How To Server

-AO2 Edition-

by Trey and scatterflower

LAST MODIFIED 15 MARCH 2021
last updated for akashi release **Apricot r1**…
…tsuserver commit **97af6e9 (6 MAR 2021)**…
…and AO client version **2.9.0**

The old serverD section of the guide can be found here.

Since making a server for *any* game is a multi-step process, I strongly recommend you start this guide from the beginning if you haven't looked at it before. Read carefully, paying special attention to **crucial information** put in bold. Any bolded text is something that *might not be obvious* and must be noted to run an Attorney Online server successfully.

Lots of people call this complicated. If you get unlucky, it might actually become complicated; that's just computers. That said, a well-set-up server, if you put in the time to set it up, could lead to lots of RP/casing/shitposting fun for you, your buddies, and even the AO community at large, and can be an educational experience for many prospective server owners. So if you want some of that, buckle in and get ready to master the art of server-ing.

(A reminder to check the FAQ if something goes wrong!)
(Another reminder to check the FAQ if something goes wrong!)
(One more reminder to check the FAQ if something goes wrong!)

TABLE OF CONTENTS

CHAPTER 1: Introduction

1a. Hosting

First off, consider why you're making this server. This'll put you into one of two boxes based on what kind of server owner you are.

Do you want to host a casual lawyering session with your friends that only lasts as long as you're willing to play? You're going to be an ad-hoc server owner. You will need a computer that can run your chosen server program, an Internet connection, and access to your router settings. Once you've followed the procedures in the following chapters and are set up, whenever you want to play, you can just start the server and tell people to join.

Do you want to build a *"""REAL"""* server with a community and stuff? You need to do at least one of these three things:

a) live on your own income and have a PC that can run 24/7 to host a server
b) have parents that let you screw around with the network settings and a PC that can run 24/7 to host a server
c) rent a VPS (Virtual Private Server) that you can run the game server on, and either be familiar with Linux on the command line or willing to learn

The least painful and least expensive option, in the long run, is C, because you can let some company worry about if the server's actually running. Hopefully, they'll be much better at it than you or me. If you don't know command line-only Linux, some good Google searching will do most of the teaching, but if you don't know where to start on that whole process, you can ask for help in the AO Discord (see end of doc).

If you can fulfill one of those three requirements, you have what you need to be a dedicated server owner. You can run your server around the clock, but this'll come with its own set of challenges: assembling a team of moderators, managing content updates, and generally keeping track of your server to make sure it's in good order. Do this diligently and you'll gain the respect of your playerbase.

1b. Port-Forwarding

Port-forwarding refers to a configuration process you need to perform on your network router to allow users to access the PC running the server. For the vast majority of home networks, **you need to port-forward.** If you do not, nobody but you will be able to connect.

For dedicated server owners using a VPS, the provider will make it easy to set up forwarding/incoming firewall rules; you should already be referring to their FAQ/help pages.

If you're on your own, though, your best source of help will be your router manual. If you don't have that, look up "how to port-forward" for the specific make and model of your router. **Depending on the manufacturer and product line, the process will look very different,** so I can't provide any specific advice for this.

Once you've figured out said process, **choose a port to forward, or two *different* ports if you want to open your server for webAO.** Forward TCP only. You can use (more or less) any number you want, but in AO, it's traditional to choose any number between 27010 and 27019. This should prevent any conflicts. Keep track of which one is the normal port and which is the webAO port; they go in their own places in the server config.

If this was successful, other players will be able to connect through the methods outlined in the section below. If that doesn't happen, check the FAQ.

1c. Connecting

The process of actually joining the server is different depending on the circumstances.

First, **if your server is running on the same computer you have your AO client on, you have to localhost into it.**

In the client, this means opening up serverlist.txt (the file that holds your Favorites), changing the port in this line to your server's normal port…

`127.0.0.1:27017:Default local server`

…and connecting to "Default local server" in the favorites menu.

In webAO, you use the direct connect method below, but the IP you connect to is 127.0.0.1. You should try the client first, though.

