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)

    Note: The Unity Licensing Server requires some additional software dependencies for Linux. For more information, see Additional software requirements for Linux below.

- 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.

Additional software requirements for Linux

This table lists the minimum library requirements to run the Unity Licensing Server on Linux:

Linux distribution:	.NET Core Runtime requirements:
Ubuntu 20.04	`libicu65`, `libssl1.1`
Ubuntu 18.04	`libicu60`, `openssl1.0`
Ubuntu 16.04	`libicu55`
CentOS 8	`libicu`
CentOS 7	`libicu`

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, Setting up 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.

Setting up the Unity Licensing Server

To set up and configure your own Unity Licensing Server, use the command-line interface (CLI), which provides the following functionality:

Command:	Syntax:	Description
setup	`Unity.Licensing.Server setup`	Configures server and manages server upgrades
import	`Unity.Licensing.Server import`	Imports license archive file
create-service	`Unity.Licensing.Server create-service`	Creates a service for your operating system
generate-report	`Unity.Licensing.Server generate-report`	Generates server troubleshooting data

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

- Linux: `./Unity.Licensing.Server --help`
- Windows: `.\Unity.Licensing.Server.exe --help`

Note: If you need to upgrade your server version, follow the instructions under Migrating your server to a new version. The migration process includes backing up the old server data in case you need to roll back the changes. For more information, see Restore database.

Warning: 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.

Important: Use a "service account" when you set up the server, not a "user account". The home directory of the user performing the setup determines the "owner" of the server, so using a neutral account makes it easier to maintain the server in the future.

Overview

To set up the Unity Licensing Server, you need to perform these tasks:

1. Unzip the `Unity.Licensing.Server.zip` file on a dedicated server machine. This becomes the launch directory for your Unity Licensing Server. For example, here are the suggested paths:
    - Linux: `./Unity.Licensing.Server setup`
    - Windows: `.\Unity.Licensing.Server.exe setup`
2. Run the setup command to store the license server configuration parameters in the `licensing-server-config.json` file and generate a `server-registration-request.xml` file to upload to the Unity ID portal page.
3. Run the import command on the license archive file you downloaded from the Unity ID portal page.
4. Run the create-service command to create a service that launches the license server automatically.
5. Test the license server.

If you run into any problems after setting up the licensing server, you can run the generate-report command, which collects and archives data you need to troubleshoot.

Configuring the license server

The setup command creates the `licensing-server-config.json` file and populates it with the license server configuration parameters you specify. See Server paths section to find where you can these files on Linux and Windows servers. If you already have a `licensing-server-config.json` file, the setup command uses the existing configuration parameters as default values.

Before you begin, make sure you prepare the following:

- Decide which protocol (HTTPS or SSL) your licensing server will use.
- If you use the HTTPS protocol, make sure you have a server certificate in the Personal Information Exchange Format (`.pfx`) format. Unity does not provide this certificate: your IT team is responsible for managing this certificate.
- Configure the firewall for your network interface to allow `Unity.Licensing.Client` to obtain a license.
- Determine which port you want to allow client connections. Make sure that the port you choose isn't already being used by another application. If you run the setup command and choose a port that isn't free, you have to restart the setup again from the beginning and select a different port.
- You can configure administrator access for specific IP addresses in order to diagnose and troubleshoot common configuration problems. However, you need to add each admin IP address to the Admin IP Whitelist to grant access to the admin API. To access the admin API, go to: `http://<SERVER-IP-ADDRESS>:<PORT>/v1/admin/status`.

The setup command also creates the following files:

- The Services Configuration (`services-config.json`) file. Re-deploy this file on all end-user computers every time you execute the setup command in order for the computers to use the new configuration.

- The Server Registration Request (`server-registration-request.xml`) file containing machine binding information. Upload this file to the Unity ID portal page to generate the server license. This file includes the following information:

Element:	Description:
`"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

Running the setup command

To start the interactive setup from the command line follow these steps:

1. From the Linux Terminal or the Windows PowerShell, run the command that matches your operating system:
    - Linux: `./Unity.Licensing.Server setup`
    - Windows: `.\Unity.Licensing.Server.exe setup`

      A welcome message appears in the Terminal or PowerShell window:

      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. Enter a name for your server.
3. Set which protocol (HTTPS or SSL) your licensing server will use:

    ```
    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":
    ```

    - If you don't have a server certificate and want to skip this step, enter `n`.
    - If you have a server certificate, enter `y`, and then set the path and password of the `.pfx` file where indicated.
4. Enter the index number of the network interface to select which interface the license server will provide service on:

    ```
    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:
    ```

    Note: This interface should have its firewalls configured so that `Unity.Licensing.Client` can communicate to obtain a license.
5. Enter the port number where you want the clients to connect. To avoid port conflicts, use a

port outside the known range (with a 4- or 5-digit number):

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

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

6. Configure administrator access. The license server can limit administrator access to admin endpoints. These endpoints contain detailed server information and are useful for diagnosing and troubleshooting common configuration problems.

