MIGRATION BROKER™
TSP™ CLIENT GUIDE

Migration Broker™ TSP™ Client Guide

Part number HEX-DC-0005-00.

Table of Contents

About this guide

This document describes how to configure and use the TSP client version 2.0.

Migration Broker documents

This table lists the documents for the Migration Broker.

Title	Content
Migration Broker Documentation Guide	Lists the documents for the Migration Broker, introduces the HexOS software, and describes CLI command modes and basic features.
Migration Broker Quick Setup Guide	Provides hardware installation procedures and minimal software configuration procedures.
Migration Broker HexOS Configuration Guide	Describes how to configure the Migration Broker using the CLI.
Migration Broker HexOS Command Reference	Describes the HexOS commands, in four sections: • Management and protocol-independent commands • Interface and access-list commands • Tunnel Broker commands • Logging and troubleshooting commands
Migration Broker TSP Client Guide	Describes how to configure and use the TSP client.

Obtaining documentation

The documents for the Migration Broker are supplied as Portable Document Format (PDF) files on the Migration Broker Software & Documentation CD-ROM. Printed copies of the documents are also available.

The Software & Documentation CD-ROM also provides HexOS image files, Tunnel Setup Protocol (TSP) Client software, the latest updates for the documentation, and HexOS software and copyright information. These items are also available on the Hexago corporate Web site.

Revision

This document is the revision 1.0 of the *Migration Broker TSP Client Guide* and describes the configuration and use of the TSP client version 2.0.

Introduction

This section gives an overview of the TSP client used with the Migration Broker.

TSP is a control protocol to establish and maintain static tunnels. The TSP client is used on the host computer to connect to the Migration Broker using the TSP protocol and get the information for its tunnel. When it receives the information for the tunnel, the TSP client creates the static tunnel on its operating system.

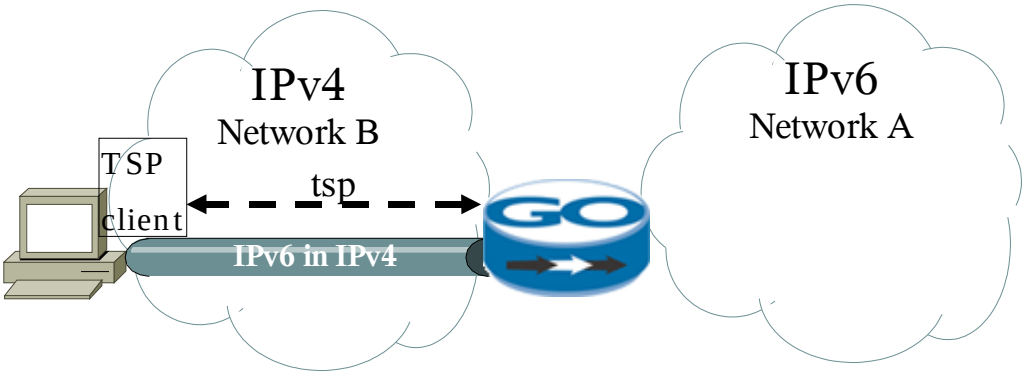

Figure 1 TSP client and Migration Broker

The TSP client code is mostly identical for all client platforms. However, creating the static tunnel is operating system dependent and is done by a script called by the TSP client.

Client components

Figure 2 shows the different components of the TSP client and their interactions with the node operating system and with the Migration Broker.

