Home

Thank you for using Advanced Instance Manager!

Advanced Instance Manager is a plugin that allows you to test multiplayer games in Unity with ease.

Contact: Want to get into contact with us? Report issues?
Github issues
support@lazy.solutions
discord.gg/pnRn6zeFEJ >Note: We accept bug reports and offer help on all three channels. Which one you wish to use, is up to your preference!

Guides: Quick Start - An introductory guide to using Advanced Instance Manager
Instance Manager Window - A guide that describes how to use the instance manager window

API: Core:
InstanceManager - The core of the API
UnityInstance - The model for secondary instances
Layout - The model for window layouts

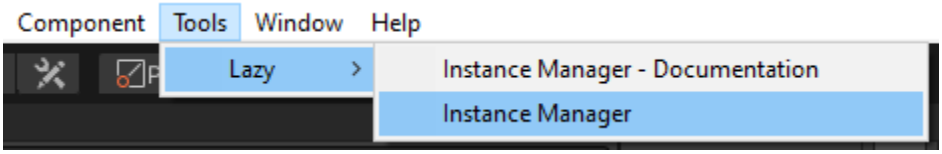
Utility:
ActionUtility - Provides functions for running tasks
CommandUtility - Provides functions for running commands in the terminal
CrossProcessEventUtility - Provides the ability to send 'events' to secondary instances and back
GUIExt - Provides a few extensions for IMGUI
InstanceUtility - Provides functions for working with UnityInstance
ProgressUtility - Provides functions for running tasks that should report indeterminate progress in editor
WindowLayoutUtility - Provides functionality for working window layouts for the editor

QuickStart

Welcome to Advanced Instance Manager!
The interface of Advanced Instance Manager is designed to be as intuitive as possible, and while we hope that this guide is not really needed, it is obviously good to have one regardless.

So here we go, lets create and run a secondary instance!

- Open instance manager window:



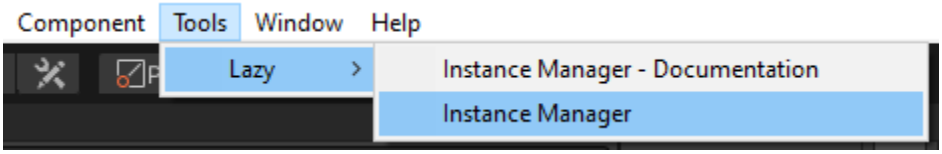
- Press 'Create new instance' button, to create a new instance
  Wait while instance is being created, progress is displayed in the lower right corner (unity 2020 and above). When instance is done being created it will appear in the list.
- Press the menu button (or right click) on the instance in the list, and press 'Options...'.
  Here you may change any settings you wish, refer to Instance Manager Window for more explanations.
  Go back using the '←' button in the upper right corner when done.
- Press 'Open' on the instance in the list, the status should change to 'Running'.
  Wait until new instance opens (it should open up on top, but may obviously appear behind primary instance / other windows if you're clicking around, so check taskbar too).
- Once the secondary unity instance has opened, the window may flash due too changing the window layout, this is normal and not really fixable. The Instance Manager window will automatically open, with the options screen for this instance visible (but with a few differences to account for actually being inside the instance).
- Enter playmode in primary instance, assuming you did not disable 'Enter / exit playmode when primary does' in options, you should now see the secondary instance also entering playmode.
  Pressing pause in primary instance will also pause in secondary (this is controlled by enter playmode setting, by the way). Unpausing or exiting playmode in primary instance should of course also do the same in the secondary instance.