   To allow access to the server from inside the server machine itself and from the publicly reachable default IP address, enter `y`. Otherwise, enter `n` 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`, then enter the new IP to be added to the whitelist. Otherwise, enter `n`.

   When the setup is completed successfully, the Terminal or PowerShell window displays something 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
```

7. 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.

Downloading and importing the license archive file

Important: Use the same (service) account to complete all of the following steps now and for any future maintenance or updates.

The import command does the following:

- Imports the license archive file.
- Stores the server license file into the current user's platform-specific directory for the license server.
- Moves the signing delegation files into the platform-specific directory.
- Sets the license signing certificate password in the `licensing-server-config.json` file.

Tip: See Server Paths for more information about where the license server files are stored.

Before you can run the import command, you need to download the license archive file from your Unity ID portal page:

1. Download the license archive (`.zip` file) from the Unity ID portal page.
2. From the Linux Terminal or the Windows PowerShell, run the command that matches your operating system to import the archive:
    - Linux: `./Unity.Licensing.Server import <path-to-downloaded-archive>`
    - Windows: `.\Unity.Licensing.Server.exe import <path-to-downloaded-archive>`

      A list of all available toolsets for this license server appears in the Terminal or PowerShell window:

      ```
      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
      ```

3. Enter the toolset ID you want to set as the default. For more information about toolsets, see Advanced Server Configuration.

   The command writes this ID to the `licensing-server-config.json` file and displays the following message in the Terminal or PowerShell window:

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

Note: The license server is now fully configured to run and issue licenses to clients, but you need to create a service. To launch the license server automatically whenever the server boots, create an auto-start service (recommended).

4. Restart the server to reload the new license configuration files.

   Tip: If the license server is running as a service, follow the instructions for restarting the service.

You can now proceed to the next step: Testing the license server.

License archive contents

The `.zip` includes the following files:

File	Description
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.

Setting up a service for the license server

The create-service command sets up a service on your operating system to run the server process. You can configure the service to start automatically on boot. You can also manually pause, stop, and restart the process.

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

Creating a service

Important: Use the same (service) account that you used to import the license archive file.

To create a service for stopping and starting the license server, follow the instructions based on your operating system:

- Linux
- Windows

Creating a service on Windows

The create-service command requires administrator privileges. Before you start, make sure you start the PowerShell window with elevated privileges. You must use an administrator account with a non-blank password.

1. Give the user the right to log on as a service. To assign this right:
    - Search for `Local Security Policy` in Windows Search and open the policy editor.
    - In the left-hand menu, navigate to Security Settings > Local Policies > User Rights Assignment and select Log on as service.
    - Click Add User or Group.
    - Type the username in the text box and click Check Names > Ok.
    - Click OK.
2. To create the service, run the following command in the PowerShell window (started with elevated privileges):

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

3. Enter the password of the admin user used to create the service. The create-service command outputs something like this:

    ```
    - - - -
    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
    ```

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

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

   Look for the `STATE` information in the output in the PowerShell window:

   ```
   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
   ```

If the `STATE` value shows anything other than `4 RUNNING`, open Windows Services and look for `Unity.Licensing.Server` the service to manage it.

Creating a service on Linux

The create-service command does the following:

- Creates a `systemd` service unit called `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.

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

1. To create the service, run the following command:

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

   You should see output similar to the following:

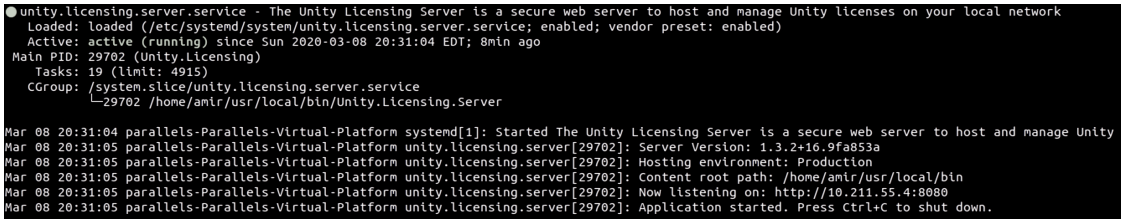
   ```
   - - - -
   Welcome to Unity Licensing Server create service command line interface.
   This command line interface will help you create Unity.Licensing.Server service.
   - - - -