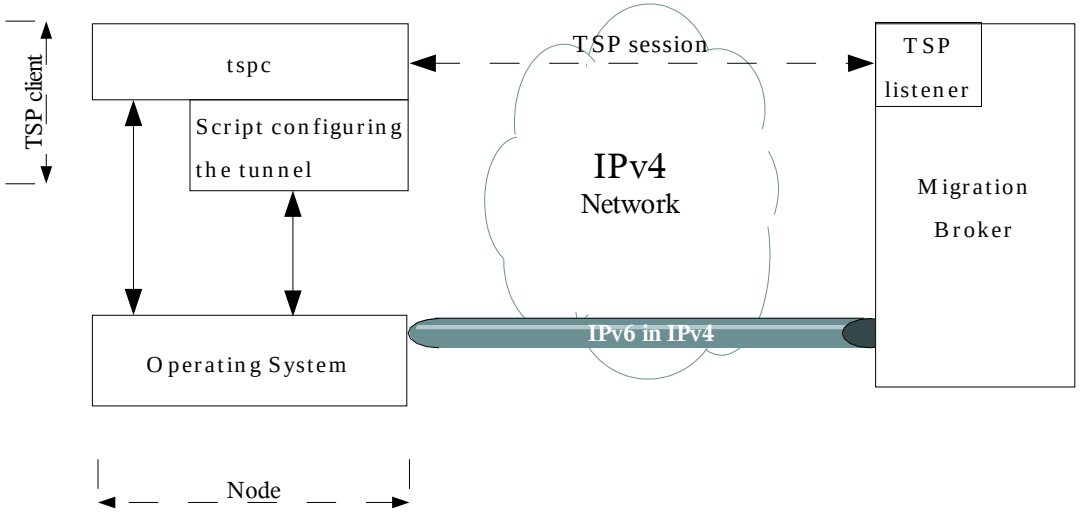

Figure 2 TSP client components

The script executed by the TSP client to configure the tunnel interface is customized for each type of supported operating system and takes care of all specifics for the target operating system. On Unix systems, it is a shell script. On Microsoft Windows, it is a batch file(.bat). This separation of the binary and script enables fast and easy additions of new operating systems, as has been shown by the community contributions for many operating systems.

Packaging

The TSP client is available either as part of the operating system distributions, such as Linux or FreeBSD; as downloadable software from the Web site of the tunnel broker service, such as Freenet6 (http://www.freenet6.net); as included in the Migration Broker CD-ROM; or directly from Hexago (http://www.hexago.com).

Configuring the TSP client

The TSP client is configured using a file called **tspc.conf**.

Configuration file `tspc.conf`

The configuration file is a text file with comments identified by the "#" character. Each statement has the format variable = value, as in rc.conf files in FreeBSD. The file statements are listed in Table 1. Many are optional.

Table 1 Statements in tspc.conf

Variable	Default value	Possible values	Description
`tsp_version`	`2.0.0`	`1.0.0` `2.0.0`	The highest version of the TSP protocol the TSP client supports. The tunnel broker adapts to the protocol version supported by the TSP client. The v6udpv4 and v6anyv4 tunnel encapsulation modes as well as the keepalive mechanism are available in version 2 and higher. The v2 TSP client also adapts to v1.0 brokers. This value must not be changed by the user.
`tsp_dir`	Current working directory	string	The TSP client directory which contains the `template` subdirectory for the shell scripts.

`auth_method`	`any`	`any`
		`digest-md5`
		`plain`
		`anonymous`
			The authentication used for the TSP session. **Digest-md5** is the most secure, where the passwords are not sent. **Plain** sends the userid and password. **Anonymous** sends no userid or password. With **any**, the TSP client uses the most secure mode based on its capabilities and the broker authentication capabilities. **Any** is recommended.

`client_v4`	`auto`	`auto` *IPv4 address*	The IPv4 address used by the TSP client as its tunnel endpoint source address. When **auto** is used, the TSP client uses the first IPv4 address given by the operating system.
`if_source`		*string*	The interface name on the operating system of the TSP client where the IPv4 address is taken for the IPv4 tunnel endpoint source address.
`userid`	`anonymous`	`anonymous` *string*	The user identification string.
`passwd`		*string*	The password for the userid.

`template`		`checktunnel`
		`cisco`
		`darwin`
		`freebsd`
		`linux`
		`netbsd`
		`openbsd`
		`solaris`
		`windows`
			The script file used to create the tunnel. The value is the name of the script file (in the template directory) which will be called by the TSP client at the end of the TSP session, to create the tunnel. When compiling the TSP client, the template variable is filled in with the correct value for the operating system the client was compiled on.

Variable	Default value	Possible values	Description
`server`	`broker.freenet6.net`	*ip_address* *hostname* *ip_address:port* *hostname:port*	The IP address or hostname (full domain name when appropriate) of the broker. A TSP port number can be specified.
`retry_delay`	0	number	When a TSP connection fails, the number of seconds to wait before retrying to connect to the tunnel broker.

`tunnel_mode`	`v6anyv4`	`v6v4`
		`v6udpv4`
		`v6anyv4`
		`v4v6`
			The tunnel encapsulation mode. See details in section “Tunnel encapsulation mode” .