Other players connect through the server's public IP. If you're using a VPS, you do what they do.

This is often a no-brainer; if your server is public (it advertises to the masterserver) you can just select it. **If you're not on the public server list, though, you need to direct connect.** This also applies if you're connecting from a different computer than the server over a LAN, like your home network: instead of the public IP, use the server machine's private one.

In the client, this means opening up serverlist.txt and making an entry for your server. It contains the IP you're connecting to, followed by its normal port, followed by a name, all separated by colons. Like so:

`203.0.113.69:27010:Public Vanilla Server`

In webAO, players need to use a site link with the right info, including its webAO port. The server from before, assuming its webAO port is the next one over, would look like this:

`http://web.aceattorneyonline.com/client.html?mode=join&ip=203.0.113.69:27011`

(If you're hosting your content for webAO, your base folder goes in an asset query on the end, like the one Official Unofficial uses…

`&asset=https://webao.animatedchatroom.net/base/`

CHAPTER 2: Choosing your Server Software

As for the software that your AO server will run on, there are two current choices. This is their rundown:

tsuserver3. A Python 3-based server with a rich set of features suited to large AO servers, and especially designed to aid case roleplaying. Comes with powerful moderation functions based around its built-in SQLite database. As it's the older, more established software, most servers currently use tsuserver3 (or custom variants of it). Before you do anything you will be required to set up Python 3 and PyYAML, which is explained in the server's readme.

akashi. A C++ based server, designed to be as simple to use as possible, while still being largely compatible with tsuserver3. It requires no setup and should work as-is, and configuration is done with simple INI files. It is much faster than tsuserver3, and uses significantly less RAM. Additionally, it supports a very flexible moderator login system that allows fine-tuned permissions for large moderation teams. However, it is still new and under development, so some bugs and unimplemented features may show up.

After you've chosen which server software to bank your future success on, read the section below for that program. It will tell you how to correctly download, configure and run each of them.

