Setup Guide:
Unity Floating Licensing

Table of Contents

About Unity Floating Licensing

The Unity Floating Licensing system has two components to install: the Unity Licensing Server on your network and the Unity Licensing Client on machines that run the Unity Editor.

The Unity Licensing Server is an HTTPS server built on the ASP.NET framework. It manages a pool of Unity licenses. Machines with the Unity Licensing Client installed connect to this server to request licenses.

When a client machine requests a license, the server removes it from the pool until the client machine releases it. If the client machine does not renew the license after a specified amount of time, the server releases it back to the pool automatically. This means that you can use the server to manage a pool of Unity Editor licenses: when a user starts the Unity Editor on their machine, the client connects to the server to request the license. When the user closes Unity, the client releases the license back to the pool.

Any client machine configured to connect to the server can request a license as long as there are still licenses left in the pool.

About entitlement licensing

The Unity Floating Licensing system is an entitlement licensing solution. An entitlement is the right to use a specific feature of Unity: for example, a Unity license might include the entitlements to use the dark theme in the Unity Editor and to use Unity in headless mode to build games.

Requirements

Server requirements

To set up the Unity Licensing Server, you need:

- A persistent machine

  The Unity Licensing Server must run on a persistent machine on your local network. If machine bindings change, the server license will become invalid.

- A supported Windows or Linux platform

  The Unity Licensing Server supports:

  - Windows 7 SP1+, Windows 8, Windows 10 (64-bit versions)
  - Ubuntu 16.04+, Red Hat Enterprise Linux 7.x, CentOS 7.x (64-bit versions)

- The server installation files

  Your Unity Support representative provides the required files. See Getting started for details.

- A server license

  To obtain a server license, extract the server files and follow the command line setup. For an overview, refer to the document Quick Start: Configuring Unity Licensing Server.

- .NET Core 2.1.x

  Required to run the dotnet commands.

Client requirements

To set up the Unity Licensing Client, the client machine must have:

- A supported operating system

  The Unity Licensing Client supports:

  - Windows: Windows 7 SP1+, Windows 8, Windows 10 (64-bit versions)
  - MacOS: macOS 10.12+
  - Linux: Ubuntu 16.04+, Red Hat Enterprise Linux 7.x, CentOS 7.x (64-bit versions)

- Unity Editor version 2018.4 or later

- The Unity Licensing Client

If you're using Unity version 2019.1 or earlier, you'll need to download and install the licensing client manually. Later versions bundle the licensing client with the Unity Editor. See Getting started for details.

- Unity Hub

  The Unity Hub must be installed on the client machine. For installation instructions, see the Hub Documentation in the Unity Manual.

Getting started

To use floating licensing, you need to configure both the Unity Licensing Server and the Unity Licensing Client.

Note: For a higher-level walkthrough of the setup process, refer to the separate PDF Quick Start: Configuring the Unity Licensing Server.

Required files

Contact your Unity Support Representative for the following .zip files:

- `Unity.Licensing.Server.zip`
- `Unity.Licensing.Client.zip`

  Note: Unity 2019.2 and later ship with the Unity Licensing Client already installed, so you'll only need the client .zip if you use Unity 2019.1 or earlier.

Overview

The first section of this guide, Configuring the Unity Licensing Server, walks you through the process of setting up and testing your license server from the command line interface. If you choose, you can also set up an OS service for your license server.

As part of the setup process, you'll generate a server registration request file and send it to your Unity support representative. Your representative will send back your license information in a .zip file. When you import this file from the command line, your server can run.

The second section of this guide, Configuring the Unity Licensing Client, helps you set up client-side configurations. These configurations should be applied on all client machines.

Note: You must also install the Unity Hub on all client machines. For details, see the Hub Documentation in the Unity Manual.

Configuring the Unity Licensing Server

Before you start

If you are upgrading your server and want to keep your existing audit history, save your existing database file in a safe location and restore it after the server is successfully installed. Refer to Restore database for instructions.

The default `LocalDatabase.sqlite` database file is located in the `LocalDatabase` folder with the server executable.

Note: Not all configuration files can be restored if damaged. You should back up your configuration files after the server is successfully set up and running so that you can restore a working server configuration if necessary. Refer to the sections Back up server configuration and Restore server configuration for details.

To set up the Unity Licensing Server, you need to extract the files on a dedicated machine and configure the server with a command-line interface.

Extract server files

To set up the Unity Licensing Server, extract the contents of the `Unity.Licensing.Server` archive to a directory on the dedicated server machine. This becomes the launch directory for your Unity Licensing Server.

Suggested paths:

Platform	Suggested Server Path
Windows	`C:\UnityLicensingServer`
Linux	`~/UnityLicensingServer`

Overview of CLI configuration

The command-line interface provides the following commands to help set up your license server:

Name	Syntax	Description
Setup	`Unity.Licensing.Server setup`	Configures server
Import	`Unity.Licensing.Server import`	Imports license archive file
Create Service	`Unity.Licensing.Server create-service`	Creates OS service
Generate Report	`Unity.Licensing.Server generate-report`	Generates server troubleshooting archive

To see the full syntax of each command, type `--help` after the Unity.Licensing.Server executable.