`if_tunnel_v6v4`		string	The tunnel interface on the operating system of the client used for IPv6 in IPv4 encapsulation.
`if_tunnel_v4udpv4`		string	The tunnel interface on the operating system of the TSP client used for IPv6 in UDP in IPv4 encapsulation.
`proxy_client`	`no`	`yes` `no`	When set to **yes**, the TSP client is not the tunnel endpoint, but is a TSP proxy for the tunnel endpoint. The TSP client is going to remotely configure the tunnel endpoint on the client side.
`keepalive`	`yes`	`yes` `no`	When set to **yes**, the TSP client sends keepalives to keep the tunnel active. This is especially useful to have the Network Address Translation (NAT) keeps its mapping to have a sustainable tunnel over UDP.
`keepalive_interval`	`30`	number	This interval in seconds should be smaller than the NAT mapping timeout for UDP. The TSP client sends a keepalive to the broker every keepalive_interval . The broker may force a higher value than the TSP client wants, given a load expected on the broker from a provider.
`syslog_facility`		string	For a TSP client that supports syslog, specifies the syslog facility.
`syslog_level`		string	For a TSP client that supports syslog, specifies the syslog level.
`host_type`	`host`	`host` `router`	Specifies if the TSP client is a **host** or a **router**. In router **mode**, the TSP client receives a prefix if prefixlen is set.

`prefixlen`	`0`	`0`
		`48`
		`64`
			The length of the prefix required by the TSP client.

`if_prefix`		string	The interface in the operating system of the TSP client used to send router advertisements with the prefix received from the broker.
`dns_server`		string	The fully qualified domain name of the DNS server for the reverse DNS delegation of the prefix.

The template variable contains the name of the script file executed at the end of the TSP session. On

Unix, the **.sh** extension is added to the file name before the script is executed and on Microsoft Windows, the **.bat** extension is added. Customization of the script is discussed at the end of this document.

Tunnel encapsulation modes

Tunnel encapsulation modes, defined in the **tunnel_mode** variable are listed in Table 2. When **v6anyv4** is sent by the TSP client, the broker tests to see if the client is behind a NAT and responds by the correct encapsulation mode.

Table 2 Tunnel encapsulation mode keywords in tspc.conf

Tunnel mode keyword	Description
v6v4	IPv6 in IPv4 encapsulation, using IPv4 protocol 41. Does not work through NAT.
v6udpv4	IPv6 in UDP in IPv4 encapsulation. Works through a NAT.
v6anyv4	IPv6 in any IPv4 encapsulation. The tunnel broker will suggest the correct encapsulation method to the TSP client based on whether or not the broker discovers a NAT is in the path. If the broker finds a NAT, then **v6udpv4** is proposed to the TSP client, otherwise, **v6v4** is proposed.
v4v6	IPv4 in IPv6 encapsulation.

Not all TSP client platforms support the NAT traversal feature, implemented with the **v6udpv4** encapsulation mode. Please refer to the Release Notes to see if a specific platform supports NAT traversal.

Executing the TSP client

The TSP client is executed by typing the command **tspc** which stands for TSP client program. When an IPv6 in UDP in IPv4 tunnel is negotiated, the TSP client program forks itself and runs in the background to carry the keepalive mechanism with the broker. When an IPv6 in IPv4 tunnel is negotiated, the TSP client program exits after setting up the tunnel.

Arguments

The **tspc** program has some arguments to the command line, as described in Table 1.

Table 1 Arguments to the tspc client program

Argument	Description
-v **-vv** **-vvv**	Sets the verbose level and type of debugging information sent to the screen. **-vvv** gives the most debugging information, such as the TSP XML content.
-i *interface_name*	Sets the interface name for IPv6 in IPv4 encapsulation.
-u *interface_name*	Sets the interface name for IPv6 in UDP IPv4 encapsulation.
-s *interface_name*	Sets the interface name to configure router advertisements of the prefix when the client is a router and has received a prefix.
-f *config_filename*	Sets the configuration file.
-r *number_of_seconds*	Sets the retry interval when the TSP connection to the broker is not successful.
-h	Shows the version and list of options.

