

```
-- Successful TCP Connection --
C:VERSION=1.1 CR LF
S:1302 Unsupported client version CR LF
  <tunnel action="list" type="broker">
     <broker>
        <address type="ipv4">1.2.3.4</address>
     </broker>
     <broker>
        <address type="dn">ts1.isp1.com</address>
     </broker>
  </tunnel>
-- Connection closed --
```

Figure 9: Example of Unsupported Client Version, with Server Redirection

If the server supports the version sent by the client, then the server sends a list of the capabilities supported for authentication and tunnels.





```
CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=ANONYMOUS AUTH=PLAIN
AUTH=DIGEST-MD5 CR LF
```

Tunnel types must be registered with IANA and their profiles are defined in Section 7. Authentication is done using SASL [RFC4422]. Each authentication mechanism should be a registered SASL mechanism. Description of such mechanisms is not in the scope of this document.

The tunnel client can then choose to close the session if none of the capabilities fit its needs. If the tunnel client chooses to continue, it authenticates to the server using one of the advertised mechanisms using SASL. If the authentication fails, the server sends an error message and closes the session.

The example in Figure 10 shows a failed authentication where the tunnel client requests an anonymous authentication that is not supported by the server.

Note that linebreaks and indentation within a "C:" or "S:" are editorial and not part of the protocol.

```
-- Successful TCP Connection --
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:300 Authentication failed CR LF
```

Figure 10: Example of Failed Authentication

Figure 11 shows a successful anonymous authentication.

```
-- Successful TCP Connection --
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=ANONYMOUS AUTH=PLAIN
  AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Success CR LF
```

Figure 11: Successful Anonymous Authentication





Digest-MD5 authentication with SASL follows [RFC2831]. Figure 12 shows a successful digest-MD5 SASL authentication.

```
-- Successful TCP Connection --
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=ANONYMOUS AUTH=PLAIN
  AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE DIGEST-MD5 CR LF
S:cmVhbG09aGV4b3Msbm9uY2U9MTExMzkwODk2OCxxb3A9YXV0aCxhbGdvcml0aG09bWQ
  1LXNlc3MsY2hhcnNldD11dGY4
C:Y2hhcnNldD11dGY4LHVzZXJuYW1lPSJ1c2VybmFtZTEiLHJlYWxtPSJoZXhvcyIsbm9
```

```
  uY2U9IjExMTM5MDg5NjgiLG5jPTAwMDAwMDAxLGNub25jZT0iMTExMzkyMzMxMSIsZG
  lnZXN0LXVyaT0idHNwL2hleG9zIixyZXNwb25zZT1mOGU0MmIzYzUwYzU5NzcxODUzZ
  jYyNzRmY2ZmZDFjYSxxb3A9YXV0aA==
S:cnNwYXV0aD03MGQ1Y2FiYzkyMzU1NjhiZTM4MGJhMmM5MDczODFmZQ==
S:200 Success CR LF
```

Figure 12: Successful Digest-MD5 Authentication

The base64-decoded version of the SASL exchange is:

```
S:realm="hexos",nonce="1113908968",qop="auth",algorithm=md5-sess,
  charset=utf8
C:charset=utf8,username="username1",realm="hexos",nonce="1113908968",
  nc=00000001,cnonce="1113923311",digest-uri="tsp/hexos",
  response=f8e42b3c50c59771853f6274fcffd1ca,qop=auth
S:rspauth=70d5cabc9235568be380ba2c907381fe
```

Once the authentication succeeds, the server sends a success return code and the protocol enters the Command phase.

### 4.4.3. Command and Response Phase

The Command phase is where the tunnel client sends a tunnel request or a tunnel update to the server. In this phase, commands are sent as XML messages. The first line is a "Content-length" directive that indicates the size of the following XML message. When the server sends a response, the first line is the "Content-length" directive, the second is the return code, and third one is the XML message, if any. The "Content-length" is calculated from the first character of the return code line to the last character of the XML message, inclusively.

