

# Namespace NGOEventExtender

## Classes

[ActionStreamExtender](#)

[EndingExtender](#)

[EventExtender](#)

[EventHelper](#)

[ExtActionNgoStream](#)

[ExtAlphaTypeToData](#)

[ExtNgoStream](#)

The blueprint used to load in a custom stream.

Make sure your variable of this class isn't static, so that your stream dialogue and your comment's language can be updated if this stream is used more than once.

[NgoExtEvent](#)

An NgoEvent with a condition attached to it.

[PlayingBlock](#)

[StreamExtender](#)

[StreamSettings](#)

## Enums

[CEffectState](#)

The states used to load in a border effect.

- In - Effect loads in and is set into its first state.
- Win\_Stop - Animation is paused.
- Win - If effect is loaded in, it is set into its primary state.
- Out - Exits the effect.

[StreamBackground](#)

In-game sprites that are used for a stream's background. You can load these into a custom stream using [SetExtStreamBG](#).

[StreamChatType](#)

Different behaviour of how stream chat comments show in a stream.

**Normal:** Comments can be selected and deleted.

**Uncontrollable:** Comments are unselectable and are grayed out.

**Celebration:** Comments are unselectable and are either super chats or grayed out randomly.

# Class ActionStreamExtender

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
[HarmonyPatch]
public class ActionStreamExtender
```

## Inheritance

[object](#) ← ActionStreamExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ActionStreamExtender()

```
public ActionStreamExtender()
```

## Methods

### IsActionStreamDiscovered(ExtActionNgoStream)

Checks to see if an [ExtActionNgoStream](#) is discovered as an idea. (but not streamed)

```
public static bool IsActionStreamDiscovered(ExtActionNgoStream actionStream)
```

#### Parameters

`actionStream` [ExtActionNgoStream](#)

The [ExtActionNgoStream](#) to check for.

Returns

[bool](#) ↗

## IsActionStreamStreamed(ExtActionNgoStream)

Checks to see if an [ExtActionNgoStream](#) has been streamed already.

```
public static bool IsActionStreamStreamed(ExtActionNgoStream actionStream)
```

Parameters

[actionStream](#) [ExtActionNgoStream](#)

The [ExtActionNgoStream](#) to check for.

Returns

[bool](#) ↗

# Enum CEffectState

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

The states used to load in a border effect.

- **In** - Effect loads in and is set into its first state.
- **Win\_Stop** - Animation is paused.
- **Win** - If effect is loaded in, it is set into its primary state.
- **Out** - Exits the effect.

```
public enum CEffectState
```

## Fields

**In** = 0

**Out** = 3

**Win** = 2

**Win\_Stop** = 1

# Class EndingExtender

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
[HarmonyPatch]
public class EndingExtender
```

## Inheritance

[object](#) ← EndingExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## EndingExtender()

```
public EndingExtender()
```

# Methods

## AddExtEndings(List<Param>, string)

Adds custom endings to the Ending [ExtList](#). This is required to load custom endings into the game. In addition, a path to a [json](#) file in your mod folder is required to save custom ending history.

```
public static void AddExtEndings(List<EndingMaster.Param> exEndList, string exHistoryPath)
```

## Parameters

exEndList [List](#)<EndingMaster.Param>

exHistoryPath [string](#)

## Remarks

The `json` path for custom ending history accounts for other mods that have custom endings.

## GetCustomEndingName(EndingType)

Loads an EndingMaster.Param based on the parsed EndingType.

```
public static EndingMaster.Param GetCustomEndingName(EndingType type)
```

## Parameters

`type` EndingType

The parsed type used to load the EndingMaster.Param.

## Returns

EndingMaster.Param

## Remarks

The parsed type is based off of the Id's unique number id at the end of the title.

## Exceptions

[ArgumentException](#)

## IsCustomEnding()

Checks if a custom ending is currently playing.

```
public static bool IsCustomEnding()
```

## Returns

[bool](#)

## IsCustomEnding(string)

Checks if a specific custom ending is currently playing.

```
public static bool IsCustomEnding(string id)
```

Parameters

**id** [string](#)

The custom EndingMaster.Param's Id to check for.