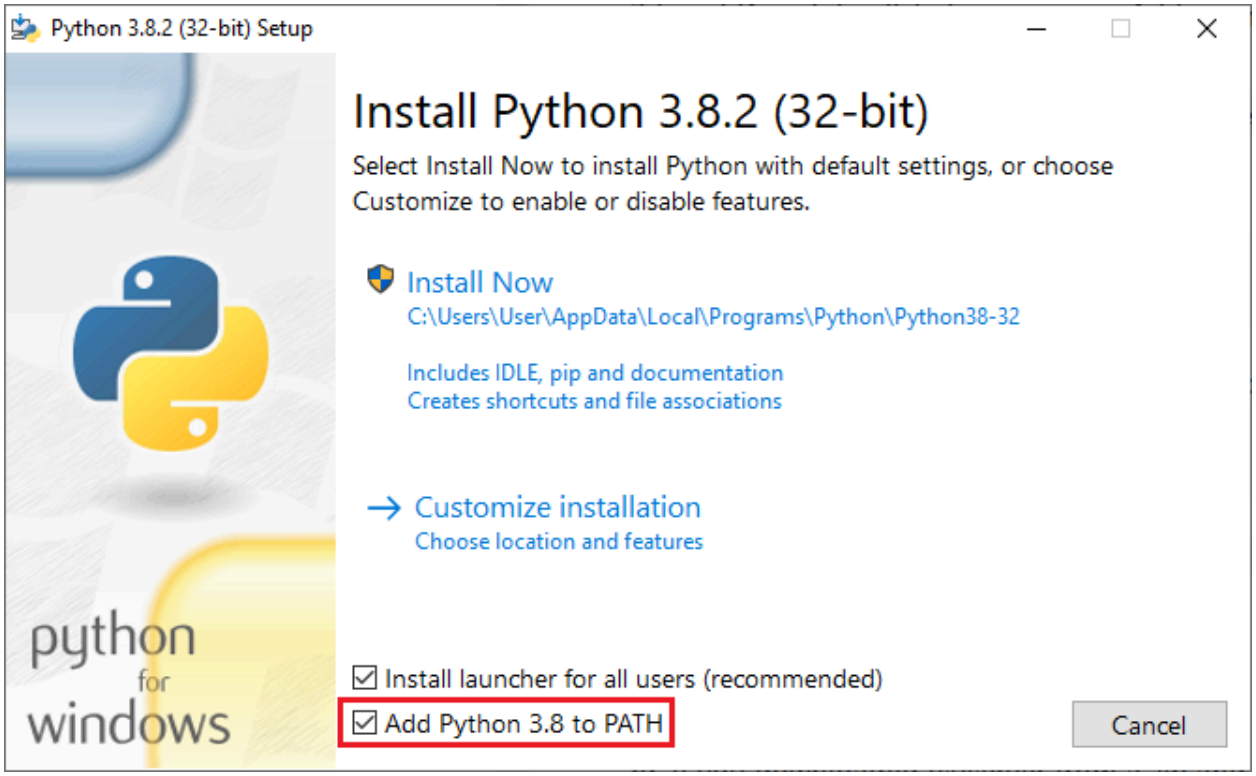
CHAPTER 3: Using tsuserver3

3a. Setup

Go to the tsuserver3 GitHub (or its GitLab mirror). Here, under the repo contents, you'll see **the readme**: this is your best reference for using the program and will be up-to-date with features as they're introduced into tsuserver.

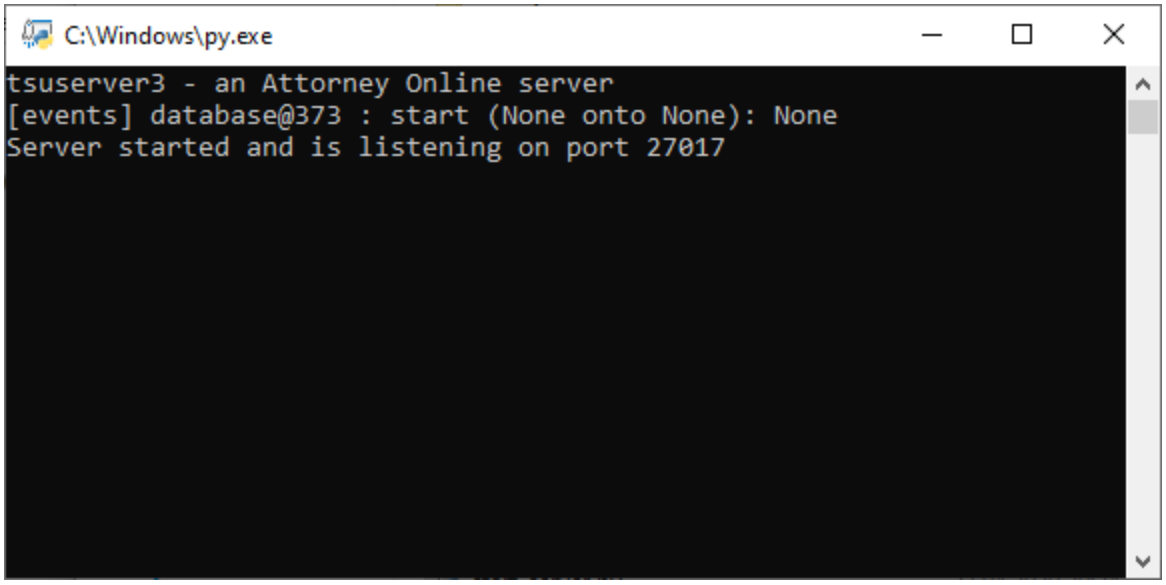
You'll have to follow the instructions in the readme closely to set it up. There's separate sections for Windows and Linux. I'm going to reiterate some of what it says here.