Spaces can be inserted freely.





```
        -- UDP session established --
        C:VERSION=2.0.0 CR LF
        S:CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=ANONYMOUS
          AUTH=PLAIN AUTH=DIGEST-MD5 CR LF
```

```
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Success CR LF

C:Content-length: 205 CR LF
<tunnel action="create" type="v6udpv4">
 <client>
  <address type="ipv4">192.0.2.135</address>
<keepalive interval="30"></keepalive>
</client>
</tunnel> CR LF

S:Content-length: 501 CR LF
200 Success CR LF
<tunnel action="info" type="v6udpv4" lifetime="604800">
  <server>
    <address type="ipv4">192.0.2.115</address>
    <address type="ipv6">
    2001:db8:8000:0000:0000:0000:0000:38b2
    </address>
  </server>
  <client>
    <address type="ipv4">192.0.2.135</address>
    <address type="ipv6">
    2001:db8:8000:0000:0000:0000:0000:38b3
    </address>
    <keepalive interval="30">
      <address type="ipv6">
      2001:db8:8000:0000:0000:0000:0000:38b2
      </address>
    </keepalive>
  </client>
</tunnel> CR LF

C:Content-length: 35 CR LF
<tunnel action="accept"></tunnel> CR LF
```

Figure 13: Example of a Command/Response Sequence

The example in Figure 13 shows a client requesting an anonymous v6udpv4 tunnel, indicating that a keep-alive packet will be sent every 30 seconds. The tunnel broker responds with the tunnel

parameters and indicates its acceptance of the keep-alive period (Section 4.6). Finally, the client sends an accept message to the server.

Once the accept message has been sent, the server and client configure their tunnel endpoint based on the negotiated tunnel parameters.

## 4.5. Tunnel Establishment

### 4.5.1. IPv6-over-IPv4 Tunnels

Once the TSP signaling is complete, a tunnel can be established on the tunnel server and client node. If a v6v4 tunnel has been negotiated, then an IPv6-over-IPv4 tunnel [RFC4213] is established using the operating system tunneling interface. On the client node, this is accomplished by the TSP client calling the appropriate OS commands or system calls.

### 4.5.2. IPv6-over-UDP Tunnels

If a v6udpv4 tunnel is configured, the same source/destination address and port used during the TSP signaling are used to configure the v6udpv4 tunnel. If a NAT is in the path between the TSP client and the tunnel broker, the TSP signaling session will have created a UDP state in the NAT. By reusing the same UDP socket parameters to transport IPv6, the traffic will flow across the NAT using the same state.

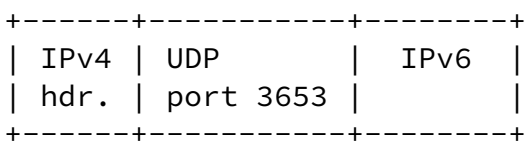


Figure 14: IPv6 Transport over UDP

At any time, a client may re-establish a TSP signaling session. The client disconnects the current tunnel and starts a new TSP signaling session as described in Section 4.4.1.2. If a NAT is present and the new TSP session uses the same UDP mapping in the NAT as for the tunnel, the tunnel broker will need to disconnect the client tunnel before the client can establish a new TSP session.

## 4.6. Tunnel Keep-Alive

A TSP client may select to send periodic keep-alive messages to the server in order to maintain its tunnel connectivity. This allows the

client to detect network changes and enable automatic tunnel





re-establishment. In the case of IPv6-over-UDP tunnels, periodic keep-alive messages can help refresh the connection state in a NAT if such a device is in the tunnel path.

For IPv6-over-IPv4 and IPv6-over-UDP tunnels, the keep-alive message is an ICMPv6 echo request [RFC4443] sent from the client to the tunnel server. The IPv6 destination address of the echo message MUST be the address from the 'keepalive' element sent in the tunnel response during the TSP signaling (Section 4.4.3). The echo message is sent over the configured tunnel.

The tunnel server responds to the ICMPv6 echo requests and can keep track of which tunnel is active. Any client traffic can also be used to verify if the tunnel is active. This can be used by the broker to disconnect tunnels that are no longer in use.