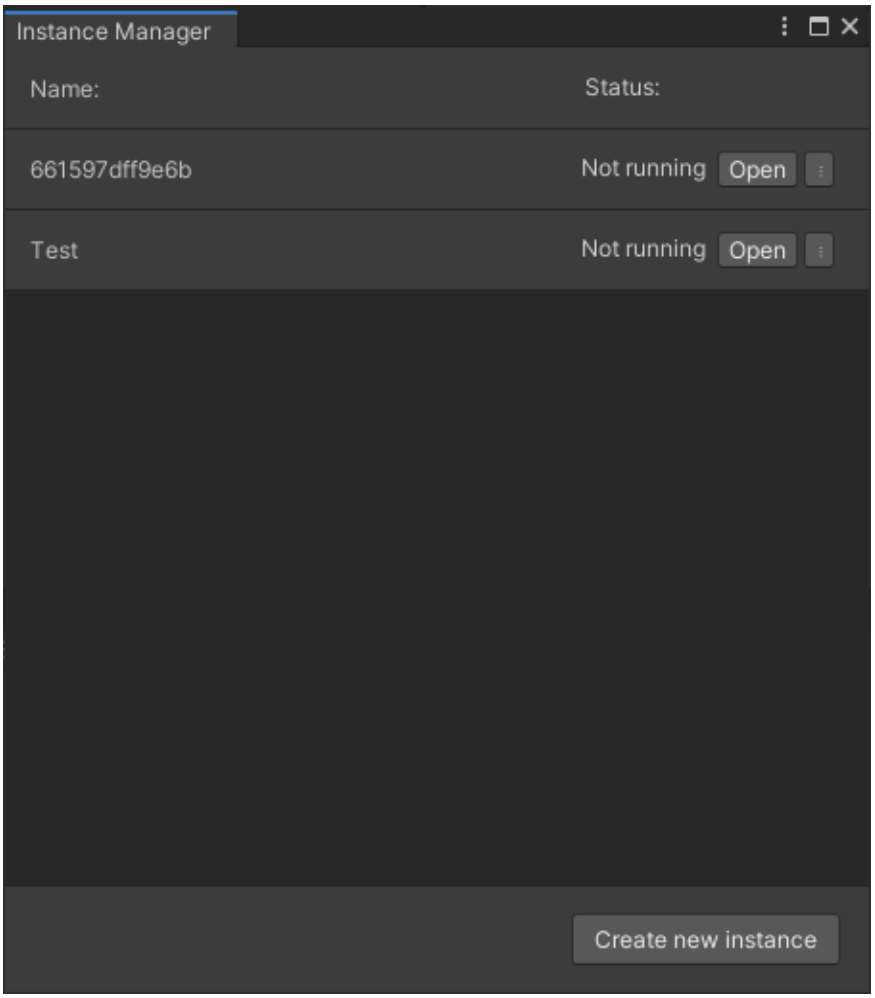
And with that we're done, you can now create more instances, experiment with options, check out the API and debug your multiplayer games with ease!

That said though, if there is anything that did not go as planned, please contact us at any of the links described in the contacts section in Home, that would be a bug :)

InstanceManagerWindowGuide

The instance manager window is the primary way of setting up secondary instances.

