Collaborate
Expert Guide

Hollie Figueroa

Getting Started

Collaborate is a cloud-based, client-server pairing that exists between the Unity Editor (client) and the Unity-owned servers. A Collaborate **Project** is assigned a unique Unity Project Identifier (**UPID**) to link the local Project to Unity servers. Each Project **Member** must be linked to this specific Project UPID to Collaborate. A member often makes changes to multiple files at the same time before making a single **Publish** to the Unity servers, creating a new **Revision** of each file. The **Collab Toolbar** notifies other Project members that an **Update** is pending. Unlike more traditional Source Control Management systems (SCM), Collaborate is base-less, meaning each Update downloads all the pending updates into the local Project. If there is a **Merge Conflict**, the updating member has the opportunity to select a change from the Collab Toolbar using **Choose mine** (edit the file to use the member's local changes) or **Choose theirs** (keep the file as is.) Selecting **Choose mine** requires that the member Publish the new change. If a member needs a closer look at the changes before making a decision, the **See differences** or **Launch external tool** options allow members to view the changes in a file comparison tool. For a more detailed look at each Revision or the current state of the Project on the Unity servers, members can visit the Collaborate Project **Dashboard** on the web. Members will also find Project management tools and Collaborate Support links.

The following sections go into more detail about all of the above. There is also a handy "cheat sheet" at the bottom of what the expected outcome of different types of merges should be. Have fun Collaborate-ing!

Members Grid

Project members can have different roles and access rights, giving teams complete control over access to the Project.

Note: Organization permissions always override Project permissions.

Permission Level	Create Project	Access In The Cloud	Add/Remove Members	Alter Unity Services Settings	Alter Billing Information
Owner	**x**	**x**	**x**	**x**	**x**
Manager		**x**	**x**	**x**	
User		**x**			

Projects

- Create
    - Projects can enter Collaborate as brand new with no member-created assets, as robust Unity Projects nearing completion, or anything in-between.
    - After enabling, Collaborate automatically starts a Publish of the Project Settings files and the default *.collabignore* file.
    - The Project member can then Publish any remaining files in the Assets folder.
    - Other Project members are able to download the Project from the cloud.

> **TIP:** A good rule of thumb is to fix any compiling errors, then save the Project locally before every Publish.

- In The Cloud
    - If the Project does not exist locally, any Project member can download the Project from the Unity servers.
    - From the Unity Editor Launcher window, select **In The Cloud** from the left column. Projects currently in the cloud are listed alphabetically.
    - Projects that do not exist on the local machine are displayed in black. Projects that already exist locally and are also stored on the Unity servers are listed at the bottom in gray.
    - Once a Project is downloaded locally, Project members no longer need to download it from In The Cloud. The local copy syncs with every Publish and Update.

> **TIP:** Members can re-download a Project from In The Cloud at any time! Store a backup locally in a different folder for peace of mind or experimentation.

- Publish
    - A member ready to sync changes from the local copy to the Unity servers can access the Publish button in the Collab Toolbar.
    - Members should make smaller, frequent commits to protect the Project from networking errors and large merge issues.
    - Publish with confidence. Canceling a Publish can introduce errors in the Project and History.
    - Publish code that compiles without errors. This ensures all scripts will sync properly to the Unity servers and keeps other members safe from updating to script errors.
    - Be mindful of Unity's handling of special folders, script execution order, and script serialization rules when creating or modifying plugins and c# reference scripts.

- If a large Publish is necessary:
  - Keep the network stable. Collaborate is an “always connected” service, but this is extremely important during a large publish.
  - If the number of changes is over 10,000 (including .meta files), a publish is split into smaller Publish “groups” automatically
  - A large publish split into smaller groups can only be canceled before the first group has been sent to the Unity servers.

TIP: A “large” publish can refer to both file size and the number of files being published. If a large file (size) is ready to be published, limit the number of changes published with it.

- Collaborate offers a **Partial Publish** feature to facilitate small, frequent publishes. Members are able to access this feature using the right-click menu in the Projects window or via the Assets drop-down and navigating to Collaborate>Publish.

A note about .meta files:

Unity uses .meta files to keep track of an asset’s settings, references, relationships, and configurations. Collaborate hides Unity .meta files in the list of changes to simplify the view, but they still get published and should be treated with care. Metafiles must *always* stay with their asset file.