The broker can send a different keep-alive interval from the value specified in the client request. The client MUST conform to the broker-specified keep-alive interval. The client SHOULD apply a random "jitter" value to avoid synchronization of keep-alive messages from many clients to the server [FJ93]. This is achieved by using an interval value in the range of [0.75T - T], where T is the keep-alive interval specified by the server.

## 4.7. XML Messaging

This section describes the XML messaging used in the TSP signaling during the command and response phase. The XML elements and attributes are listed in the DTD (Appendix A).

### 4.7.1. Tunnel

The client and server use the tunnel token with an action attribute. Valid actions for this profile are: 'create', 'delete', 'info', 'accept', and 'reject'.

create: action used to request a new tunnel or update an existing tunnel. Sent by the tunnel client.

delete: action used to remove an existing tunnel from the server.

Sent by the tunnel client.

info: action used to request current properties of an existing tunnel. This action is also used by the tunnel broker to send tunnel parameters following a client 'create' action.





accept: action used by the client to acknowledge the server that the tunnel parameters are accepted. The client will establish a tunnel.

reject: action used by the client to signal the server that the tunnel parameters offered are rejected and no tunnel will be established.

The tunnel 'lifetime' attribute is set by the tunnel broker and specifies the lifetime of the tunnel in minutes. The lifetime is an administratively set value. When a tunnel lifetime has expired, it is disconnected on the tunnel server.

The 'tunnel' message contains three elements:

<client>: Client's information

<server>: Server's information

<broker>: List of other servers

### 4.7.2. Client Element

The 'client' element contains 3 sub-elements: 'address', 'router', and 'keepalive'. These elements are used to describe the client request and will be used by the server to create the appropriate tunnel. The client element is the only element sent by a client.

The 'address' element is used to identify the client IP endpoint of the tunnel. When tunneling over IPv4, the client MUST send only its IPv4 address to the server. When tunneling over IPv6, the client MUST only send its IPv6 address to the server.

The broker then returns the assigned IPv6 or IPv4 address endpoint and domain name inside the 'client' element when the tunnel is created or updated. If supported by the broker, the 'client' element MAY contain the registered DNS name for the address endpoint assigned to the client.

Optionally, a client MAY send a 'router' element to ask for a prefix delegation.

Optionally, a client MAY send a 'keepalive' element that contains the keep-alive time interval requested by the client.





#### 4.7.3. Server Element

The 'server' element contains two elements: 'address' and 'router'. These elements are used to describe the server's tunnel endpoint. The 'address' element is used to provide both IPv4 and IPv6 addresses of the server's tunnel endpoint, while the 'router' element provides information for the routing method chosen by the client.

#### 4.7.4. Broker Element

The 'broker' element is used by a tunnel broker to provide an alternate list of brokers to a client in the case where the server is not able to provide the requested tunnel.

The 'broker' element contains an 'address' element or a series of 'address' elements.

## 5. Tunnel Request Examples

This section presents multiple examples of requests.

### 5.1. Host Tunnel Request and Reply

A simple tunnel request consist of a 'tunnel' element that contains only an 'address' element. The tunnel action is 'create', specifying a 'v6v4' tunnel encapsulation type. The response sent by the tunnel

broker is an 'info' action. Note that the registered Fully-Qualified Domain Name (FQDN) of the assigned client IPv6 address is also returned to the tunnel client.





```
-- Successful TCP Connection --
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=ANONYMOUS CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Authentication successful CR LF
C:Content-length: 123 CR LF
  <tunnel action="create" type="v6v4">
     <client>
         <address type="ipv4">1.1.1.1</address>
     </client>
  </tunnel> CR LF
S: Content-length: 234 CR LF
   200 OK CR LF
   <tunnel action="info" type="v6v4" lifetime="1440">
     <server>
        <address type="ipv4">192.0.2.114</address>
        <address type="ipv6">
        2001:db8:c18:ffff:0000:0000:0000:0000
        </address>
```

```
            </server>
            <client>
               <address type="ipv4">1.1.1.1</address>
               <address type="ipv6">
               2001:db8:c18:ffff::0000:0000:0000:0001
               </address>
               <address type="dn">userid.domain</address>
            </client>
         </tunnel> CR LF
      C: Content-length: 35 CR LF
         <tunnel action="accept"></tunnel> CR LF
```

