Reduce Use of Microformats in IPC Parameters

Proposal: 0009
Authors: Operator Foundation
Status: Initial proposal

Implementation: TBD

Introduction

The PT 2.1 specification section on the IPC protocol has several parameters where the parameter value is in a microformat, where a microformat is defined here as text which must be in a specific format depending on the parameter. This proposal introduces new alternative parameters that reduce the use of microformats.

Motivation

Launching a transport client or server using the dispatcher is a complex process to learn and understand because of the number of parameters required. Several of the parameters have values which must be specified in microformats that the user must learn. Users making mistakes typing these microformatted values is a major source of problems for dispatcher users. By simplifying the format of parameter values to avoid microformats, the dispatcher can be easier to use for developers, particularly new developers.

Proposed solution

New parameters will be introduced which have simplified values. The old parameters will be kept for backwards compatibility.

Design

For each parameter that uses a microformat, a new parameter or set up related parameters will be introduced that has simplified values. These new parameters will only be added as command line flags and will not affect the environment variables, which will be kept the same for backwards compatibility. In all cases, the new form cannot be used simultaneously with the old form as they both specify the same information.

1. -transports [transport1,transport2,...,transportN]

This parameter value is in a comma-separated list microformat.

The new simplified parameter will be -transport [transportName] which allows only one transport to be specified. This is the most common use case.

2. -options [json]

This parameter value is in JSON format. This is especially problematic to type on the command line as it will contain quotes which must be escaped.

The new simplified parameter will be -optionsFile [filename], which allows the JSON value to be read out of a file instead of entered directly on the command line.

3. Transport Client -bindaddr [address:port]

This parameter value is in address:port microformat.

Two simplified parameters will be added: -bindhost [address] and -bindport [port].

4. Transport Server -bindaddr [transportName-address:port]

This parameter value is in a unique microformat. It is the most problematic because it is similar to the transport client -bindaddr microformat, but not compatible. Users often forget to leave out the transport name and this causes problems.

Two simplified parameters will be added: -bindhost [address] and -bindport [port]. These can be used in conjunction with the new single-transport specifier -transport to specify the address and port for a single transport. In this case, the format and behavior of -bindhost and -bindport are consistent between the client and server.

When multiple transports are specified on the server with -transports, then the older -bindaddr version must be used instead.

5. -target [address:port]

This parameter value is in address:port microformat.

Two simplified parameters will be added: -targethost [address] and -targetport [port]

6. -proxylistenaddr [address:port]

This parameter value is in address:port microformat.

Two simplified parameters will be added: -proxylistenhost [address] and -proxylistenport [port]

7. -transparent and -udp

The dispatcher supports four different modes of operation: transparent TCP, transparent UDP, SOCKS5, and STUN UDP. Currently the four modes are accessed by a combination of including or excluding the -transparent and -udp flags. This can be confusing to users when they want to figure out how to access a specific mode.

The simplified parameter value is -mode [modeName] where modeName is case-insensitive and can be any one of the following:

- transparent-TCP
- transparent-UDP
- socks5
- STUN

The default if no mode is specified is the same as the current default, which is SOCKS5-TCP.

Effect on API Compatibility

These changes only affects the IPC protocol. There is no effect on APIs.

Effect on IPC Compatibility

These changes to the API only add new command line flags and do not remove or change the meaning of any existing flags. Therefore, this change is backwards-compatible and would qualify as a minor change.

Alternatives considered

The alternative is to continue as things are currently. The current command line syntax is confusing for users and the source of many support requests. The most problematic is -bindaddr because of its inconsistent format between the client and the server and because the server-side format is unfamiliar to most users, but is similar to the more familiar client-side format. Users have also been confused when entering in any address:port value as to whether the port is required and sometimes do not remember that the “:” separator is used in this microformat.