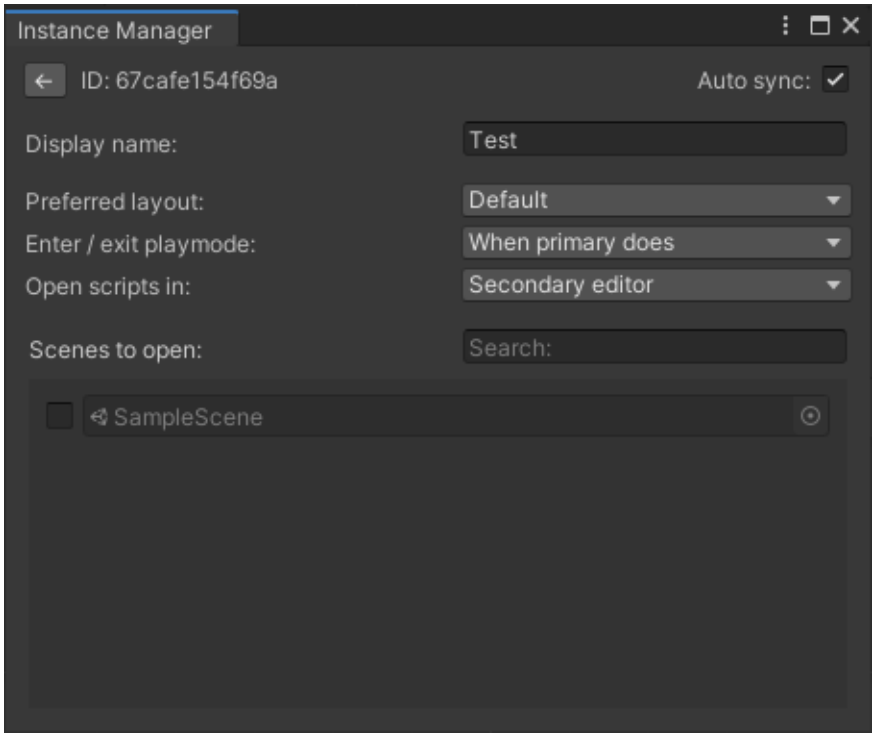




Instance list:
Lists all instances added for this project.
First column displays either Name, or ID, as fallback, if none specified.
Second column displays the status of the instance, which can be one of the following: * Not running * Running * Needs repair

Button next to status changes depending on the status of the instance, to perform the relevant function. The menu button (which can also be accessed using right click) has the following buttons: * Open, Close, Repair - Same as button next to status * Show in explorer - Opens the instance folder in explorer * Options - Displays the options for this instance * Delete - Deletes this instance

Footer:
Contains the 'Create new instance' button, which will create a new instance when pressed. Be aware that this takes a small amount of time, and progress will be displayed in lower right corner in unity 2020, but in 2019, no progress is shown.



Options

The options screen provides options for changing behavior of secondary instances.

- Autosync:
  Syncs asset databse automatically if checked, otherwise you will need manually to press the refresh button inside the secondary instance
- Display name:
  The name to display in the instance list
- Preferred layout:
  The window layout you wish to apply to the secondary instance
- Enter / exit playmode:
  Automatically enter or exit playmode in secondary instances when primary instance does
- Open scripts in:
  Primary editor: Open scripts in the editor associated with the primary instance.
  Secondary editor: Open a editor for each instance.
- Scenes to open:
  The scenes to open when the unity instance is opened

InstanceManager

`InstanceManager.InstanceManager`

The main class of Instance Manager.

Events:

System.Action OnSecondInstanceStarted Occurs during startup if current instance is secondary.

System.Action OnPrimaryPause Occurs when primary instance is paused.

System.Action OnPrimaryUnpause Occurs when primary instance is unpaused.

System.Action OnPrimaryEnterPlayMode Occurs when primary instance enters play mode.

System.Action OnPrimaryExitPlayMode Occurs when primary instance exiting play mode.

System.Action OnPrimaryAssetsChanged Occurs when primary instance has had its assets changed.

Properties:

System.Collections.Generic.IEnumerable<InstanceManager.Models.UnityInstance> instances { get; } The secondary instances that have been to this project.

InstanceManager.Models.UnityInstance instance { get; } The current instance. null if primary.

bool isPrimaryInstance { get; } Gets if the current instance is the primary instance.

bool isSecondaryInstance { get; } Gets if the current instance is a secondary instance.

string id { get; } Gets the id of the current instance.

Methods:

void SyncWithPrimaryInstance()	Sync this instance with the primary instance, does nothing if current instance is primary.

UnityInstance

`InstanceManager.Models.UnityInstance`

Represents a secondary unity instance.