Returns

[bool](#)

## SetEndingFlag(EndingType)

Sets an ending flag based on the EndingType.

```
public static void SetEndingFlag(EndingType ending)
```

Parameters

**ending** EndingType

The EndingType used to set the ending flag. If setting a custom ending flag, use [GetUniqueEndNum](#), or use your unique number at the end of your ID and add 1000 to it, then cast it as an EndingType

## SetExtEndingFlag(string)

Sets a custom ending flag based on the custom ending's Id.

```
public static void SetExtEndingFlag(string id)
```

Parameters

**id** [string](#)

The EndingMaster.Param's Id used to set the custom ending flag.

## StartEndingScreen(bool)

Starts the ending blue screen that signifies achieving a new ending.

```
public static void StartEndingScreen(bool isWindowFirst)
```

### Parameters

**isWindowFirst** [bool](#)

If true, opens the "the end" window first before showing the end screen. Otherwise, shows the end screen instantly.

# Class EventExtender

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
[HarmonyPatch]
public class EventExtender
```

## Inheritance

[object](#) ← EventExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## EventExtender()

```
public EventExtender()
```

# Methods

## CanHeadpat(bool)

Enables or disables the ability to headpat Ame's head. Useful if you don't want Ame's head to be interactable during events.

```
public static void CanHeadpat(bool isPat)
```

## Parameters

isPat [bool](#)

Can Ame's head be patted or not?

## Remarks

Ability to pat will be reset to true during the boot screen, reloading a day or going to the next day.

## HijackOriginalEvent(NgoEvent, NgoExtEvent)

Hijacks a queued original event and loads a custom event instead.

If the custom event has a condition, if will only replace the original if the condition is met.

```
public static void HijackOriginalEvent(NgoEvent original, NgoExtEvent replacement)
```

## Parameters

**original** NgoEvent

The event to replace.

**replacement** [NgoExtEvent](#)

The event that will be loaded instead.

## Remarks

This method adds the custom event to a list, where it will track when its related original event is dequeued.

Therefore, it's not required to call the same method with the same arguments more than once.

## HijackOriginalEvent(NgoEvent, List<NgoExtEvent>)

Hijacks a queued original event and loads a custom event instead.

If the custom event has a condition, if will only replace the original if the condition is met.

```
public static void HijackOriginalEvent(NgoEvent original, List<NgoExtEvent> replacements)
```

## Parameters

**original** NgoEvent

The event to replace.

replacements [List](#) <NgoExtEvent>

The list of events that will be loaded instead.

## Remarks

This method adds the custom events to a list, where it will track when its related original event is dequeued.

Therefore, it's not required to call the same method with the same arguments more than once.

## HijackOriginalEvent<T>(NgoExtEvent)

Hijacks a queued original event and loads a custom event instead.

If the custom event has a condition, it will only replace the original if the condition is met.

```
public static void HijackOriginalEvent<T>(NgoExtEvent replacement) where T : NgoEvent
```

## Parameters

replacement [NgoExtEvent](#)

The event that will be loaded instead.

## Type Parameters

T

The event to replace.

## Remarks

This method adds the custom event to a list, where it will track when its related original event is dequeued.

Therefore, it's not required to call the same method with the same arguments more than once.

## HijackOriginalEvent<T>(List<NgoExtEvent>)

Hijacks a queued original event and loads a custom event instead.

If the custom event has a condition, it will only replace the original if the condition is met.

```
public static void HijackOriginalEvent<T>(List<NgoExtEvent> replacements) where T : NgoEvent
```

## Parameters

**replacements** [List](#)<[NgoExtEvent](#)>

The list of events that will be loaded instead.

## Type Parameters

**T**

The event to replace.

## Remarks

This method adds the custom events to a list, where it will track when its related original event is dequeued.

Therefore, it's not required to call the same method with the same arguments more than once.

## HijackOriginalEvent<T, ExT>()

Hijacks a queued original event and loads a custom event instead.

If the custom event has a condition, it will only replace the original if the condition is met.

```
public static void HijackOriginalEvent<T, ExT>() where T : NgoEvent where ExT : NgoExtEvent
```

## Type Parameters

**T**

The event to replace.

**ExT**

The event that will be loaded instead.

## Remarks

This method adds the custom event to a list, where it will track when its related original event is dequeued.

Therefore, it's not required to call the same method with the same arguments more than once.

## SetRandomDayExtEvents(List<NgoExtEvent>)

Sets custom Random Day Events. How these events are loaded are based on the event's title.