Figure 15: Simple Tunnel Request Made by a Client

## 5.2. Router Tunnel Request with a /48 Prefix Delegation and Reply

A tunnel request with a prefix consists of a 'tunnel' element that contains an 'address' element and a 'router' element. The 'router' element also contains the 'dns_server' element that is used to request a DNS delegation of the assigned IPv6 prefix. The 'dns_server' element lists the IP address of the DNS servers to be registered for the reverse-mapping zone.





Tunnel request with prefix and static routes.

```
C: Content-length: 234 CR LF
   <tunnel action="create" type="v6v4">
    <client>
     <address type="ipv4">192.0.2.9</address>
     <router>
      <prefix length="48"/>
      <dns_server>
       <address type="ipv4">192.0.2.5</address>
       <address type="ipv4">192.0.2.4</address>
       <address type="ipv6">2001:db8::1</address>
```

```
        </dns_server>
       </router>
      </client>
     </tunnel> CR LF
S: Content-length: 234 CR LF
     200 OK CR LF
     <tunnel action="info" type="v6v4" lifetime="1440">
      <server>
       <address type="ipv4">192.0.2.114</address>
       <address type="ipv6">
       2001:db8:c18:ffff:0000:0000:0000:0000
       </address>
      </server>
      <client>
       <address type="ipv4">192.0.2.9</address>
       <address type="ipv6">
       2001:db8:c18:ffff::0000:0000:0000:0001
       </address>
       <address type="dn">userid.domain</address>
       <router>
        <prefix length="48">2001:db8:c18:1234::</prefix>
        <dns_server>
         <address type="ipv4">192.0.2.5</address>
         <address type="ipv4">192.0.2.4</address>
         <address type="ipv6">2001:db8::1</address>
        </dns_server>
       </router>
      </client>
     </tunnel> CR LF
C: Content-length: 35 CR LF
     <tunnel action="accept"></tunnel> CR LF
```

Figure 16: Tunnel Request with Prefix and DNS Delegation





## 5.3. IPv4 over IPv6 Tunnel Request

This is similar to the previous 'create' action, but with the tunnel type is set to 'v4v6'.

```
-- Successful TCP Connection --
C:VERSION=1.0 CR LF
S:CAPABILITY TUNNEL=V4V6 AUTH=DIGEST-MD5 AUTH=ANONYMOUS
  CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:OK Authentication successful CR LF
C:Content-length: 228 CR LF
  <tunnel action="create" type="v4v6">
     <client>
         <address type="ipv6">
         2001:db8:0c18:ffff:0000:0000:0000:0001
         </address>
     </client>
  </tunnel> CR LF
```

Figure 17: Simple Tunnel Request Made by a Client

If the allocation request is accepted, the broker will acknowledge the allocation to the client by sending a 'tunnel' element with the attribute 'action' set to 'info', 'type' set to 'v4v6' and the 'lifetime' attribute set to the period of validity or lease time of the allocation. The 'tunnel' element contains 'server' and 'client' elements.

```
S: Content-length: 370 CR LF
   200 OK CR LF
   <tunnel action="info" type="v4v6" lifetime="1440">
     <server>
        <address type="ipv4" length="30">
        192.0.2.2
        </address>
        <address type="ipv6">
        2001:db8:c18:ffff:0000:0000:0000:0002
        </address>
     </server>
     <client>
        <address type="ipv4" length="30">
        192.0.2.1
        </address>
        <address type="ipv6">
        2001:db8:c18:ffff::0000:0000:0000:0001
        </address>
     </client>
   </tunnel> CR LF
```

Figure 18: IPv4 over IPv6 Tunnel Response

In DSTM [DSTM] terminology, the DSTM server is the TSP broker and the Tunnel Endpoint (TEP) is the tunnel server.