Properties:

bool needsRepair { get; } Gets if this instance needs repairing.

string displayName { get; set; } The display name of this instance.

string effectiveDisplayName { get; } Gets either displayName has value.

string preferredLayout { get; set; } Gets or sets the window layout.

bool autoSync { get; set; } Gets or sets whatever this instance should auto sync asset changes.

bool openEditorInPrimaryEditor { get; set; } Gets or sets whatever scripts should open in the editor that is associated with the primary instance.

bool enterPlayModeAutomatically { get; set; } Gets or sets whatever this instance should enter / exit play mode automatically when primary instance does.

string[] scenes { get; set; } Gets the scenes this instance should open when starting.

bool isRunning { get; } Gets whatever this instance is running.

string id { get; } Gets the id of this instance.

string primaryID { get; } Gets the primary instance id that this instance is associated with.

string path { get; } Gets the path of this instance.

bool isSettingUp { get; } Gets if the instance is currently being set up.

System.Diagnostics.Process InstanceProcess { get; set; } Gets the process of this instance, if it is running.

Methods:

void Save() Saves the instance settings to disk.

void Remove() Removes the instance from disk.

void Refresh() Refreshes this UnityInstance.

void SetScene(string path, System.Nullable enabled = null, System.Nullable index = null) Set property of scene.

enabled: Set whatever this scene is enabled or not.

index: Set the index of this scene.

void ToggleOpen() Open if not running, othewise close.

void Open() Open instance.

void Close() Closes this instance.

void Close(System.Action onClosed = null) Closes this instance.

onClosed: Callback when instance is fully closed, since closing happens async.

Layout

`InstanceManager.Utility.WindowLayoutUtility+Layout`

Represents a window layout.

Properties:

string path { get; } Path on disk to this layout.

string name { get; } The name of this layout.

Methods:

void Apply() Applies this layout, if available.

ActionUtility

`InstanceManager.Utility.ActionUtility`

Provides utility functions for working with Action.

Methods:

void Try(this System.Action action, bool hideError = False) Runs the Action in a try catch block.

action: The action to run.

hideError: If true, then the error won't be rethrown.

void Try(this System.Action action, System.Exception& exception) Runs the Action in a try catch block.

action: The action to run.

exception: The exception that occured.

CommandUtility

`InstanceManager.Utility.CommandUtility`

An utility class for running commands in the system terminal.

Methods:

System.Threading.Tasks.Task RunCommand(string windows = null, string linux = null, string osx = null) Runs a command in the system terminal, chosen depending on which platform we're currently running on. Error is logged in console.

System.Threading.Tasks.Task RunCommandWindows(string command) Runs the command in the windows system terminal. Error is logged in console.

System.Threading.Tasks.Task RunCommandLinuxOSX(string command) Runs the command in the linux system terminal. Error is logged in console.

CrossProcessEventUtility

`InstanceManager.Utility.CrossProcessEventUtility`

Provides utility functions for sending 'events' to secondary instances.

Methods:

void Send(string name, string param = null) Sends an event to all open secondary instances.

name: The name of the event.

param: The parameter to send. Must be single line.

void Send(InstanceManager.Models.UnityInstance instance, string name, string param = null) Sends an event to the specified secondary instance.

instance: The instance to send the event to.

name: The name of the event.

param: The parameter to send. Must be single line.

void SendToHost(string name, string param = null) Sends an event to the primary instance.

name: The name of the event.

param: The parameter to send. Must be single line.

void On(string name, System.Action action) Adds a listener to the specified event.

void On(string name, System.Action action) Adds a listener to the specified event.

GUIExt

`InstanceManager.Editor.GUIExt`

Contains a few extra gui functions.

Methods:

void BeginColorScope(UnityEngine.Color color) Begins a color scope, this sets EndColorScope.

See also EndColorScope()

void EndColorScope() Ends the color scope, that was started with BeginColorScope(UnityEngine.Color).

void BeginEnabledScope(bool enabled, bool overrideWhenAlreadyFalse = False) Begins an enabled scope, this sets EndEnabledScope.