```
public static void SetRandomDayExtEvents(List<NgoExtEvent> extEventList)
```

### Parameters

**extEventList** [List](#)<[NgoExtEvent](#)>

The list of custom random day events to load.

## SetSpecialDayExtEvent(List<NgoExtEvent>)

Sets custom Special Day Events.

In-game Special Day Events (fixed/milestones) take priority over your events, so your conditions have to be unique.

```
public static void SetSpecialDayExtEvent(List<NgoExtEvent> extEventList)
```

### Parameters

**extEventList** [List](#)<[NgoExtEvent](#)>

The list of custom special day events to load.

## SetSpecialMidnightExtEvent(List<NgoExtEvent>)

Sets custom Special Midnight Events. (the time after Sleep To Tomorrow is pressed, but before the day transitions to the next day.)

In-game Special Midnight Events take priority over your events, so your conditions have to be unique.

```
public static void SetSpecialMidnightExtEvent(List<NgoExtEvent> extEventList)
```

## Parameters

**extEventList** [List](#)<[NgoExtEvent](#)>

The list of custom special day events to load.

## SetSpecialNightExtEvent(List<NgoExtEvent>)

Sets custom Special Night Events.

In-game Special Night Events take priority over your events, so your conditions have to be unique.

```
public static void SetSpecialNightExtEvent(List<NgoExtEvent> extEventList)
```

## Parameters

**extEventList** [List](#)<[NgoExtEvent](#)>

The list of custom special day events to load.

## StartEvent(NgoEvent)

Starts a custom event.

```
public static void StartEvent(NgoEvent ngoEvent)
```

## Parameters

**ngoEvent** NgoEvent

The event that will be loaded.

## Exceptions

[ArgumentException](#)

## StartEvent<T>()

Starts a custom event.

```
public static void StartEvent<T>() where T : NgoEvent, new()
```

## Type Parameters

T

The event that will be loaded.

## Exceptions

[ArgumentException](#) ↗

# Class EventHelper

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
[HarmonyPatch]
public class EventHelper
```

## Inheritance

[object](#) ← EventHelper

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## EventHelper()

```
public EventHelper()
```

# Methods

## ChangeJineAmeIcon(Sprite)

Changes Ame's JINE icon to a new Sprite.

```
public static void ChangeJineAmeIcon(Sprite sprite)
```

## Parameters

**sprite** Sprite

The Sprite to change Ame's icon too.

## ChangeTweetAmeIcon(Sprite)

Changes Ame's Tweeter/Poketter icon to a new Sprite.

```
public static void ChangeTweetAmeIcon(Sprite sprite)
```

Parameters

**sprite** Sprite

The Sprite to change Ame's icon too.

## ChangeTweetKAngellIcon(Sprite)

Changes KAngel's Tweeter/Poketter icon to a new Sprite.

```
public static void ChangeTweetKAngelIcon(Sprite sprite)
```

Parameters

**sprite** Sprite

The Sprite to change Ame's icon too.

## ResetIcons(bool, bool, bool)

Resets all changed Icons to its default.

```
public static void ResetIcons(bool resetJineIcon = true, bool resetAngelTweetIcon = true,  
bool resetAmeTweetIcon = true)
```

Parameters

**resetJineIcon** bool

Reset Ame's Jine icon? Default is **true**.

**resetAngelTweetIcon** bool

Reset KAngel's Tweeter/Poketter icon? Default is `true`.

### `resetAmeTweetIcon` [bool](#)

Reset Ame's Tweeter/Poketter icon? Default is `true`.

## `SetLastDayNotif(Action, CompositeDisposable)`

Spawns in a notification that does an action you choose after. Meant for any endings on the last day.

```
public static void SetLastDayNotif(Action action, CompositeDisposable disposables)
```

### Parameters

#### `action` [Action](#)

The `Action` to play when the notification is pressed.

#### `disposables` [CompositeDisposable](#)

The `CompositeDisposable` to use to dispose the subscribed notification event.

## `SetShortcutTaskbarState(bool, float)`

Sets the interactivity of the shortcuts and the taskbar.

```
public static void SetShortcutTaskbarState(bool isActive, float alphaIfInactive = 0.4)
```

### Parameters

#### `isActive` [bool](#)

Can you interact with the shortcuts and the taskbar?

#### `alphaIfInactive` [float](#)

The alpha / opaqueness of the shortcuts if it can't be interactable.