You should run the Setup and Import commands first. Once you confirm that the server is running successfully, run the Create Service command.

Setup CLI command

The Setup command creates a file that contains the license server configuration parameters.

Overview

All server configurations are saved in the file `licensing-server-config.json`. Refer to Server paths section to find the location of the file.

If `licensing-server-config.json` already exists, running the Setup command uses the existing configurations as default values.

Initial setup

To start the interactive setup from the command line, run the Setup command as shown in the following table:

Platform	Command
Windows	`.\Unity.Licensing.Server.exe setup`
Linux	`./Unity.Licensing.Server setup`

1. When prompted, enter a meaningful name for your server.

1. When prompted, enter a meaningful name for your server.

```
- - - -
Welcome to Unity Licensing Server setup command line interface.
This setup will help you configure your license server and generate server registration
request file.
- - - -

Enter the server name (e.g. LicenseServer):  [DESKTOP] TestServer
```

2. Configure HTTPS/SSL.

```
Do you want the licensing server to use HTTPS? [Y/n]
Enter path to the certificate PFX file (Press Enter to skip):
Enter the PFX password for "httpscertificate.pfx":
```

   To use HTTPS, you need to have a server certificate in Personal Information Exchange Format (.pfx) format. Unity does not provide this certificate: your IT team is responsible for managing this certificate.

   - If you don't have a server certificate and want to skip this step, enter `n` and press Enter.
   - If you have a server certificate, enter `y` and press Enter, then set the path and password of the .pfx file.

3. Choose the network interface for the license server to provide service on. This interface should have its firewalls configured so that Unity.Licensing.Client can communicate to obtain a license.

```
List of available network interfaces on this host
- [1] en0 (8C:85:90:CA:72:DC) 192.168.0.51
- [2] gpd0 (02:50:41:00:01:01) 10.1.4   2228
Enter the index number of the network interface which server will operate on:
```

   Enter the index number of the interface and press Enter. (In the example above, entering `1` selects en0.)

4. Enter the port number that the clients will connect to and press Enter.

```
Enter server's listening port number (between 0 and 65535):  [80]
```

   To avoid port conflicts, use a port outside the known range (with a 4- or 5-digit number).

   Note: If the port you choose is already used by another application and the server fails to launch, run the Setup command again from step 1 and select a different port.

5. Configure admin access.

   The license server has the ability to limit administrator access to admin endpoints. These

endpoints contain detailed server information and are useful for diagnosing and troubleshooting common configuration problems.

Only the IP addresses that are set in the Admin IP Whitelist are allowed to access the admin API. To access the admin API, go to: http://SERVER-IP-ADDRESS:PORT/v1/admin/status.

- To allow the server to be accessed from inside the server machine itself and from the publicly reachable default IP address, enter `y` and press Enter. Otherwise, enter `n` and press Enter to manually add an Admin IP whitelist.

```
Add default addresses to the Admin IP Whitelist (127.0.0.1, ::1, 192.168.0.51)? [Y/n]
List of current white-listed admin IP addresses:
`- 127.0.0.1
`- ::1
`- 192.168.0.51
Add an additional admin IP address to the white list? [y/N] y
Enter admin IP address (Press Enter to skip):
```

To define an additional admin IP, enter `y` and press Enter, then enter the new IP to be added to the whitelist. Otherwise, enter `n` and press Enter.

When setup is completed successfully, the CLI displays output like the following:

```
Generating signing key... Done

Generating server configuration ...Done
Generating services configuration ...Done
Reloading configuration... Done
Generating server registration request file... Done

`- - - -
Setup Completed!
`- - - -

> Server Registration Request
Please go to Unity ID portal -> Organizations -> [Organization Name] -> License servers,
upload this file and follow the instructions to receive a compressed license archive
file.

File to upload:
/home/adminuser/server-registration-request.xml

> Services Configuration File
Generated services-config.json file must be copied to the following location on all end
user computer in order to enable floating licensing:
   - Windows: %PROGRAMDATA%\Unity\config\
   - MacOS:  /Library/Application Support/Unity/config/
   - Linux:  /usr/share/unity3d/config/

File to copy:
/home/adminuser/services-config.json

WARNING: Please make sure to backup the following directory as it contains essential data
for operating the licensing server:
   ~/.config/unity3d/Unity/LicensingServer
```

The output includes the paths of the following generated files:

- Services Configuration (`services-config.json`)
- Server Registration Request (`server-registration-request.xml`)

Every time you execute the setup command, the file `services-config.json` needs to be deployed on all end-users' computers for the new configuration to be used.