Troubleshooting

To troubleshoot the TSP client, use the **-v** or **-vv** or **-vvv** as command line argument. The **`tspc.log`** file contains logging information of the TSP session and the tunnel configuration.

Scenarios

This section describes typical scenarios for the TSP client.

Scenario 1: Single host, temporary IPv6 address

A single node is attached to the IPv4 Internet and needs a temporary IPv6 address and connectivity as shown in Figure 1. This is the state of the default configuration file when the TSP client software is installed. It uses the anonymous authentication mode which requires no prior registration of a username. Required variables for this scenario are the following:

```
auth_method=anonymous
userid=anonymous
host_type=host
```

Scenario 2: Single host, permanent IPv6 address

To get a permanent IPv6 address, the IPv6 address is bound to a username. The user must subscribe to and get a userid/password from the tunnel broker. Then the userid and password must be placed in the configuration file to be used for authentication of the TSP session with the tunnel broker. Required variables for this scenario are the following:

```
auth_method=any
userid=your_username
passwd=your_password
host_type=host
```

Scenario 3: Router, delegated IPv6 prefix

A TSP client router(R1) is forwarding IPV6 packets between the tunnel interface to the tunnel broker and another interface, specified as **if_prefix** in the configuration file. The TSP client also requested an IPv6 prefix from the tunnel broker to be advertised on its attached network. Figure 3 shows an example of this scenario where R1 is a TSP client which is authenticated by the tunnel broker and where the TSP client requested and received an IPv6 /64 prefix for its attached network.

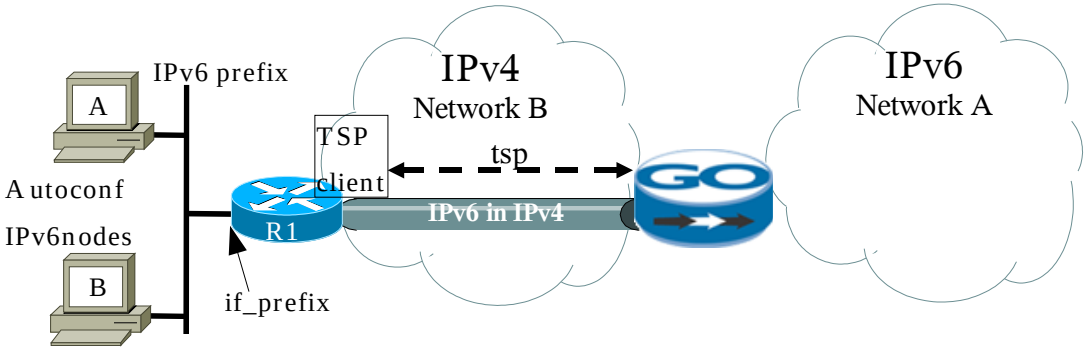

Figure 3 TSP client as router

The tunnel is established and R1 advertises the /64 received prefix on its attached network. Nodes A and B autoconfigure themselves based on the advertised prefix. A userid and password must be used in this configuration. Even if R1 changes its IPv4 address, the IPv6 prefix for nodes A and B remains permanent and stable. Required variables for this scenario are the following:

```
auth_method=any
userid=your_username
passwd=your_password
host_type=router
if_prefix=interface_name
prefixlen=64
```

Scenario 4: Behind an IPv4 Network Address Translator (NAT)

The TSP client, which is either a host or router as described above, can be behind a NAT. When this is the case, IPv6 in UDP in IPv4 encapsulation must be used to traverse the NAT. Figure 4 shows an example of this type of scenario.