## `StartYearsPass()`

Loads in the graphic where a number of days pass (found in Painful Future, Labor Is Evil, etc.)

```
public static void StartYearsPass()
```

# Class ExtActionNgoStream

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
public abstract class ExtActionNgoStream : ExtNgoStream
```

## Inheritance

[object](#) ← [ExtNgoStream](#) ← ExtActionNgoStream

## Inherited Members

[ExtNgoStream.SetCondition\(\)](#) , [ExtNgoStream.SetStreamSettings\(\)](#) , [ExtNgoStream.AfterStream\(\)](#) ,  
[ExtNgoStream.StreamTitle](#) , [ExtNgoStream.StartingAnim](#) , [ExtNgoStream.NowPlaying](#) ,  
[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ExtActionNgoStream()

```
protected ExtActionNgoStream()
```

## Fields

### isDiscovered

```
protected bool isDiscovered
```

Field Value

[bool](#)

## Properties

## ActionStreamId

```
public abstract string ActionStreamId { get; }
```

Property Value

[string](#) ↗

## CommandResult

```
public virtual CmdMaster.Param CommandResult { get; }
```

Property Value

CmdMaster.Param

## HintData

```
public abstract ExtAlphaTypeToData HintData { get; }
```

Property Value

[ExtAlphaTypeToData](#)

## SearchResult

```
public virtual Egosamaster.Param SearchResult { get; }
```

Property Value

Egosamaster.Param

## TweetResult

```
public abstract List<TweetType> TweetResult { get; }
```

Property Value

[List](#) <TweetType>

# Class ExtAlphaTypeToData

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
public class ExtAlphaTypeToData : AlphaTypeToData
```

## Inheritance

[object](#) ← NetaTypeToData ← AlphaTypeToData ← ExtAlphaTypeToData

## Inherited Members

AlphaTypeToData.NetaType , AlphaTypeToData.level , NetaTypeToData.NetaGenreJP ,  
NetaTypeToData.NetaNameJP , NetaTypeToData.NetaNameEN , NetaTypeToData.NetaGenreEN ,  
NetaTypeToData.NetaGenreCN , NetaTypeToData.NetaNameCN , NetaTypeToData.NetaGenreTW ,  
NetaTypeToData.NetaNameTW , NetaTypeToData.NetaGenreKO , NetaTypeToData.NetaNameKO ,  
NetaTypeToData.getJouken , [object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ExtAlphaTypeToData(Param)

```
public ExtAlphaTypeToData(TenCommentMaster.Param param)
```

## Parameters

**param** TenCommentMaster.Param

## Properties

### paramId

```
public TenCommentMaster.Param paramId { get; }
```

Property Value

TenCommentMaster.Param

# Class ExtNgoStream

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

The blueprint used to load in a custom stream.

Make sure your variable of this class isn't static, so that your stream dialogue and your comment's language can be updated if this stream is used more than once.

```
public abstract class ExtNgoStream
```

Inheritance

[object](#) ← ExtNgoStream

Derived

[ExtActionNgoStream](#)

Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ExtNgoStream()

```
protected ExtNgoStream()
```

## Properties

### NowPlaying

The playing list.

```
public virtual List<Playing> NowPlaying { get; }
```

Property Value

[List ↗](#) <Playing>

## StartingAnim

The starting animation of a stream; only applies if KAngel is the first one talking.

```
public virtual string StartingAnim { get; }
```

Property Value

[string ↗](#)

## StreamTitle

The stream title.

```
public virtual string StreamTitle { get; }
```

Property Value

[string ↗](#)

## Methods

### AfterStream()

What happens after the stream script ends.

```
public abstract UniTask AfterStream()
```

Returns

UniTask

## SetCondition()

The condition for the event.

Event will only start if the condition returns true.

```
public abstract bool SetCondition()
```

Returns

bool

Remarks

`SetCondition` is not read if this `ExtNgoStream` is used with `StartCustomStream`.

## SetStreamSettings()

Sets any relevant attributes to the stream before it starts (music, effects, etc).