The file `server-registration-request.xml` contains machine binding information and is used to generate the server license. It includes the following information:

- `"FirstPhysicalAddress"`: the machine's MAC address
- `"Platform"`: the machine's operating system
- `"ProcessorCount"`: the number of processor cores on the machine
- `"MachineName"`: the name of the machine

Go to the Unity ID portal (https://id.unity.com) > Organizations > [Organization Name] > License

servers and upload this file. Follow the instructions on screen to download a compressed license archive in .zip format.

Restore database (optional)

If you have your old database file saved and want to restore it for your new server installation to keep the audit history.

1. Put the existing database file into the following folder:

Platform	Launch Command
Windows	`%LOCALAPPDATA%\Unity\LicensingServer\data\`
Linux	`~/.config/unity3d/Unity/LicensingServer/data/`

2. Rename the file to `LicensingServerDb.sqlite`.

Import CLI command

When you receive the .zip from your Unity support representative, run the Import command.

This command imports the license archive, stores archived files at the correct paths, and sets corresponding keys in the `licensing-server-config.json` file.

The .zip includes the following files:

- Server License
  Signed server license file (.xml) for a purchased subscription.
- License Signing Certificate
  Personal Information Exchange Format (.pfx) signing certificate file and password that the server uses to sign licenses issued for client machines.
- Server License Certificate
  Certificate (.cer) that the server uses to validate the license file. This validation activates the entitlement that allows the server to issue licenses.

Overview

The Import command does the following:

- Puts the server license file into a platform-specific directory for the current user to be used by

license server.
- Puts signing delegation files into the platform-specific directory and configures the license signing certificate password, in `licensing-server-config.json`.

For more information about the paths, refer to Server Paths.

Note: Once the Import command is successfully completed, the server must be restarted to be able to read the new license file. If the license server is running as a service, refer to Restarting the Service for more information.

To import a license archive, run the following command and replace `[license archive path]` with the full path of your license archive.

Platform	Command
Windows	`Unity.Licensing.Server.exe import [license archive path]`
Linux	`./Unity.Licensing.Server import [license archive path]`

Output is similar to:

```
Extracting files...Done
Validating archive content...Done
        LSD type import detected
Importing server license files...Done
Importing delegation file...Done
List of available toolsets on this server
  - [1] LicenseServer_3573461705080_1 [Unity Enterprise for Games(Floating)]
Enter the index number of the toolset that should be used by default: 1
```

At this step you will be shown a list of all available toolsets for this license server. Select a default toolset Id. This id will be saved in the `licensing-server-config.json` file and will be used at the time serving licenses. For more information about toolsets refer to Advanced Server Configuration.

Once import is successfully finished the output will be similar to:

```
Successfully imported licensing files. You may run the server.
Please note that if the server is already running, you will have to restart it.
```

Once this step is completed, proceed to Testing the license server.

Note: At this stage, the license server is fully configured to run and issue licenses to clients, but still needs to be started manually. To create a service that launches automatically at boot, follow the steps in Create Service CLI command.

Create Service CLI command

The Create Service command allows the server process to run as an OS service.

Once this step is completed, you can proceed to Testing the license server.

The service can be configured to start automatically on boot. The service can also be manually paused, stopped and restarted.

Creating a service

The steps to create a service differ by platform.

Creating a service on Windows

Note: This command requires administrator privileges. Before you start, make sure the console is running with elevated privileges. You must use an administrator account with a non-blank password.

The user must also be given the right to log on as a service. To assign this right:

1. Search for `Local Security Policy` in Windows Search and open the policy editor.
2. In the left-hand menu, navigate to Security Settings > Local Policies > User Rights Assignment and select Log on as service.
3. Click Add User or Group.
4. Type the username in the text box and click Check Names > Ok. 5.
5. Click OK.

To create the service, run the following command:

```
.\Unity.Licensing.Server.exe create-service
```

Enter the password of the admin user used to create the service.

Successful output is similar to the following example:

```
- - - -
Welcome to Unity Licensing Server create service command line interface.
This command line interface will help you create Unity.Licensing.Server service.
- - - -

Enter password for user 'HQ\adminuser':  ***********
Windows service installation successful!
Service Name = Unity.Licensing.Server
Description = The Unity Licensing Server is a secure web server to host and manage Unity
licenses on your local network
```

To check the status of the service, run the following command:

```
sc.exe query Unity.Licensing.Server
```

If the service is running correctly, the output is similar to the following example (look for `STATE`):

```
SERVICE_NAME: Unity.Licensing.Server
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

To troubleshooting, manage the service under `Unity.Licensing.Server` in Windows Services.

Creating a service on Linux

Note: This command requires administrator privileges. Make sure to run this command with elevated privileges by entering `sudo` in the terminal.

To create the service, run the following command:

```
sudo ./Unity.Licensing.Server create-service
```

Successful output is similar to the following example:

```
- - - -
Welcome to Unity Licensing Server create service command line interface.
This command line interface will help you create Unity.Licensing.Server service.
- - - -