For Windows users: **when you're installing Python, remember to check "Add Python 3.x to PATH"**. Doing this will save some headache.



this will be outdated, of course

Once you've finished the "Install Dependencies" section of the readme, you're ready to start the next and longest step: configuration. Your goal throughout the process is to have a tsuserver window that says something like this…



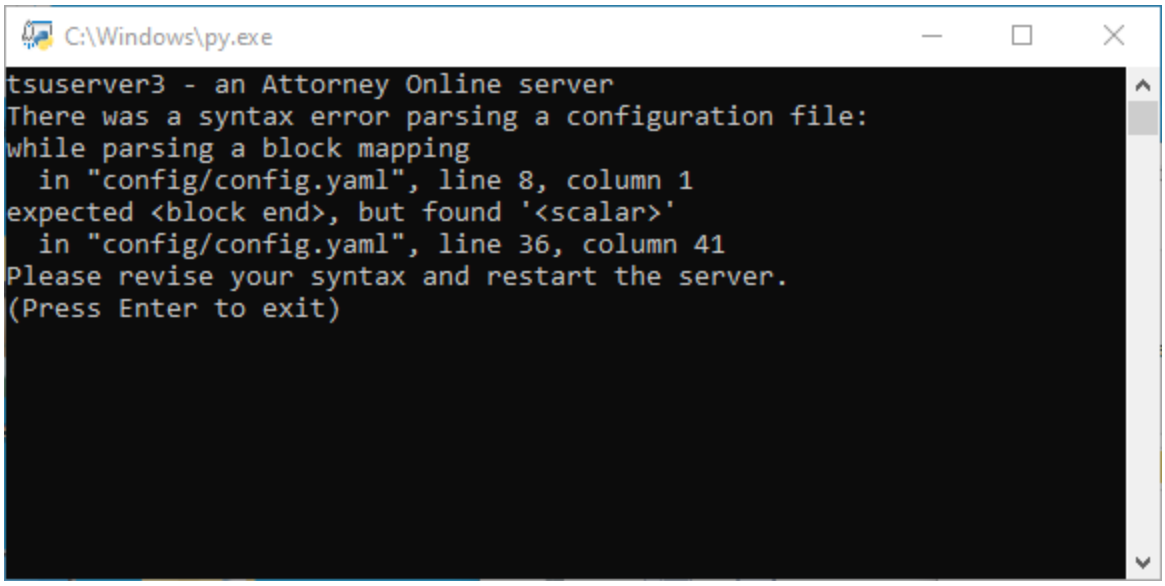
…meaning you've run the server with no configuration issues.

Read the rest of the setup instructions in the readme, and then continue to the next section for more details.

3b. Configuration

Did you read the “Configure tsuserver” section of the readme? We’re going over that first.

If you ever get something wrong when configuring tsuserver’s YAML config files, its window will look more like this:



This contains the first error that the config parser found, in techy YAML terms. Look at each line of the file it mentions **from the bottom up** (here I would look first at line 36) and make sure you didn’t mess up anything obvious (remembering the YAML tips I’m about to mention). If you can’t see it, use a YAML checker like http://www.yamllint.com/, as the readme says.

In your config folder, freshly renamed, there are seven YAMLs and a text file. You’ll want to at least look at each of them before your server is “ready”. Keep the following in mind, again, when editing YAMLs:

- Use spaces only; **do not use tabs** to indent.
- **If you need to use symbols in a string of text,** like in an MotD or song name, **put 'single quotes' around it.** If you need to use a

single quote *in* that string, make it '' two of them; it'll be read as one single quote. This is what the pros call an escape sequence, and it lets the config parser know you didn't just end the string.

The following is a reference of each config file and what they specify.

config.yaml

This is the main config file, containing the server's most vital settings, like IP/port info, server name/description/MotD, and whether or not to advertise on the masterserver list. Each option is commented and explained in the file. Note that your normal port goes in `port` and your webAO port goes in `websocket_port`. Remember: they cannot be the same port.