## 5.4. NAT Traversal Tunnel Request

When a client is capable of both IPv6 over IPv4 and IPv6 over UDP over IPv4 encapsulation, it can request the broker, by using the "v6anyv4" tunnel mode, to determine if it is behind a NAT and to send the appropriate tunnel encapsulation mode as part of the response. The client can also explicitly request an IPv6 over UDP over IPv4 tunnel by specifying "v6udpv4" in its request.

In the following example, the client informs the broker that it requests to send keep-alives every 30 seconds. In its response, the broker accepted the client-suggested keep-alive interval, and the IPv6 destination address for the keep-alive packets is specified.

```
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 TUNNEL=V6UDPV4 AUTH=DIGEST-MD5 CR LF
C:AUTHENTICATE ... CR LF
S:200 Authentication successful CR LF
C:Content-length: ... CR LF
  <tunnel action="create" type="v6anyv4">
     <client>
         <address type="ipv4">10.1.1.1</address>
         <keepalive interval="30"></keepalive>
     </client>
  </tunnel> CR LF
S: Content-length: ... CR LF
   200 OK CR LF
   <tunnel action="info" type="v6udpv4" lifetime="1440">
     <server>
        <address type="ipv4">192.0.2.114</address>
        <address type="ipv6">
        2001:db8:c18:ffff:0000:0000:0000:0002
        </address>
     </server>
     <client>
        <address type="ipv4">10.1.1.1</address>
        <address type="ipv6">
        2001:db8:c18:ffff::0000:0000:0000:0003
        </address>
        <keepalive interval="30">
           <address type="ipv6">
           2001:db8:c18:ffff:0000:0000:0000:0002
           </address>
        </keepalive>
     </client>
   </tunnel> CR LF
```

Figure 19: Tunnel Request Using v6anyv4 Mode

## 6. Applicability of TSP in Different Networks

This section describes the applicability of TSP in different networks.

### 6.1. Provider Networks with Enterprise Customers

In a provider network where IPv4 is dominant, a tunneled infrastructure can be used to provide IPv6 services to the enterprise customers, before a full IPv6 native infrastructure is built. In order to start deploying in a controlled manner and to give enterprise customers a prefix, the TSP framework is used. The TSP server can be in the core, in the aggregation points or in the Points





of Presence (PoPs) to offer the service to the customers. IPv6 over IPv4 encapsulation can be used. If the customers are behind an IPv4 NAT, then IPv6 over UDP-IPv4 encapsulation can be used. TSP can be used in combination with other techniques.

## 6.2. Provider Networks with Home/Small Office Customers

In a provider network where IPv4 is dominant, a tunneled infrastructure can be used to provide IPv6 services to the home/small office customers, before a full IPv6 native infrastructure is built. The small networks such as Home/Small offices have a non-upgradable gateway with NAT. TSP with NAT traversal is used to offer IPv6 connectivity and a prefix to the internal network.

Automation of the prefix assignment and DNS delegation, done by TSP, is a very important feature for a provider in order to substantially decrease support costs. The provider can use the same Authentication, Authorization, and Accounting (AAA) database that is used to authenticate the IPv4 broadband users. Customers can deploy home IPv6 networks without any intervention of the provider support people.

With the NAT discovery function of TSP, providers can use the same TSP infrastructure for both NAT and non-NAT parts of the network.

## 6.3. Enterprise Networks

In an enterprise network where IPv4 is dominant, a tunneled infrastructure can be used to provide IPv6 services to the IPv6 islands (hosts or networks) inside the enterprise, before a full IPv6 native infrastructure is built [RFC4057]. TSP can be used to give IPv6 connectivity, prefix, and routing for the islands. This gives the enterprise a fully controlled deployment of IPv6 while maintaining automation and permanence of the IPv6 assignments to the islands.

## 6.4. Wireless Networks

In a wireless network where IPv4 is dominant, hosts and networks move and change IPv4 address. TSP enables the automatic re-establishment of the tunnel when the IPv4 address changes.