- Update
  - After a publish, the Collab button will display the orange Update arrow for other team members signaling them to sync their local copies.

    - Collaborate is built on the principle of centralized development. Merging happens implicitly every Update.
    - Collaborate incorporates a tool called UnityYAMLMerge that can merge .scene and .prefab files in a semantically correct way.
    - Collaborate requires members to be Up To Date before they can publish. Each member should update often to prevent any publishing delays.
    - Staying up to date prevents errors that can occur for members that are far behind in revision History.
    - The Collab History window is a great resource for team members to view incoming changes in the Update.
- Unity upgrades
    - Upgrading to a new version of Unity always comes with an inherent risk. It is important to reference the Upgrade guides in any Project upgrade with or without Collaborate.
    - Before a Project member opens a Collaborate Project in a new version of Unity, it is recommended to have a backup on the local disc. This can be a second version of the Project downloaded from “In The Cloud” or an exported unitypackage.
    - One Project member should be the “upgrade master” responsible for when the Project is ready to upgrade. The upgrade master should open the Project in the new version of Unity, run the API Updater, and confirm the Project’s integrity **before** publishing the version changes.
    - If the Project upgrade is a major version change (ex. Unity 5.5 > Unity 5.6), it is recommended for other Project members to download the upgraded Project from “In The Cloud” and open in the upgraded version of Unity.
- A note about third party DLL's
    - Any third party code should always be quality evaluated before publishing as part of the Project to the Unity servers.
    - Project members should be mindful of any version upgrades to the DLL's that may cause compiling errors or issues for other Project members. (ex. Upgrading an asset store plugin while it is open and in use in the Unity Editor can cause compiling errors.)

TIP: The Unity Editor does not currently warn when Projects downloaded from In The Cloud have been edited in a different version. Make sure Projects are opened in the appropriate version.

Always Connected

Collaborate is cloud-based and requires a strong, stable connection to function properly and ensure Project members maintain access to all of Collaborate's team features. Collaborate must check for revisions often for members to keep the local copy in sync with the Unity servers.

Collaborate has a separate Network Reporter tool for Project members to use in order to help identify more precisely what is happening on the network and if there is an issue. Visit this link to see the instructions on how to download and use the tool.

Collab Ignore

Collaborate uses *.collabignore* to define files intentionally not synced. These files are *ignored* by Collaborate in the same way Git ignores those files specified in a *gitignore* file.

- Certain files and directories are ignored by default while others can never be ignored. All team members should be familiar with these settings by visiting the Collaborate Ignore section of the Unity Manual.
- Assets can be added to the *.collabignore* file at any time. After making local edits to the *.collabignore* file, members will need to restart the Unity Editor for the change to take effect.
- Situations to consider adding an asset to the ignore file include:
    - Recomputed assets. If a material, shader, mesh, or lighting is recomputed every time the scene enters Play mode, teams should consider adding the asset to the *.collabignore* file after the initial sync.

    - Assets that are 100% complete. This can be an easy way to “lock” an asset and prevent accidental changes to a completed task.
    - Large textures. After an initial sync, add large textures to the *.collabignore* file to reduce Update times.
- Members can use the “All Excluded” filter in the Project window to quickly see which assets have been added to the *.collabignore* file. Note: The Project window must be set to the “two-column” layout in order to see filters.

Go back to..., Restore, and Revert

Collaborate offers three powerful tools to assist members in controlling the “current” revision of a Project. **Go back to...** and **Restore** in the Collab History window provide access to previous revisions, while **Revert** in the Collab Toolbar and Asset menu allows members to access individual local file changes. Understanding a few key principles can help members use these tools effectively.

- Go back to...
    - While the action is referred to as “Go back” or “Rollback”, Collaborate actually implements forward linear development timeline. Rather than actually “going back” the changes in the revision selected to “rollback” are brought to the head of the Project. Therefore “reverting” any changes will merely revert it back to the most forward state.
    - Because Collaborate is “forward only” development, it can be difficult to remember what revision members have chosen to “roll back” to. When using the “Go back to...” feature, be sure to remember what revision number you selected. The revision number can be found in small blue print immediately to the left of the revision text.
    - To avoid merge conflicts, all team members should make sure they are up to date and there are no pending changes before using “Go back to...”
    - Members should remember, the “rollback” changes are treated as a Publish and are seen as local (unpublished) changes until then.