The options from `timeout:` downward are debugging- and security-based, and they can probably stay at their default (they're based on Official Unofficial's settings). However, you may want to change `debug:` to true if you want to diagnose any runtime problems, and when getting support on the AO Discord for such problems, they'll ask you to turn it on.

areas.yaml

Your chatrooms where all the fun goes down. Configured with a series of area entries that start with their name.

area: The area's name.

background: The name of the background that this area will start with. Should be one of the options in backgrounds.yaml.

bglock <true/false>: Default setting for this area's background lock, which sets whether or not *anyone* can come in and change the background.

evidence_mod: Special keyword that determines who can change evidence in this area. Refer to /evidence_mod command in the readme.

locking_allowed <true/false>: Whether users can lock this area for themselves and their buddies.

iniswap_allowed <true/false>: Whether players can use the iniswap "glitch" in this area, or send emotes not found in the characters/ folder.

In addition, the following options can be added manually to specify their non-default value. Their default value is CAPITALIZED here:

showname_changes_allowed <TRUE/false>: Whether players can change their showname from the default (name of the character)

shouts_allowed: <TRUE/false>: Whether players can use Objection! and other such shouts.

jukebox: <true/FALSE>: Whether the area uses jukebox mode for music management by default. See the /jukebox… commands in the readme.

noninterrupting_pres <true/FALSE>: Whether non-interrupting preanimations are enabled by default. This means all characters will begin talking immediately without a preanim delay.

the characters/ folder

If you do not have any areas in *areas.yaml* with iniswap disallowed, this folder is not used.

Otherwise, it should be a folder, organized the same as the characters/ folder in the client, with only the char.ini files. **You do not need to put any files in here besides char.inis.** tsuserver will check if all emotes used in an iniswap-locked area match the emote settings specified in these char.inis, and reject any that don't. Any characters that do *not* have a char.ini placed here, and that appear in your *characters.yaml,* will be free to use any emote.

The .inis in the vanilla client come with tsuserver and should already be in the folder.

characters.yaml

A list of characters (by folder name) that will be loaded into your server's character menu.

backgrounds.yaml

A list of backgrounds (by folder name) that will be available to switch between. The actual background that each area has on server startup is set in areas.yaml.

iniswaps.yaml

Exceptions to the iniswap_allowed setting. Shows which characters can be swapped out for each other with the iniswap “glitch”.

music.yaml

A list of file names to go into the music list. tsuserver is fancy and does them in categories. For your own safety, put 'single quotes' around each song and category name to avoid errors.

`category:` `Name of the category that each of the following songs will appear under.`

`songs:` `Paired -name and length entries for each song that goes into the category:`

`name:` `Name of the .mp3 file as it appears in sounds/music in your server’s contents, e.g.` **`'Annonce the Truth(T&T).mp3'`**

`length:` `Length in seconds of the song before it loops again. Set to 0 for no looping.`

dice.yaml

The lines used for the /rolla and /8ball dice commands. Each header (besides `8ball:`) is an ability dice set. Refer to the readme.

iprange_ban.txt

A list of banned IP ranges: networks or autonomous system numbers that are automatically denied access. It currently does not support specifying netmasks, so just include 1 to 4 octets of the address.

“In English please”…?

This feature stems from the need of Official Unofficial, the warzone of AO, to ban ranges of IP addresses used by VPN providers or Tor exit nodes, thereby preventing spammers from using these services to fake their IPs and bypass bans.

This feature is not foolproof and the list that default tsuserver comes with may change occasionally. However, the probability that this list prevents normal users from connecting is minimal, so unless you have a reason to allow all of these things access to your server, you can keep the default list. If you just don’t like the idea, feel free to blank the file; the bans will no longer be in effect.

In the event that you want to allow someone specific who’s connecting from a banned IP range or ASN (and you don’t just want to clear the list), consult your debug logs and ask the AO Discord if you need further help (see end of doc).