In a wireless network where IPv6 is dominant, hosts and networks move. TSP enables the automatic re-establishment of the IPv4 over IPv6 tunnel.





## 6.5. Unmanaged Networks

An unmanaged network is where no network manager or staff is available to configure network devices [RFC3904]. TSP is particularly useful in this context where automation of all necessary information for the IPv6 connectivity is handled by TSP: tunnel endpoint parameters, prefix assignment, DNS delegation, and routing.

An unmanaged network may (or may not) be behind a NAT. With the NAT discovery function, TSP works automatically in both cases.

## 6.6. Mobile Hosts and Mobile Networks

Mobile hosts are common and used. Laptops moving from wireless, wired in an office, home, etc., are examples. They often have IPv4 connectivity, but not necessarily IPv6. The TSP framework enables the mobile hosts to have IPv6 connectivity wherever they are, by having the TSP client send updated information of the new environment to the TSP server, when a change occurs. Together with NAT discovery and traversal, the mobile host can always be IPv6 connected wherever it is.

Mobile here means only the change of IPv4 address. Mobile-IP mechanisms and fast hand-off take care of additional constraints in mobile environments.

Mobile networks share the applicability of the mobile hosts. Moreover, in the TSP framework, they also keep their prefix

assignment and can control the routing. NAT discovery can also be used.

## 7. IANA Considerations

A tunnel type registry has been created by IANA. The following strings are defined in this document:

- "v6v4" for IPv6 in IPv4 encapsulation (using IPv4 protocol 41)
- "v6udpv4" for IPv6 in UDP in IPv4 encapsulation
- "v6anyv4" for IPv6 in IPv4 or IPv6 in UDP in IPv4 encapsulation
- "v4v6" for IPv4 in IPv6 encapsulation

Registration of a new tunnel type can be obtained on a first come, first served policy [RFC5226]. A new registration should provide a point of contact, the tunnel type string, and a brief description on the applicability.





IANA assigned 3653 as the TSP port number.

## 8. Security Considerations

Authentication of the TSP session uses the SASL [RFC4422] framework, where the authentication mechanism is negotiated between the client and the server. The framework uses the level of authentication needed for securing the session, based on the policies.

Static tunnels are created when the TSP negotiation is terminated. Static tunnels are not open gateways and exhibit less security issues than automated tunnels. Static IPv6 in IPv4 tunnel security considerations are described in [RFC4213].

In order to help ensure that the traffic is traceable to its correct source network, a tunnel server implementation should allow ingress filtering on the user tunnel [RFC3704].

A customer A behind a NAT can use a large number of (private) IPv4 addresses and/or source ports and request multiple v6udpv4 tunnels. That would quickly saturate the tunnel server capacity. The tunnel

broker implementation should offer a way to throttle and limit the number of tunnel established to the same IPv4 address.

## 9. Conclusion

The Tunnel Setup Protocol (TSP) is applicable in many environments, such as: providers, enterprises, wireless, unmanaged networks, mobile hosts, and networks. TSP gives the two tunnel endpoints the ability to negotiate tunnel parameters, as well as prefix assignment, DNS delegation and routing in an authenticated session. It also provides an IPv4 NAT discovery function by using the most effective encapsulation. It also supports the IPv4 mobility of the nodes.

## 10. Acknowledgements

This document is the merge of many previous documents about TSP. Octavio Medina has contributed to an earlier document (IPv4 in IPv6). Thanks to the following people for comments on improving and clarifying this document: Pekka Savola, Alan Ford, Jeroen Massar, and Jean-Francois Tremblay.





## 11. References

### 11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831,

May 2000.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[W3C.REC-xml-2004] Yergeau, F., Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, February 2004.

## 11.2. Informative References

[DSTM] Bound, J., Toutain, L., and JL. Richier, "Dual Stack IPv6 Dominant Transition Mechanism", Work in Progress, October 2005.

[FJ93] Floyd, S. and V. Jacobson, "The Synchronization of Periodic Routing Messages", Proceedings of ACM SIGCOMM, September 1993.