See also EndColorScope()

void EndEnabledScope() Ends the enabled scope, that was started with BeginEnabledScope(bool).

void AddItem(this UnityEditor.GenericMenu menu, UnityEngine.GUIContent content, System.Action action, bool enabled = True, bool isChecked = False, UnityEngine.GUIContent offContent = null) Adds an item to this GenericMenu.

menu: The GenericMenu.

content: The content of this item.

action: The action to perform when click, if enabled.

enabled: Sets whatever this item is enabled.

isChecked: Sets if checked.

offContent: The content to display when item disabled, defaults to content if false.

void AddItem(this UnityEditor.GenericMenu menu, UnityEngine.GUIContent content, System.Action action, bool isChecked, bool enabled = True, UnityEngine.GUIContent offContent = null) Adds an item to this GenericMenu.

menu: The GenericMenu.

content: The content of this item.

action: The action to perform when click, if enabled.

isChecked: Sets if checked.

enabled: Sets whatever this item is enabled.

offContent: The content to display when item disabled, defaults to content if false.

bool UnfocusOnClick() Unfocuses elements when blank area of EditorWindow clicked. Returns true if element was unfocused, you may want to Repaint() then.

InstanceUtility

`InstanceManager.Utility.InstanceUtility`

Provides utility functions for working with secondary instances.

Events:

System.Action onInstancesChanged Occurs when an instance is changed.

Fields:

string instanceFileName The name of the instance settings file.

Methods:

InstanceManager.Models.UnityInstance LocalInstance() Loads local instance file. Returns null if none exists or instance is primary.

InstanceManager.Models.UnityInstance Find(string id) Finds the secondary instance with the specified id.

System.Collections.Generic.IEnumerable<InstanceManager.Models.UnityInstance> Enumerate() Enumerates all secondary instances for this project.

InstanceManager.Models.UnityInstance Create() Create a new secondary instance. Returns null if current instance is secondary.

System.Threading.Tasks.Task Repair(InstanceManager.Models.UnityInstance instance, string path) Repairs the instance. No effect if current instance is secondary.

bool NeedsRepair(InstanceManager.Models.UnityInstance instance) Gets if the instance needs to be repaired.

bool IsInstanceBeingSetUp(InstanceManager.Models.UnityInstance instance) Gets if the UnityInstance is being set up, this would be when its being created, or when being removed.

ProgressUtility

`InstanceManager.Utility.ProgressUtility`

Provides functions for running tasks that display progress.

Note that progress is only displayed in Unity 2020 and higher, task will still run in earlier versions of unity, but no progress will be shown.

Methods:

System.Threading.Tasks.Task RunTask(string displayName, System.Threading.Tasks.Task task, System.Action<System.Threading.Tasks.Task> onComplete = null, string description = null, int minDisplayTime = 250, bool canRun = True, bool hideProgress = False, bool hideError = False) Shows progress in unity for a task. Only inderminate progress bar is supported.

displayName: The display name for the progress item.

task: The task that display progress for. The task cannot be running already.

onComplete: The callback to be invoked when task is done (minDisplayTime has no effect).

description: The description for the progress item.

minDisplayTime: The minimum display time of the progress bar, makes sure that the progress is displayed and readable, instead of just flickering.

canRun: Prevents the task from running and does not create a progress item if false.

hideError: Prevents the error from getting logged.

hideProgress: Prevents progress from being shown.

WindowLayoutUtility

`InstanceManager.Utility.WindowLayoutUtility`

Provides methods for enumerating and applying window layouts.

Properties:

bool isAvailable { get; } Gets whatever the utility was able to find the internal unity methods or not.

string layoutsPath { get; set; } The path to the layouts folder.

InstanceManager.Utility.WindowLayoutUtility.Layout[] availableLayouts { get; } Finds all available layouts.

Methods:

InstanceManager.Utility.WindowLayoutUtility.Layout Find(string name) Finds the specified layout by name.

System.Nullable<InstanceManager.Utility.WindowLayoutUtility.Layout> GetCurrent() Gets the current layout.