- Restore
    - Restore will discard all unpublished changes on every file. There is no new Publish needed. Restore is a useful tool for “resetting” a local repository back to the last synced revision.
- Revert
    - After making any change to an asset, members are able to use the Revert button from the Collab Toolbar to revert an asset back to its state on the Unity servers. This is incredibly useful for prototyping, balancing, and experimenting with an asset’s settings.
    - Because revert brings an asset back to its state on the Unity server, new files that have recently been added or created cannot be reverted. However, recently deleted assets can be reverted if the delete has not been published.
    - After a merge conflict resolution, if a member has selected “Choose mine”, revert will bring the asset to the “Choose theirs” state. It is important to Publish any conflict resolution changes immediately before continuing work.
    - Revert is not available in the Collab Toolbar for individual files if an Update is pending. Members can still access Revert by using the Assets>Collaborate menu.

Parallel development within the Project

Collaborate defines parallel development as two or more members working on a single Project, at the same time. While parallel development can be a very rapid, positive solution for teams, there are some considerations when working with Collaborate.

- Scenes
    - It is always a suggestion to work in separate scenes whenever possible.
    - Sometimes it is beneficial or even critical for members to work in the same scene together, Collaborate uses Unity’s YAMLMerge tool to smartly merge changes. The Smart Merge tool uses the premerge parameters.
    - It is always recommended to confirm the merge was successful and as expected before continuing work.

- In-Progress
    - Members can use the In-Progress feature of Collaborate to know when other members have unpublished changes on a Scene or Prefab.
    - A small icon will appear on the Scene asset in the Project window to notify other members someone is working on the Scene.
    - Members can also manage In-Progress assets via the Collaborate dashboard. Members can easily see who is currently editing and which Scenes and Prefabs are being worked on.
    - In an open scene, the icon also appears at the top of the Hierarchy window. Members can hover over the icon to display the usernames of the member or members with unpublished changes.
    - Prefabs will also show a small icon on the asset in the Project window and in the Inspector.
    - Prefab instances will not display the icon as they are now Game Objects in a scene and not the Prefab asset.
- Partial Publish
    - Partial Publish provides members with a safe alternative to working in a scene or prefab with other members outstanding changes. With partial publish, members are able to publish an individual or small selection of changes, such as those to a scene they are done working in, without publishing everything.

Problems solved with Parallel development:

Teams may choose parallel development for the bulk of their development cycle. Often members do not need to work in the same space, even within a single Project, and work can continue on multiple features at the same time.

TIP: Communication is key in any Project with any team, but even more so with parallel development.

Centralized development as a service

Collaborate is founded on the principle of centralized development. In other words, it is based on the idea that there is a single “central” copy of the Project (on the Unity servers) and members publish changes to this copy. This simplifies version control and workflow to three basic steps:

- Update to any changes that exist pending on the Unity servers
- Make changes locally
- Publish changes back to the central copy for other members to see

It is important to point out, merging happens implicitly every Update. This is critical to remember when working with certain asset types and multiple Project members.

- Scenes and Best practices
    - When making changes to the same scene at the same time, it is absolutely critical for members to communicate. Members should use In-Progress and Publish messages in combination with team communication.
    - Keep Scene organization simple and easy to adhere to. Members will easily be able to spot oddities or unwanted merging if the Scene organization is simple.
- Prefabs and Best practices
    - Making concurrent changes to Prefabs should follow the same safety guidelines as working with scenes.
    - Members are encouraged to create new specialized Prefabs rather than specializing prefab instances. This practice will allow for In-Progress to track any uncommitted changes from other Project members.
- Partial Publish
    - Members can use this feature to avoid large publishes and keep the Project as up to date as possible. Keep in mind, each publish should be a self-contained unit.