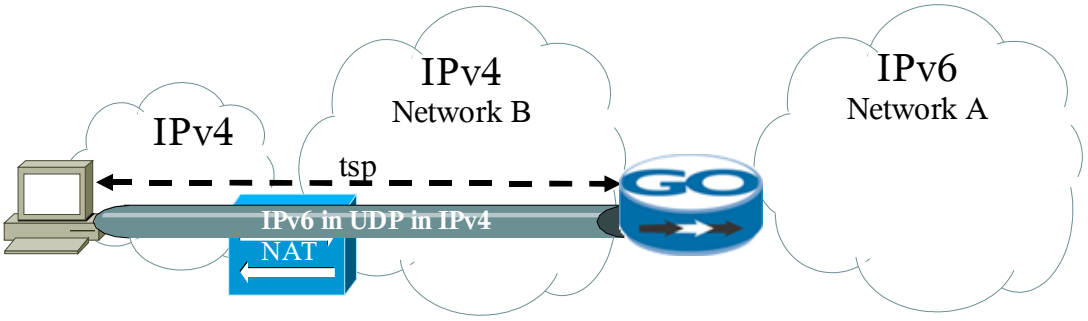

Figure 4 TSP Client behind an IPv4 NAT

Note that a TSP client cannot easily know whether nodes are behind a NAT or not. For example, a node can be using a public address even though it is behind a NAT. A node can be using a private address space, but does not traverse a NAT to reach its tunnel broker. For these reasons, the **v6anyv4** is used as the tunnel mode by the TSP client. The TSP client sends the tunnel request to the broker and, because the broker can verify if the TSP client is behind a NAT or not, the broker decides which encapsulation is appropriate for the tunnel requested by the client. This configuration is the recommended one to cover all possible cases, especially for mobile nodes.

Required variables for this scenario are the following:

```
auth_method=any
userid=your_username
passwd=your_password
tunnel_mode=v6anyv4
keepalive=yes
keepalive_interval=30
if_tunnel_v6udpv4=interface_name
```

The keepalive interval is used to keep the NAT mapping up. If the tunnel remains up for less the the keepalive interval, it might be because the NAT mapping lifetime is shorter than the keepalive interval. Decrease the keepalive interval. Observations in the field tells that NAT mapping varies from hours to few seconds.

Scenario 5: Mobile Node

A mobile node connects to the IPv4 Internet either with or without a NAT. The above configuration with **tunnel_mode=v6anyv4** enables the mobile node to get the best tunnel encapsulation mode in all cases.

The TSP client must be rerun everytime the IPv4 address changes. The TSP client can be put in the boot sequence, however, if the IPv4 address is changed without rebooting the TSP client might or might not reconnect, depending on the length of time between the address changes. A safe way to ensure that the tunnel is always re-established when the IPv4 address changes is to bind the process that changes the IPv4 address with the TSP client. For example, on Unix OSs such as Linux or FreeBSD, the DHCP client, ISC dhclient, can be customized to rerun the TSP client when the IPv4 address changes. For example, the **/etc/dhclient-exit-hooks** would contain the following commands:

```
if [ x$old_ip_address = x ] || [ x$old_ip_address != x$new_ip_address ];
then
   tspc
fi
```

Advanced Features

This section describes some advanced features and configuration of the TSP client.

TSP transport and encapsulation

The TSP session is initiated by the TSP client to the broker, specified by the server variable in the **tspc.conf** file. The TSP client first tries to connect to the broker over UDP. If no connection is made, then the TSP session is restarted over TCP. **v6udpv4** requires (because of a NAT) UDP transport, because when the TSP session is terminated, the same UDP channel is used to tunnel the IPv6 traffic. This permits re-use of the same NAT mapping which guarantees the reliability of the establishment of the tunnel for all types of NATs.

TSP versions

Version 1.X of the TSP protocol uses TCP as transport. Version 2.X uses UDP or TCP. Both use the assigned IANA port 3653. When a TSP client connects to a broker, the TSP client advertises to the broker the highest version of the TSP protocol it supports.

When a 2.X client connects to a version 1.X Migration Broker, the TSP client first connects using UDP. Because a 1.X broker does not listen on UDP, the TSP client times out, automatically falls back to TCP, and establishes the TSP session using TCP.

When a 1.X client connects to a version 2.X Migration Broker, the broker adapts to the TSP client by using the 1.X TSP protocol.

The client also advertise the 2.0.0 TSP protocol version. If the broker is 1.X TSP protocol compliant, the broker will deny the TSP client request. The TSP client request then restarts the TSP session advertising the 1.X TSP protocol version.

Version 2.0 of the TSP protocol added the **v6udpv4** encapsulation mode for NAT traversal and a keepalive mechanism, to keep the NAT mapping up while the tunnel is being used.

Operating system specifics

Linux

TSP client was tested on Redhat 7.2, 8.0 and Fedora Core 1. It is reported to work on most Linux distributions with kernel versions 2.4 or 2.6. Among the Linux distributions, Debian has made a package for the TSP client, called freenet6, available at: http://packages.debian.org/stable/net/freenet6.