Creating service for user 'adminuser' ...
Enabling service to start automatically after (re)boot ... Done
Starting service ... Done
unity.licensing.server.service has been successfully created and running.
```

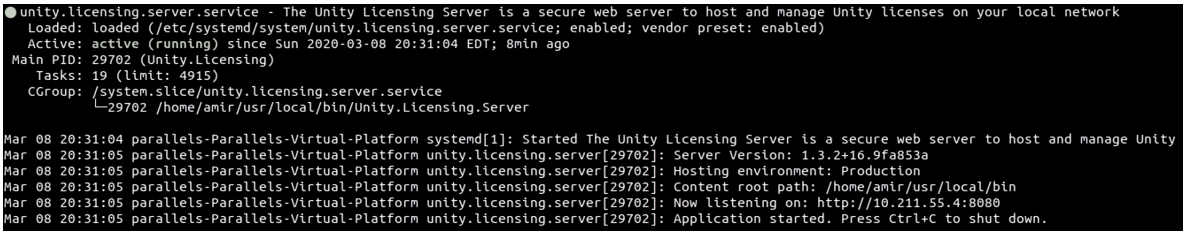
The command does the following:

- Creates a systemd service unit named `unity.license.server.service`.
- Configures the service to run under the current (non-root) user account.
- Configures the service to start automatically on boot.
- Starts the service.

To check the status of the service, run the following command:

```
sudo systemctl status unity.licensing.server.service
```

If the service is running correctly, the output is similar to the following (look for `Active:`):



Restarting the service

The service can be stopped or restarted using standard system commands.

Please follow instructions based on the chosen platform:

Note: This operation will interrupt the license server's availability. Make sure this operation is performed during hours when traffic is low or during your organization's maintenance period.

Restarting the service on Windows

1. Run the following command:

   ```
   sc.exe query Unity.Licensing.Server | findstr /i State
   ```

   If the output shows `STATE : 4 RUNNING`, run the following command before proceeding:

   ```
   sc.exe stop Unity.Licensing.Server
   ```

   When the output shows `STATE : 1 STOPPED`, proceed to the next step.

2. Run the following command:

   ```
   sc.exe start Unity.Licensing.Server
   ```

3. Run the following command:

```
sc.exe query Unity.Licensing.Server | findstr /i State
```

When the output shows `STATE : 4 RUNNING`, the service is successfully restarted.

Restarting the service on Linux

1. Run the following command to start or restart the service.

   ```
   sudo systemctl restart unity.licensing.server.service
   ```

2. Run the following command to verify that the service is running:

   ```
   sudo systemctl status unity.licensing.server.service | grep Active
   ```

   If the service is running correctly, the output resembles the following:

   ```
   Active: active (running) since Fri 2020-02-14 16:44:59 EST; 2s ago
   ```

Deleting the service

If required, the service can be deleted. The steps to delete a service differ by platform.

Deleting the service on Windows

To completely delete the current service:

1. Run the following command to stop the service:

   ```
   sc.exe stop Unity.Licensing.Server
   ```

2. Run the following command to check the server status:

   ```
   sc.exe query Unity.Licensing.Server | findstr /i State
   ```

   Once `STATE` shows `STOPPED`, proceed to the next step.

3. Run the following command to delete the service:

   ```
   sc.exe delete Unity.Licensing.Server
   ```

   When the service is deleted, the following message is displayed:

   ```
   [SC] DeleteService SUCCESS
   ```

Deleting the service on Linux

To completely delete the current service:

1. Run the following command to stop the service:

   ```
   sudo systemctl stop unity.licensing.server.service
   ```

2. Run the following command to disable the service and prevent it from running on the next boot:

   ```
   sudo systemctl disable unity.licensing.server.service
   ```

3. Run the following command to verify that the service is disabled:

   ```
   sudo systemctl status unity.licensing.server.service | grep Loaded
   ```

   If the service is disabled, the following output is displayed:

   ```
   Loaded: loaded (/etc/systemd/system/unity.licensing.server.service; **disabled**; vendor preset: enabled)
   ```

4. Run the following command to delete the service unit:

   ```
   sudo rm -rf /etc/systemd/system/unity.licensing.server.service
   ```

5. Run the following command to reflect the change:

   ```
   sudo systemctl daemon-reload
   ```

Back up server configuration

To back up the existing configuration of the server, save a copy of the `LicensingServer` folder in a safe location. This folder is located in:

Platform	LicensingServer directory path
Windows	`%LOCALAPPDATA%\Unity\LicensingServer\`
Linux	`~/.config/unity3d/Unity/LicensingServer/`

Restore server configuration

To restore the server configuration from the backup, replace the `LicensingServer` folder with your

backed-up version.

Generate Report CLI command

This command collects and archives licensing related data required for troubleshooting the licensing server.

Platform	Launch Command
Windows	`.\Unity.Licensing.Server.exe generate-report --output-directory [directory path]`
Linux	`./Unity.Licensing.Server generate-report --output-directory [directory path]`

`--output-directory` argument is optional. If you omit it, by default archive will be created in your current directory.

To generate the archive in a specific directory, replace `[directory path]` by the destination path.

If command executed successfully, you will see the generated archive path output.

Here is the sample output of a successful execution for `.\Unity.Licensing.Server.exe generate-report --output-directory=D:\` command:

```
Creating temporary folder...Done
Collecting configuration files...Done
Collecting license files...Done
Collecting delegation files...Done
Collecting log files...Done
Creating summary file...Done
Archiving files...Done

Archive successfully created: D:\Licensing.Server.Support.2020-06-12-025617.zip
```

Status CLI command

The status command shows information for these health checks:

Health Check	Description
`HttpConfigHealthCheck`	Kestrel configuration
`ServerLicenseHealthCheck`	Status of licenses and delegations
`LicensingHealthCheck`	License configuration and signing key
`DatabaseHealthCheck`	Status of the connection to the database
`PluginInformationProvider`	Server plugins

You can also see the status of these health checks on the server admin page.

Execute the command with the following syntax:

Platform	Launch Command
Windows	`.\Unity.Licensing.Server.exe status`
Linux	`./Unity.Licensing.Server status`

The output appears similar to this:

```
Server Version: 1.7.0.7cc2de3
Healthy
HttpConfig: Healthy
HTTP config is healthy
  [OK] Kestrel: Found config section Kestrel
  [OK] Kestrel:Endpoints:HTTP:Url: Found valid config Kestrel:Endpoints:HTTP:Url:
http://10.255.255.148:80
ServerLicense: Healthy
Server license and delegation are valid
  [OK] ServerLicense::Context::FirstPhysicalAddress: 48:2a:e3:87:3b:19
  [OK] ServerLicense::Context::Platform: Win32NT
  [OK] ServerLicense::Context::ProcessorCount: 12
  [OK] ServerLicense::Context::MachineName: LAPTOP-2Q28U8QR
  [OK] ServerLicense::ContextCount: Found 4 out of 4 context values
  [OK] ServerLicense::EntitlementGroup::test-licenseServer_3573567292644_1: EntitlementGroup:
test-licenseServer_3573567292644_1 (from: test-
licenseServer_3573567292644_1-20200925_145658.xml) has valid server entitlements
  [OK] ServerLicense::EntitlementGroupCount: Found 1 valid server license(s)
  [OK] ServerLicense::Delegation::test-licenseServer-delegation-20200925_145658.xml: Found
valid server delegation: C:\Users\user.name\AppData\Local\Unity\licenses\delegations\test-
licenseServer-delegation-20200925_145658.xml
  [OK] ServerLicense::DelegationsCount: Found 1 valid server delegations
LicensingConfig: Healthy
Licensing configuration is healthy
  [OK] licensing: Found config section licensing
  [OK] licensing:licenseSigningCertLoading: Successfully loaded license signing key
Plugins: Healthy
Plugins
  [OK] PluginAssemblies: 0
  [OK] RegisteredPlugins: 0
Database: Healthy
Connection state for Microsoft.Data.Sqlite.SqliteConnection is Open
  [OK] ConnectionStatus: Connection state for Microsoft.Data.Sqlite.SqliteConnection is Open
Successfully started the licensing server: http://10.255.255.148:80/v1/admin/status
Press Ctrl+C to shut down...
```

Testing the license server

To test the server, open a console or terminal and navigate to the Unity Licensing Server launch directory. Run the following command and access one of the following two server endpoints:

Platform	Launch Command
Windows	`.\Unity.Licensing.Server.exe`
Linux	`./Unity.Licensing.Server`

- `http://SERVER-IP-ADDRESS:PORT/v1/status` provides basic information about the server,

including:

- serverStatus: Indicates if server is configured correctly and ready to serve licenses
- serverUpTime: Indicates the time different since server has been started to current date time
- serverUpTimeMs: Indicates the time different since server has been started to current date time (in milliseconds)
- version: Indicates the version of the license server.

  The output looks like this:

```
{
"serverStatus": "Unhealthy",
"serverUpTime": "0 days 1 hours 41 minutes 3 seconds",
"serverUpTimeMs": 6063668,
"version": "1.7.0.7cc2de3"
}
```

- `http://SERVER-IP-ADDRESS:PORT/v1/admin/status`: Provides additional information about the server, including machine bindings, license configuration, and database information. The output looks like this:

```
{
  "serverContext": {
     "data": {
           "FirstPhysicalAddress":  "48:2a:e3:87:3b:19",
           "Platform": "Win32NT",
           "ProcessorCount": "12",
           "MachineName": "LAPTOP-2Q28U8QR"
     },
     "status": "Healthy",
     "statusMessage": "Found 4 out of 4 context values"
},
"serverLicenses": {
  "data": [
      {
        "fileName": "C:\\Users\\user.name\\AppData\\Local\\Unity\\licenses\\test-
licenseServer_3573567292644_1-20200925_145658.xml",
        "status": "Ok",
        "message": "EntitlementGroup: test-licenseServer_3573567292644_1 (from: test-
licenseServer_3573567292644_1-20200925_145658.xml) has valid server entitlements",
        "licenseEntitlementGroupId": "test-licenseServer_3573567292644_1",
        "product": "Unity Enterprise for Product Lifecycle(Floating)",
        "licenseCount": 10,
        "validFrom": "2020-09-25T14:52:32.061Z",
        "validTo": "2022-09-24T00:00:00Z"
      }
  ],
  "status": "Healthy",
```