Project and team size considerations

Project size 10-15 GB
Team size ~ 7 people

Merge Matrix

Move or Rename/Edit	User 1 moves or renames the asset. User 2 edits the asset. After User 2 updates the move should happen successfully and there should be a pending Publish with the edit.
Edit/Delete	User 1 edits the asset. User 2 deletes the asset. The result will be a merge conflict. The options will be to Choose theirs (the edits) or Choose mine (the delete).
Move or Rename/Delete	User 1 moves or renames the asset. User 2 deletes the asset. Current behavior, although not ideal, the move will be processed at the Update. There will be a pending Publish with an Add/Delete. This is the metafile resolving. **Note**: Because this process has a delay in synchronizing the relevant .meta with its asset file. It is a risky behavior and should be avoided.
Divergent move	User 1 moves or renames the asset. User 2 moves or renames the asset to somewhere or something else. Current behavior will auto-resolve to select User 1's rename/move.
Cycle move	User 1 moves Assets/folder1 to Assets/folder2/folder1 User 2 moves Assets/folder2 to Assets/folder1/folder 2 Current behavior will auto-resolve this to Assets/folder2/folder1/folder2. To correct the hierarchy, move the nested folder1/folder2 to the root Assets/ and delete the Assets/folder2
Move/add	User 1 moves assetA to folder1. User 2 adds assetA to folder1. Current behavior will create a "file missing" error. Dismissing the error you can Choose mine or Choose theirs, but this will overwrite the other asset.
Added evil twin	User 1 moves an asset to a specific location. User 2 add an asset of the same name to that same location. Collaborate currently does not support this and one machine will be out of sync with the server repro. Please contact support for this scenario.

Collaborate LOVES inch-pebble development!

Check-in Project changes very often to Collaborate. Keep each publish a self-contained small and easy to understand set of changes (hence inch and pebble!).

Collaborate best practices

- Pick a naming convention and stick to it.
    - Avoid changing file or directory names if possible.
    - If members need to rename a folder or file, be sure the asset is up to date and then rename the file as a separate publish. Be sure all Project members update the change before working on the asset.
- Pick a Project structure and stick to it.
    - Keep it simple and keep it consistent.
    - Projects, Scenes, and Scripts should all follow a similar structure. This consistency makes it easier to spot any errors in your Project.
    - If possible, avoid moving folders and files, especially those with large contents or dependencies.
    - If members need to move a folder containing assets, have each member publish all changes and update before the re-structure is published. Doing so reduces the risk of merge conflicts. Before continuing work, be sure that all users are up to date after the change before continuing work.
- Use descriptive, logical Publish messages.
    - Each Publish message should exactly describe what is in each Publish. Smaller, concise publishes can have smaller, concise messages.
- Use each Publish as a sensible unit.
    - This promotes thoughtful, agile development.
    - Small unit publishes set up clear milestone and task tracking.
    - Test planning and implementation become faster and easier.
- Stay on top of Updates.
    - Updates tend to be riskier if the local repository is behind or the number of changes is large.
- Publish your changes frequently.

- Be mindful of generated asset types. They might be good candidates to include in the *.collabignore* file.
    - This includes lighting. Turn auto-generation off and have one team member responsible for light baking.
- Be mindful of OS differences
    - Some operating systems are case-sensitive while others are not. In your naming scheme, keep this in mind if Project members are working on different operating systems.
- Pick a merge tool and stick to it as a team.
    - Collaborate works with external merge tools to resolve merge conflicts. Each merge tool treats certain cases (ie. line number differences) differently. Project members should use the same merge tool to avoid inconsistencies. Collaborate integrates well with Apple File Merge, Beyond Compare 4, SourceGear DiffMerge, TkDiff, and WinMerge.

Current Limitations and remarks

- The Collaborate web dashboard is a fantastic tool. Access it via the "Go to Dashboard" link in the Services window and use it frequently.
    - See your publish timeline and which files were changed when.
    - Download individual assets in their previous versions.
    - See the Project's current storage usage.
    - Quickly see and manage the In-Progress assets
    - Easily get in touch with the Collaborate Support team if you run into issues.
- Collab Event Logs
    - o Enable extra logging either for debugging purposes, troubleshooting, or reporting a bug. Launch Unity from command line with the argument `-enableCollabEventLogs`
- How to submit a bug
    - In the Unity Editor, select **Help** > **Report a Bug**. It also launches automatically if you experience a crash.