   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.
   ```

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

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

Look for the `Active:` information in the output in the Terminal:



Restarting the service

You can stop or restart the service using standard system commands. Follow the instructions based on your operating system:

- Linux
- Windows

Note: This operation interrupts the license server's availability. Make sure you perform this operation when traffic is typically low or during your organization's maintenance period.

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 status command shows something like this:

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

Restarting the service on Windows

1. Run the following command to start (or restart) the service:

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

2. If you see `STATE : 4 RUNNING` in the output, run the following command to stop the service:

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

3. If you see `STATE : 1 STOPPED` in the output, run the following command start the service again:

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

4. To check the state of the service again, run the following command:

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

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

Deleting the service

You can delete the service if you need to. Follow the instructions based on your operating system:

- Linux
- Windows

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 status command outputs something like this:

   ```
   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
   ```

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
```

3. If you see `STATE : 1 STOPPED` in the output, 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
   ```

Back up server configuration

To back up the existing configuration of the server, save a copy of the `LicensingServer` folder in a safe location. The location of the folder to back up differs based on your operating system:

- Linux: `~/.config/unity3d/Unity/LicensingServer/`
- Windows: `%LOCALAPPDATA%\Unity\LicensingServer\`

Restore server configuration

To restore the server configuration from the backup, replace the `LicensingServer` folder with the version you backed up.

Generating a report archive for troubleshooting

The generate-report command collects and archives licensing related data required for troubleshooting the licensing server.

Tip: By default, the command writes the report archive to your current directory, but you can optionally use the `--output-directory` argument followed by the destination path to specify a different location.

From the Linux Terminal or the Windows PowerShell, run the command that matches your operating system:

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

If command executed successfully, it displays the generated archive path output. For example, the command outputs something like the following after you run `.\Unity.Licensing.Server.exe generate-report --output-directory=D:\` in the PowerShell window:

```
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
```

Checking the status of your server

The status command can perform the following health checks and display the results:

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:

- Linux: `./Unity.Licensing.Server status`
- Windows: `.\Unity.Licensing.Server.exe status`

The output looks 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

The testing output contains two levels of feedback:

- Basic: status, time since last started, and server version
- Extended: machine bindings, license configuration, and database information

To test the server:

1. From a PowerShell or Terminal window, navigate to the Unity Licensing Server launch directory and run the following command:
    - Linux: `./Unity.Licensing.Server`
    - Windows: `.\Unity.Licensing.Server.exe`
2. Access one of the following two server endpoints:
    - For Basic: `http://<SERVER-IP-ADDRESS>:<PORT>/v1/status`
    - For Extended: `http://SERVER-IP-ADDRESS:PORT/v1/admin/status`

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

The server is usable when the `serverStatus` information appears as Healthy.

Support for multiple product licenses

You can configure the server with licenses for multiple different products. Each product is represented by a license file with a specific set of entitlements. All these files are available as an archive and you can import them using the import command.

If multiple product licenses are available on the server, you can set one license as the default. For more information, see Advanced Server Configuration.

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.

For instructions on configuring the required product licenses for the client, see Copying the JSON.

Basic test report

Open `http://<SERVER-IP-ADDRESS>:<PORT>/v1/status` to see basic information about the server, including:

Entry:	Description:
`serverStatus`	Whether the server is configured correctly and ready to serve licenses
`serverUpTime`	The amount of the time between when the server started and the current date time
`serverUpTimeMs`	The amount of the time in milliseconds between when the server started and the current date time
`version`	The version of the license server

For example, the output looks similar to this:

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

Extended test report

Open `http://SERVER-IP-ADDRESS:PORT/v1/admin/status` to see the basic report, plus additional information about the server, including machine bindings, license configuration, and database information.

The output looks similar to 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"
```

```
        "delegationStart": "2020-09-25T14:32:32.001Z",
        "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"
}
```

Migrating your server to a new version

Newer server versions can often require a different server database or configuration (for example, if you have version 1.6 installed and you want to run version 1.9). If your server detects that it is running a version of the Unity Licensing Server that is incompatible with the new version, it displays the following message and then stops:

```
Cannot start the licensing server. Incompatible version is already installed. Please, make
sure to stop existing Licensing.Server service (if it is running) and run setup command.
```

To resolve this incompatibility, run the command that matches your operating system from the Linux Terminal or the Windows PowerShell:

- Linux: `./Unity.Licensing.Server setup`
- Windows: `.\Unity.Licensing.Server.exe setup`

```
- - -
Welcome to Unity Licensing Server setup command line interface.
This setup will help you configure your license server and generate server registration
request file.
- - -
Upgrading from 1.6.0 to 1.9.0...
Backup installed version 1.6.0...Done
Scripts to run: 1.9.0
Executing 1.9.0...Done
Stamping version 1.9.0...Done
Successfully migrated from 1.6.0 to 1.9.0
```

The setup command detects your current server version and copies your server data to the `LicensingServerBackups` directory. The location of the pre-migration backup data is different for Linux and Windows:

- Linux: `~/.config/unity3d/Unity/LicensingServerBackups`
- Windows: `%LOCALAPPDATA%\Unity\LicensingServerBackups`

Then it proceeds to convert your data to be compatible with the new server version. When the migration is complete, it begins the configuration process (the questions you already answered in the previous setup of server). However, since you already configured your server and successfully migrated your settings, you can use the Ctrl+C keyboard shortcut to exit the command.

Note: If the command encounters any problems during the migration, it cancels the migration and restores your data to its pre-migration state.

Restore database

When you upgrade (migrate) your server to a new version, the setup command automatically creates a backup copy of your database file. If the restoration fails, the following critical error message appears:

```
An error occurred while restoring a backup.
```

This indicates that your server is in an invalid state which might prevent you from continuing to use your previous server version. To fix this, copy the contents of the `LicensingServerBackups/<your_old_server_version>/` directory to the `LicensingServer` directory to manually restore the backup data. For example:

```
├── LicensingServer
│   ├── config
│   │   ├── key-b4573620-ca22-423f-8d52-e3a767391234.xml
│   │   ├── licensing-server-config.json
│   │   └── unity.licensing.delegation.key
│   ├── data
│   └── wwwroot\v1\package\acl
│
├── LicensingServerBackups
│   └── 1.6.0
│       ├── config
│       ├── data
│       └── wwwroot
```

To roll back the server to a previous version, follow these steps:

1. Put the existing database file into the following folder:
    - Linux: `~/.config/unity3d/Unity/LicensingServer/data/`
    - Windows: `%LOCALAPPDATA%\Unity\LicensingServer\data\`
2. Rename the file to `LicensingServerDb.sqlite`.

Advanced server configuration

When you have successfully set up your floating license server, you can find the `licensing-server-config.json` file in the LicensingServer directory.

To modify your configuration, edit the `licensing-server-config.json` file. 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

Advanced configuration keys include:

Key	Description
`floatingLeaseRenewIntervalInMinutes`	Specifies 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`	Specifies 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. The import command automatically sets this value.
`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`	Stores the identifier of the default product license. If multiple product licenses are available on the server and the client does not request any specific one, the default license is served. If the default value is not set, the first available license from the list is served. To find all available product identifiers, open the server admin status page and locate the `licenseEntitlementGroupId` field.

Server paths

The location of the server files depends on the platform.

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 data root directory	`~/.config/unity3d/Unity/LicensingServer/`
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/`

Type:	Path:
Server license directory (all users)	`/usr/share/unity3d/licenses/`
Server delegations directory	`~/.config/unity3d/Unity/licenses/delegations`
Pre-migration Server backup	`~/.config/unity3d/Unity/LicensingServerBackups`

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 data root directory	`%LOCALAPPDATA%\Unity\LicensingServer\`
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`
Pre-migration Server backup	`%LOCALAPPDATA%\Unity\LicensingServerBackups`

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,
  "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
enableEntitlementLicensing	Enables entitlement licensing.
enableFloatingApi	Enables floating license API in client (only set this flag if you are using Unity.Licensing.Client v1.3.0 or earlier).
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.

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
2020.2.1f1	1.6.0
2020.1.17f1	1.6.0
2019.4.17f1 LTS	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,
  "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
```

Licensing client exit codes

The Licensing Client exit codes indicate successful or failed execution of a command. They could be used in CLI scripts to control execution flow.

Exit code	Status
0	OK
1	Invalid arguments
2	Invalid credentials
3	Organization ID is missing
4	Package ACL file download failed
5	Context initialization failed
6	Replication service initialization failed
7	Orchestrator initialization failed
8	Floating service initialization failed
9	Package service initialization failed
10	Access Token initialization failed
11	Multi Client Pipe Server start failed
12	License activation generation failed
13	Syncing entitlements failed
14	No valid entitlement found
15	License update failed
16	Unable to get list of user seats

Exit code	Status
17	Seats activation/deactivation failed
18	Getting entitlements failed
19	Acquiring license failed
20	Renewing floating lease failed
21	Returning floating lease failed
1000	UnknownError