```
  "statusMessage": "Found 1 valid server license(s)"
},
"licenseSigningCertificateStatus": {
  "status": "Healthy",
  "statusMessage": "Successfully loaded license signing key"
},
"serverDelegations": {
  "data": [
      {
         "fileName": "test-licenseServer-delegation-20200925_145658.xml",
         "status": "Ok",
         "message": "Found valid server delegation:
C:\\Users\\user.name\\AppData\\Local\\Unity\\licenses\\delegations\\test-licenseServer-
delegation-20200925_145658.xml",
         "delegationId": "Delegation_test-licenseServer",
         "organizationId": "3573567206524",
         "serverId": "test-licenseServer",
         "delegationStart": "2020-09-25T14:52:32.061Z",
         "delegationEnd": "2022-09-24T00:00:00Z"
       }
   ],
   "status": "Healthy",
   "statusMessage": "Found 1 valid server delegations"
},
"databaseStatus": {
  "data": {
     "connectionStatus": 1,
     "connectionType": "Microsoft.Data.Sqlite.SqliteConnection"
   },
   "status": "Healthy",
   "statusMessage": "Connection state for Microsoft.Data.Sqlite.SqliteConnection is Open"
},
"plugins": {
  "data": [],
  "status": "Healthy",
  "statusMessage": "Plugins"
},
"serverStatus": "Healthy",
"serverUpTime": "0 days 0 hours 41 minutes 36 seconds",
"serverUpTimeMs": 2496435,
"version": "1.7.0.7cc2de3"
}
```


In either case, if the server status is shown as healthy (`"serverStatus": "Healthy"`), the server is usable.

Note: To access the `admin/status` endpoint, you must configure admin access for the machine you're using to test the server.

Support for multiple product licenses

The server can be configured with licenses for multiple different products. Each product is

represented by a license file with a specific set of entitlements. All these files are be available as an archive and can be imported using the Import command.

If multiple product licenses are available on the server, one license can be set as the default. Refer to the Advanced Server Configuration section for instructions.

The default license is used if no specific product license is requested by the client. Otherwise, the server provides a license for the requested product as long as a license is available.

Refer to the Copying the JSON section for instructions to configure required product licenses for the client.

Advanced Server Configuration

When the floating license server is successfully set up, the file `licensing-server-config.json` is generated in the LicensingServer directory. Refer to Server paths to find the location of this file.

You can edit `licensing-server-config.json` to modify your configuration. The following example includes all the possible configuration parameters:

```
{
    "Logging": {
        "LogLevel": {
            "Default": "Warning"
        }
    },
    "AllowedHosts": "*",
    "Kestrel": {
        "Endpoints": {
            "HTTPS": {
                "Url": "https://*:443",
                "Certificate": {
                    "Path": "[Full path of HTTPS PFX certificate - see Warning below]",
                    "Password": "[Password for HTTPS PFX certificate]"
                }
            }
        }
    },
    "licensing": {
        "serverId": "[Server name, to be set by setup CLI command]",
        "licenseUsernameObfuscation": false,
        "floatingLeaseRenewIntervalInMinutes": "15",
        "floatingLicenseExpirationInMinutes": "480",
        "licenseSigningCertificatePassword": "[Encrypted password of license signing
certificate]",
        "PluginsDirectory": "Plugins",
        "defaultToolset": "[default license entitlement group Id]"
    },
    "adminIpWhitelist": "127.0.0.1;10.20.30.40;localhost",
    "serverDirectory": "[Full directory path for Licensing Server executable]",
    "serverVersion": "[Licensing Server version]"
}
```

Note: On Windows, when using backslashes in a double-quoted string, you must use two backslashes for every instance of a backslash.

Advanced configuration keys include:

Key	Description
`floatingLeaseRenewIntervalInMinutes`	Sets how long the server holds a lease for the client. If the server does not receive a ping for a specific lease before the end of this interval, it will release the license associated to that lease. The default setting is 15 minutes.
`floatingLicenseExpirationInMinutes`	Sets the expiration time of the issued client license. The default setting is 8 hours.
`licenseSigningCertificatePassword`	The encrypted password of the signing certificate in in PFX format, set automatically by the Import CLI command.
`PluginsDirectory`	The full path of the directory the server reads plugins from.
`adminIpWhitelist`	Specifies the IP addresses that are allowed to access administrator endpoints as a semi-colon separated string. The licensing server does not support wildcard notation or subnets.
`defaultToolset`	Default product license identifier. All available product identifiers can be found on the server admin status page in `licenseEntitlementGroupId` field. If multiple product licenses are available on the server and client is not requesting any specific one, default one will be served. If default value is not set, the first available license from the list you see on admin status page will be served.

Server Paths

The location of the server files depends on the platform.

Server paths on Windows

Type	Path
Server log file	`%LOCALAPPDATA%\Unity\Unity.Licensing.Server.log`
Server setup log file	`%LOCALAPPDATA%\Unity\Unity.Licensing.ServerSetup.log`
Server import log file	`%LOCALAPPDATA%\Unity\Unity.Licensing.ServerImport.log`
Server create-service log file	`%LOCALAPPDATA%\Unity\Unity.Licensing.CreateService.log`
licensing-server-config.json file	`%LOCALAPPDATA%\Unity\LicensingServer\config\licensing-server-config.json`
Audit database file	`%LOCALAPPDATA%\Unity\LicensingServer\data\LicensingServerDb.sqlite`
Server license directory (current user)	`%LOCALAPPDATA%\Unity\licenses\`
Server license directory (all users)	`%PROGRAMDATA%\Unity\licenses\`
Server delegations directory	`%LOCALAPPDATA%\Unity\licenses\delegations`

Server paths on Linux

Type	Path
Server log file	`~/.config/unity3d/Unity/Unity.Licensing.Server.log`
Server setup log file	`~/.config/unity3d/Unity/Unity.Licensing.ServerSetup.log`
Server import log file	`~/.config/unity3d/Unity/Unity.Licensing.ServerImport.log`
Server create-service log file	`~/.config/unity3d/Unity/Unity.Licensing.CreateService.log`
licensing-server-config.json file	`~/.config/unity3d/Unity/LicensingServer/config/licensing-server-config.json`
Audit database file	`~/.config/unity3d/Unity/LicensingServer/data/LicensingServerDb.sqlite`
Server license directory (current user)	`~/.config/unity3d/Unity/licenses/`
Server license directory (all users)	`/usr/share/unity3d/licenses/`
Server delegations directory	`~/.config/unity3d/Unity/licenses/delegations`

Configuring the Unity Licensing Client

To configure the Unity Licensing Client on a user's machine, you need to set up both the the Services Configuration file (`services-config.json`) and the license client application (Unity.Licensing.Client).

Copying the JSON

Unity.Licensing.Client is configured in `services-config.json`. This file is automatically generated after running the setup command and is pre-populated with the following:

```
{
  "licensingServiceBaseUrl": "http://SERVER-IP-ADDRESS:PORT",
  "enableEntitlementLicensing": true,
  "enableFloatingApi": true,
  "clientConnectTimeoutSec": 5,
  "clientHandshakeTimeoutSec": 10
}
```

Supported configuration keys include:

Key	Description
licensingServiceBaseUrl	Network address of floating license server. Unity.Licensing.Client connects to different endpoints on this address to obtain a lease and a license file.
enableEntitlementLicensing	Enables entitlement licensing.
enableFloatingApi	Enables floating license API in client.
licenseClientApplicationPath	Full path of the Unity.Licensing.Client executable. Not required for the Hub from version 2.3.0 or for the Unity Editor from version 2019.2. See Setting the client application path for more information.
clientConnectTimeoutSec	Sets client connect timeout in seconds. 5.0 seconds by default. Can be between 0.5 and 20.0.
clientHandshakeTimeoutSec	Sets client handshake timeout in seconds. 2.0 seconds by default. Can be between 0.5 and 20.0.
clientResolveEntitlementsTimeoutSec	Sets resolving entitlements timeout in seconds. 2.0 seconds by default. Can be between 0.5 and 20.0.
clientUpdateLicenseTimeoutSec	Sets updating license timeout in seconds. 10.0 seconds by default. Can be between 0.5 and 20.0.
useLsd	Specifies which server endpoint should be called. Default value is True.
toolset	Specifies the product license you want to request from the floating server. If you want to request multiple product licenses, you can separate each product identifier with a comma. To see all available product identifiers, go to the Server Admin Status Page and see the list under `licenseEntitlementGroupId`. The server processes each license in the order you specify. If a license is not available, the server goes to the next license in the list. If you do not specify the toolset, the server serves the default toolset. For more information, see [Advanced Server Configuration](ServerSetup.md#advanced-server-configuration.

On each of the client machines that will run the Unity Editor, copy `services-config.json` to the path specified in the table below. If any folders in the path are missing, create them.

Platform	Services Configuration Path
Windows	`%PROGRAMDATA%\Unity\config\`
macOS	`/Library/Application Support/Unity/config/`
Linux	`/usr/share/unity3d/config/`

Finding the client application

Check the table below to see if your version of the Editor is bundled with the client application.

Unity Editor version	Bundled version of Unity.Licensing.Client
2019.4.7f1	1.6.0
2019.3	1.3.1
2019.2	1.2.0
2019.1	-
2018.4 LTS	-

If your version of the Unity Editor includes the client application, proceed to the next section.

If your version of the Editor does not include the client application, contact your Unity support representative to obtain the Unity.Licensing.Client archive in .zip format. When you receive the .zip, extract its contents to a persistent directory on all client machines.

Platform	Recommended path for client application
Windows	`C:\UnityLicensingClient`
macOS	`/home/<username>/UnityLicensingClient`
Linux	`/home/<username>/UnityLicensingClient`

Setting the client application path

If your version of the Editor does not include the client application or if your version of the Hub is 2.3.0 or

earlier, you must specify the full path of the Unity.Licensing.Client executable.

1. Open services-config.json in a text editor.
2. Add the key `"licenseClientApplicationPath"` and set its value to the full path of the Unity.Licensing.Client executable. Note: On Windows, when using backslashes in a double-quoted string, you must use two backslashes for every instance of a backslash.
    - If you manually extracted the Unity.Licensing.Client files, this is the path you specified in the previous section. If you used the recommended path, enter the value as shown in the following table.

Platform	License client application path (manual installation)
Windows	`"licenseClientApplicationPath": "C:\\UnityLicensingClient\\Unity.Licensing.Client.exe"`
macOS	`"licenseClientApplicationPath": "/home/<username>/UnityLicensingClient/Unity.Licensing.Client"`
Linux	`"licenseClientApplicationPath": "/home/<username>/UnityLicensingClient/Unity.Licensing.Client"`

    - If the client application is bundled with your version of the Editor, enter the full path of Unity.License.Client as shown in the following table:

Platform	License client application path (bundled installation)
Windows	`[Unity Editor Dir]\\Data\\Resources\\Licensing\\Client\\Unity.Licensing.Client.exe`
macOS	`[Unity Editor Dir]/Contents/Frameworks/Unity.Licensing.Client.app/Contents/Resources/Unity.Licensing.Client`
Linux	`[Unity Editor Dir]/Data/Resources/Licensing/Client/Unity.Licensing.Client`

Your services_config.json file now resembles the following example (on Windows, using the recommended path):