On Linux, the **ipv6** kernel module should be pre-loaded. When using the **v6udpv4** encapsulation mode, the **tun** kernel module should also be pre-loaded. The following commands load the modules:

```
# modprobe ipv6
# modprobe tun
```

The TSP client on linux executes **`modprobe ipv6`** and **`modprobe tun`** to ensure IPv6 is enabled.

The **template** variable in `tspc.conf` should be set to **linux**.

The tunnel interface used on Linux by the TSP client depends on the configuration of the operating system. While the default is provided for most cases, if the tunnel is not established, look for changing the **if_tunnel_v4v6** or the **if_tunnel_v6udpv4** variables in the tspc.conf file by looking at which interfaces are already used by the node.

The TSP client requires the superuser (root) privileges.

FreeBSD

TSP client supports FreeBSD 4.X and up.

IPv6 must be enabled before the TSP client executes. To enable IPv6, add the following line in the **`/etc/rc.conf`** file:

```
ipv6_enable="YES"
```

The **template** variable in `tspc.conf` should be set to **freebsd**.

The TSP client is also included in the FreeBSD ports as `freenet6` under the `net` and `ipv6` subdirectories (`/usr/ports/net/freenet6`, `/usr/ports/ipv6/freenet6`). The port installs the files in the following locations:

```
/usr/local/bin/tspc
/usr/local/etc/tspc.conf
/usr/local/bin/tspc-freebsd.sh
/usr/local/bin/checktunnel.sh
/usr/local/etc/rc.d/freenet6.sh
```

The TSP client requires the superuser (root) privileges.

Installing the TSP client from the source code

To install the TSP client from the source code, the following steps should be done:

1. get the source code (.tgz or zip) and decompress it.
2. Go in the tspc directory and type: **make target=***osname* **all**
   where osname = windowsXP, freebsd or linux

When the compilation is completed, the following files are necessary for executing the TSP client:

- tspc binary file (tspc) is in the bin directory
- a sample tspc.conf file
- the template subdirectory containing the OS scripts.

To install the TSP client in /usr/local/tspc with the necessary files, use the following command:
make target=*osname* **installdir=***/usr/local/tspc* **install**

Customizing the TSP client

When the TSP client completes its transaction with the broker, the TSP client calls the shell script, as named by the **template** variable in `tspc.conf`, in the `tsp_dir/template` directory. All the information needed to configure the tunnel are pushed as environment variables from the `tspc` program to the shell script. The following table lists all the environment variables.

Table 1 Environment variables pushed to the OS script

Environment variable name	Description and value
TSP_TUNNEL_MODE	Tunnel encapsulation mode. Values=V6V4, V6UDPV4
TSP_HOST_TYPE	Type of node. Values=HOST, ROUTER.
TSP_TUNNEL_INTERFACE	Tunnel interface name on the host operating system. On FreeBSD for example, values=gif0, tun0.
TSP_CLIENT_ADDRESS_IPV4	IPv4 address of the TSP client.
TSP_SERVER_ADDRESS_IPV4	IPv4 address of the tunnel server
TSP_CLIENT_ADDRESS_IPV6	IPv6 address of the TSP client
TSP_SERVER_ADDRESS_IPV6	IPv6 address of the tunnel server
TSP_TUNNEL_PREFIXLEN	Prefix length used on the tunnel link.
TSP_HOME_INTERFACE	In router mode, the interface name used to advertise the prefix (TSP_PREFIX). The interface should be attached to a link where other IPv6 nodes are autoconfigured.
TSP_PREFIX	In router mode, the IPv6 prefix allocated by the tunnel broker.
TSP_PREFIXLEN	In router mode, the length of the IPv6 prefix allocated by the tunnel broker.
TSP_VERBOSE	Level of debug messages. 0=none. Values=0,1,2,3

These variables are useful to know only when one wants to modify the script.

TSP client license

The TSP client is provided under two licenses: GPL and commercial. The GPL license is provided with the source code in the GPL_LICENSE.txt. If you want to use the source code of the TSP client for commercial purposes that are prohibited in the GPL, please contact Hexago (info@hexago.com) to get information on the commercial license.

Copyright notice