You can set more attributes to the stream by changing fields from `Live`, using a Singleton to create an instance for `Live`.

```
public abstract void SetStreamSettings()
```

# Class NgoExtEvent

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

An NgoEvent with a condition attached to it.

```
public abstract class NgoExtEvent : NgoEvent
```

## Inheritance

[object](#) ← NgoEvent ← NgoExtEvent

## Inherited Members

NgoEvent.type , NgoEvent.eventName , NgoEvent.token1 , NgoEvent.token2 , NgoEvent.isStarted ,  
NgoEvent.cursorMode , NgoEvent.hotSpot , NgoEvent.compositeDisposable , NgoEvent.\_player ,  
NgoEvent.Awake() , [NgoEvent.ObiActive\(bool\)](#) , [NgoEvent.DelaySkippable\(int\)](#) ,  
[NgoEvent.startEvent\(CancellationToken\)](#) , [NgoEvent.startEvent\(CancellationToken, bool\)](#) ,  
NgoEvent.endEvent() , NgoEvent.OnDestroy() , [NgoEvent.ClickableAllScreen\(bool\)](#) , NgoEvent.GoOut() ,  
NgoEvent.BackFromOdekake() , [NgoEvent.NadeNade\(Action\)](#) ,  
[NgoEvent.tweetFromTip\(AlphaType, int\)](#) , NgoEvent.Player , [object.ToString\(\)](#) ,  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.ReferenceEquals\(object, object\)](#) ,  
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#)

## Constructors

### NgoExtEvent()

```
protected NgoExtEvent()
```

## Properties

### Id

```
public virtual string Id { get; }
```

Property Value

[string](#)

## Methods

### SetCondition()

The condition for the event.

Event will only start if the condition returns true.

```
public abstract bool SetCondition()
```

Returns

[bool](#)

Remarks

`SetCondition` is not read if this `NgoExtEvent` is used with `StartCustomEvent`.

# Class PlayingBlock

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
public class PlayingBlock
```

## Inheritance

[object](#) ← PlayingBlock

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

PlayingBlock()

```
public PlayingBlock()
```

## Methods

AngelTalk(string, string)

Dialogue from KAngel.

```
public static Playing AngelTalk(string chatId, string animId = "")
```

### Parameters

chatId [string](#)

The TenComment.Param's Id used to load the dialogue.

animId [string](#)

The animation that KAngel uses when she speaks.

Returns

Playing

## AngelTalkToHaters(string, string, string)

A hate comment which KAngel responds to.

```
public static List<Playing> AngelTalkToHaters(string aChatId, string hChatId, string animId  
= "")
```

Parameters

**aChatId** [string](#)

The TenComment.Param's Id used to load the dialogue.

**hChatId** [string](#)

The TenComment.Param's Id used to load the comment.

**animId** [string](#)

The animation that KAngel uses when she speaks.

Returns

[List](#) <Playing>

## KTalk(string)

An ordinary chat comment.

```
public static Playing KTalk(string chatId)
```

Parameters

### **chatId** [string](#)

The KusoComment.Param's Id used to load the comment.

Returns

Playing

## KusoKTalk(string)

A stressful chat comment. Delete these to reduce Stress by -1.

```
public static Playing KusoKTalk(string chatId)
```

Parameters

### **chatId** [string](#)

The KusoComment.Param's Id used to load the comment.

Returns

Playing

## PlayEffect(ChanceEffectType, CEffectState)

Plays a border effect during a stream.

```
public static Playing PlayEffect(ChanceEffectType chanceEffect, CEffectState state)
```

Parameters

### **chanceEffect** ChanceEffectType

The type of border effect.

### **state** [CEffectState](#)

The border effect state.

Returns

Playing

## PlaySound(SoundType, bool)

Plays a sound during a stream.

```
public static Playing PlaySound(SoundType sound, bool looping)
```

Parameters

**sound** SoundType

The sound that plays.

**looping** bool

If the sound loops or not.

Returns

Playing

## ReadComments()

The time when KAngel starts reading super chat comments.

```
public static Playing ReadComments()
```

Returns

Playing

## SetMobs(string)

Used to load random chat comments, based on follower count.

"**first**": Comments that usually appear at the start of the stream.

"middle": Comments that usually appear around the middle of the stream.

"last": Comments that usually appear when the stream ends.

"rainbow": Creates a super chat rainbow.

```
public static Playing SetMobs(string state = "middle")
```

## Parameters

**state** [string](#)

What random chat comments are loaded. Refer to the summary for more details.

## Returns

Playing

## Remarks

You can also use other states such as "**delete**" and "**deleteAll**" to remove stream comments during a stream.

"**delete**" and "**deleteAll**" will not delete superchats, in addition, any stressful comments deleted with this command will not apply -1 Stress.

## SuperKTalk(string, string, string)

A Super Chat comment, which KAngel can read near the end of the stream.