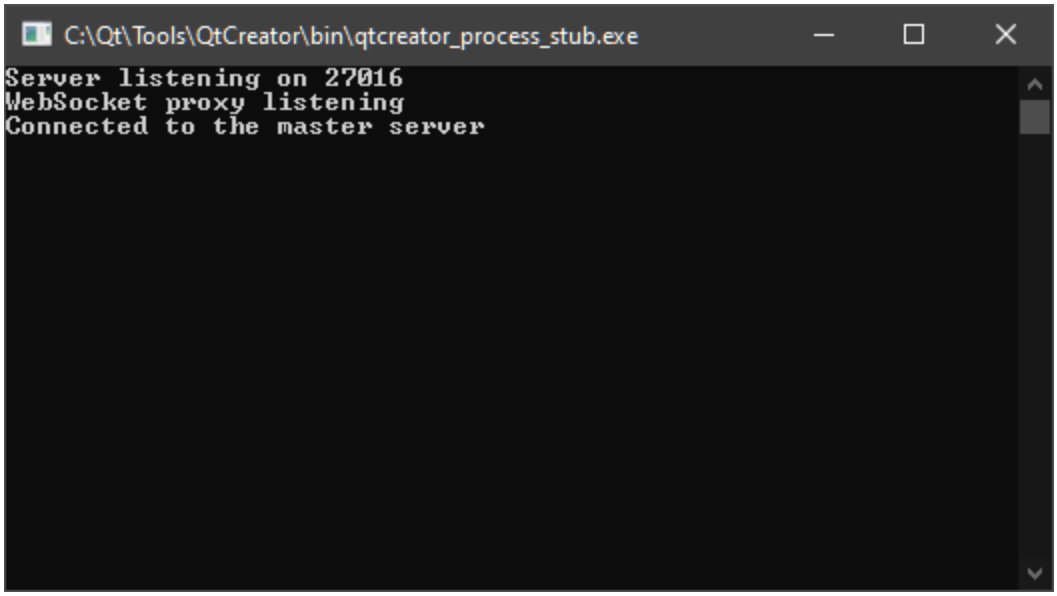
CHAPTER 4: Using akashi

4a. Setup

akashi is a relatively new addition to the AO server software lineup, supporting most of the major features of tsuserver3. It is written in C++ using Qt, and boasts faster performance across the board than tsuserver. You may want to use it if you're worried about performance, or if you don't want to have to install anything extra to run it.

Go to the akashi GitHub. On the right of the page, you should see "Releases". Download the latest release for your operating system (only Windows and Linux are supported, currently) and extract it to the location you want to run the server from.

Once you've done that, you're ready to start the next and longest step: configuration. Your goal throughout the process is to have a window that says something like this…



…meaning you've run the server with no configuration issues.

4b. Configuration

In your config folder, there are five files. You'll want to at least look at each of them before your server is "ready".

config.ini

This is the main config file, containing the server's most vital settings, like IP/port info, server name/description, and whether or not to advertise on the masterserver list. Each option is commented and explained in the file. Note that your normal port goes in `port` and your webAO port goes in `webao_port`. Remember: they cannot be the same port.

It's recommended that you set your `modpass` value now. You'll need it in order to set up permissions later.

areas.ini

Your chatrooms where all the fun goes down. Configured with a series of area entries that start with their name.

[id:area]: The area's name and id. Areas will be put into the area list according to their ids. Doubles as a section header.

background: The name of the background that this area will start with. Should be one of the options in backgrounds.txt.

evidence_mod: Special keyword that determines who can change evidence in this area. Refer to /evidence_mod command in the readme.

protected_area <true/false>: Whether or not any user can become a CM (Case Manager) in this area. Note that users with the "CM" permission will be able to use CM commands regardless of their CM status, and are thus unaffected by this setting.

iniswap_allowed <true/false>: Whether players can use the iniswap "glitch" in this area.

characters.txt

A list of characters (by folder name) that will be loaded into your server's character menu.

backgrounds.txt

A list of backgrounds (by folder name) that will be available to switch between. The actual background that each area has on server startup is set in *areas.ini.*

music.txt

A list of file names to go into the music list. In akashi, any file name that does not contain a period (".") is considered a category. You can use this to organize your music.

4c. Configuration (part deux)

This part is technically optional, but it's heavily recommended that you set up **advanced permissions** for your server. Once your server is up and running, login with the modpass set by `modpass` in config.ini.