```
{
  "licensingServiceBaseUrl": "http://127.0.0.1:4433",
  "enableEntitlementLicensing": true,
  "enableFloatingApi": true,
  "licenseClientApplicationPath": "C:\\UnityLicensingClient\\Unity.Licensing.Client.exe"
}
```

Note: Be sure to add <,> to the end of the line before <"licenseClientApplicationPath">.

Validating the JSON

For best results, validate the .json file after each change. You can use any JSON validator or run a console command like the following:

```
python -m json.tool services-config.json
```

If validation succeeds, the console displays the contents of the .json file. If validation fails, the console displays an error.

Testing the license client

To test the client configuration, navigate to the directory where Unity.Licensing.Client is extracted and run the following command in the console or terminal:

Platform	Command to acquire a floating license
Windows	`.\Unity.Licensing.Client.exe --acquire-floating`
macOS	`./Unity.Licensing.Client --acquire-floating`
Linux	`./Unity.Licensing.Client --acquire-floating`

If the configuration is valid, the server assigns a floating license with a token identifier. The console output resembles the following example:

```
Trying to acquire floating license from: 10.1.34.126 ...
License lease Created with token e8b1afba-895d-4c54-aa50-5eadcc4d95a7. Expires: July 12, 2019 6:47:57 PM
```

Troubleshooting

Logs

The Unity floating licensing server and client generate several types of log file that you can use to diagnose problems. The tables below show where to find different types of logs.

Log paths on Windows

Type	Path
Unity Editor	`%LOCALAPPDATA%\Unity\Editor\`
Licensing Logs (Client, Server, Audits)	`%LOCALAPPDATA%\Unity\`
Unity Hub	`%APPDATA%\UnityHub\logs\`

Log paths on macOS

Type	Path
Unity Editor	`~/Library/Logs/Unity/`
Licensing Logs (Client, Server, Audits)	`~/Library/Logs/Unity/`
Unity Hub	`~/Library/Application\ Support/UnityHub/logs/`

Log paths on Linux

Type	Path
Unity Editor	`~/.config/unity3d/`
Licensing Logs (Client, Server, Audits)	`~/.config/unity3d/Unity/`
Unity Hub	`~/.config/UnityHub/logs/`

Licensing server report

There is a CLI command available for the licensing server to collect and archive licensing related data required for troubleshooting the licensing server.

You can generate and send the archive to Unity support for investigation.

Platform	Launch Command
Windows	`.\Unity.Licensing.Server.exe report --output-directory [directory path]`
Linux	`./Unity.Licensing.Server report --output-directory [directory path]`

`--output-directory` argument is optional. If you omit it, by default archive will be created in your current directory.

To generate the archive in the specific directory, replace `[directory path]` by the destination path.

If command executed successfully, you will see the generated archive path output.

Here is the sample output of a successful execution for `.\Unity.Licensing.Server.exe generate-report --output-directory=D:\` command:

```
Creating temporary folder...Done
Collecting configuration files...Done
Collecting license files...Done
Collecting delegation files...Done
Collecting log files...Done
Creating summary file...Done
Archiving files...Done

Archive successfully created: D:\Licensing.Server.Support.2020-06-12-025617.zip
```