[RFC3053] Durand, A., Fasano, P., Guardini, I., and D. Lento, "IPv6 Tunnel Broker", RFC 3053, January 2001.





[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

[RFC3904] Huitema, C., Austein, R., Satapati, S., and R. van der Pol, "Evaluation of IPv6 Transition Mechanisms for Unmanaged Networks", RFC 3904,

September 2004.

[RFC3964] Savola, P. and C. Patel, "Security Considerations for 6to4", RFC 3964, December 2004.

[RFC4057] Bound, J., "IPv6 Enterprise Network Scenarios", RFC 4057, June 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[UNP] Stevens, R., Fenner, B., and A. Rudoff, "Unix Network Programming, 3rd edition", Addison Wesley ISBN 0-13-141155-1, 2004.

## Appendix A. The TSP DTD

```
<?xml version="1.0"?>
<!DOCTYPE tunnel  [
<!ELEMENT tunnel (server?,client?,broker?)>
  <!ATTLIST tunnel action
              (create|delete|info|accept|reject) #REQUIRED >
  <!ATTLIST tunnel type
              (v6v4|v4v6|v6anyv4|v6udpv4) #REQUIRED >
  <!ATTLIST tunnel lifetime CDATA "1440"    >

<!ELEMENT server        (address+,router?)>

<!ELEMENT client        (address+,router?)>

<!ELEMENT broker        (address+)>

<!ELEMENT router        (prefix?,dns_server?)>

<!ELEMENT dns_server    (address+)>

<!ELEMENT prefix        (#PCDATA)>
  <!ATTLIST prefix length CDATA #REQUIRED>

<!ELEMENT address       (#PCDATA)>
  <!ATTLIST address type (ipv4|ipv6|dn) #REQUIRED>
  <!ATTLIST address length CDATA "">

<!ELEMENT keepalive (address?)>
  <!ATTLIST keepalive interval CDATA #REQUIRED>
]>
```

Figure 20: TSP DTD

## Appendix B. Error Codes

Error codes are sent as a numeric value followed by a text message describing the code, similar to SMTP. The codes are sent from the broker to the client. The currently defined error codes are shown below. Upon receiving an error, the client will display the appropriate message to the user.

New error messages may be defined in the future. For interoperability purpose, the error code range to use should be from 300 to 599.

The reply code 200 is used to inform the client that an action successfully completed. For example, this reply code is used in response to an authentication request and a tunnel creation request.

The server may redirect the client to another broker. The details on how these brokers are known or discovered is beyond the scope of this document. When a list of tunnel brokers follows the error code as a referral service, then 1000 is added to the error code.

The predefined values are:

200 Success: Successful operation.

300 Authentication failed: Invalid userid, password, or authentication mechanism.

301 No more tunnels available: The server has reached its capacity limit.

302 Unsupported client version: The client version is not supported by the server.

303 Unsupported tunnel type: The server does not provide the requested tunnel type.

310 Server side error: Undefined server error.

500 Invalid request format or specified length: The received request has invalid syntax or is truncated.

501 Invalid IPv4 address: The IPv4 address specified by the client

is invalid.

502 Invalid IPv6 address: The IPv6 address specified by the client is invalid.





506 IPv4 address already used for existing tunnel: An IPv6-over-IPv4 tunnel already exists using the same IPv4 address endpoints.

507 Requested prefix length cannot be assigned: The requested prefix length cannot be allocated on the server.

521 Request already in progress: The client tunnel request is being processed by the server. Temporary error.

530 Server too busy: Request cannot be processed, insufficient resources. Temporary error.

## Authors' Addresses

Marc Blanchet
Viagenie
2600 boul. Laurier, suite 625
Quebec, QC G1V 4W1
Canada

Phone: +1-418-656-9254
EMail: Marc.Blanchet@viagenie.ca

Florent Parent
Beon Solutions
Quebec, QC
Canada

Phone: +1 418 265 7357
EMail: Florent.Parent@beon.ca