Once you're logged in, type "/changeauth" in OOC to begin the process of converting the server to advanced authentication mode. Follow the instructions the server gives you, and you will be logged in as root. Create new users with /adduser {username} {password}, and assign them permissions with /addperms {username} {permission}. The available permissions are as follows:

- **KICK** - Users with this permission can kick other users.
- **BAN** - Users with this permission can ban other users.
- **BGLOCK** - Users with this permission can modify the "background lock" status of areas with /bglock in order to prevent other users from changing the background.
- **MODIFY_USERS** - Users with this permission have the ability to create, remove, and modify users. This is a dangerous permission to grant!
- **CM** - Users with this permission can use CM-only commands in areas that they do not own.
- **GLOBAL_TIMER** - Users with this permission can modify the server-wide timer using /timer 0.
- **EVI_MOD** - Users with this permission can change the evidence_mod of areas.
- **MOTD** - Users with this permission can modify the MOTD, which is sent to all users when they join.
- **ANNOUNCE** - Users with this permission can send server-wide official announcements.
- **MODCHAT** - Users with this permission can see and use moderator-only messaging channels, and can speak officially with

/gm

- **MUTE** - Users with this permission can mute other users.
- **SUPER** - Granting this permission will give the user all other permissions. This is a dangerous permission to grant!

CHAPTER 5: FAQ

Q. Other people can't connect to my server. What next?

- *Check your port forwarding and check again.*

{{Make sure it matches the config in tsuserver: that they're using the normal port on client AO or the websocket_port for webAO.}}make generic Make sure you have TCP forwarded; you do not need UDP for AO to work. Make sure that the local IP you forwarded the port to is the same one you get in the IPv4 Address field of ipconfig; don't mix up your default gateway with your IP address. If your router makes you apply port forwarding presets and then enable them, make sure you've done that and that it has actually applied.

- *If you're sure about all that, there may be a software-level firewall in the way.*

Turn Windows firewall off (or whatever your instance of Linux is using) and see if the problem is resolved. This won't really subject you to anything dangerous; in normal home network setups, you're already being protected by your router. This isn't a situation where software-level protection is needed.

If the firewall is the problem, you can either configure an exception, or just leave it off if you weren't using it for anything program-specific anyway. If you would rather configure an exception, on Windows:

open up Firewall with Advanced Security, go to Inbound rules, add a new rule, set to custom, choose all programs, select TCP and set remote port to the one you're using, keep all IP addresses, set allow connection, have profile as private and public, then finally name it and finish

- *If neither of those are the case, your problem is likely specific to your hardware set-up.*

Specifically, make sure you don't have two routers that are doing NAT at the same time. If the router you're connecting to says its public/WAN IP is a private IP (starts with 192.168…, 172.16to31…, or 10…) then it's pointing to another local router (probably ISP-issued hardware) and traffic cannot go through there. The best fix to this is to either set two

port-forwarding rules, one on each router, that route through each other, or have one of your routers run in bridged mode and the other as the normal router you forward the port for. Consult Google.

In the case that you can't configure the NAT-ing router in question (crap college dorm, etc), port-forwarding may be impossible. You won't be able to host servers from that network. If you were planning an ad-hoc session, someone else may have to host. If you were planning a 24/7 session, consider renting a VPS to host it for you.

Q. I can't connect to my server. What next?

- *Are you sure you're using the right method? See section 1c.*

If you get "Server started and is listening on port X" in tsuserver (no errors), and can't connect *with the proper method* while it's running, contact the Discord.

Q. This is useless. I have a different problem.

Contact the AO Discord. Ask your question in #tech_support, and ping @Works IT if you don't get attention. (And don't ask *to ask*, just ask the question.)

Also, if you can't do this or you fail to find your solution there, you can ask direct questions about server-making stuff at

publicvanilla@gmail.com.

Be aware that this will be much slower than simply asking in live chat.

More AO stuff at aceattorneyonline.com