```
public static Playing SuperKTalk(string kChatId, string aChatId, string aAnimId = "")
```

## Parameters

**kChatId** [string](#)

The KusoComment.Param's Id used to load the comment.

**aChatId** [string](#)

The TenComment.Param's Id used to load her response.

**aAnimId** [string](#)

The animation that KAngel uses when she speaks.

Returns

Playing

## SuperKTalk\_RepMore(string, List<string>, List<string>)

A Super Chat comment, which KAngel can read near the end of the stream. Gives out two or more responses to the same comment.

```
public static Playing SuperKTalk_RepMore(string kChatId, List<string> aChatId,  
List<string> aAnimId)
```

Parameters

kChatId [string](#)

The KusoComment.Param's Id used to load the comment.

aChatId [List](#)<[string](#)>

The [List](#) of TenComment.Param Ids used to load her responses.

aAnimId [List](#)<[string](#)>

The [List](#) of animation Ids that KAngel does when she speaks, per response.

Returns

Playing

# Enum StreamBackground

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

In-game sprites that are used for a stream's background. You can load these into a custom stream using `SetExtStreamBG`.

```
public enum StreamBackground
```

## Fields

`BigHouse` = 10

`Black` = 12

`Default` = 0

`Gold` = 2

`Guru` = 8

`Horror` = 9

`MileFive` = 7

`MileFour` = 6

`MileOne` = 3

`MileThree` = 5

`MileTwo` = 4

`None` = 1000

`Roof` = 11

`Silver` = 1

`Void` = 13

# Enum StreamChatType

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

Different behaviour of how stream chat comments show in a stream.

**Normal:** Comments can be selected and deleted.

**Uncontrollable:** Comments are unselectable and are grayed out.

**Celebration:** Comments are unselectable and are either super chats or grayed out randomly.

```
public enum StreamChatType
```

## Fields

Celebration = 2

Normal = 0

Uncontrollable = 1

# Class StreamExtender

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
[HarmonyPatch]
public class StreamExtender
```

## Inheritance

[object](#) ← StreamExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## StreamExtender()

```
public StreamExtender()
```

# Methods

## AddConditionalStream(ExtNgoStream)

Adds a custom stream to the game, where it will only play if its conditions are met.  
Useful if you want to replace a queued stream with your own.

```
public static void AddConditionalStream(ExtNgoStream stream)
```

## Parameters

**stream** [ExtNgoStream](#)

The custom stream to add.

## Remarks

Your streams take priority over others, so make sure your conditions are specific.

## AddConditionalStream<T>()

Adds a custom stream to the game, where it will only play if its conditions are met.  
Useful if you want to replace a queued stream with your own.

```
public static void AddConditionalStream<T>() where T : ExtNgoStream
```

## Type Parameters

T

The custom stream to add.

## Remarks

Your streams take priority over others, so make sure your conditions are specific.

## AddExtActionStream(ExtActionNgoStream)

Adds a custom normal stream to the game, where it will only play if its conditions are met.  
(Normal: the streams you can choose during a night.)  
Useful if you want to replace a queued stream with your own.

```
public static void AddExtActionStream(ExtActionNgoStream stream)
```

## Parameters

stream [ExtActionNgoStream](#)

The custom stream to add.

## Remarks

Your streams take priority over others, so make sure your conditions are specific.

## AddExtActionStream<T>()

Adds a custom normal stream to the game, where it will only play if its conditions are met.  
(Normal: the streams you can choose during a night.)  
Useful if you want to replace a queued stream with your own.

```
public static void AddExtActionStream<T>() where T : ExtActionNgoStream
```

### Type Parameters

T

The custom stream to add.

### Remarks

Your streams take priority over others, so make sure your conditions are specific.

## AwaitCustomStream(ExtNgoStream)

Saves a custom stream to be used later when the Broadcast window is opened. Useful if you want to start a stream without the transformation playing.

```
public static UniTask AwaitCustomStream(ExtNgoStream stream)
```

### Parameters

stream [ExtNgoStream](#)

The custom stream to load later on.

### Returns

UniTask

### Remarks

Will only be saved for one upcoming stream only. (it will reset after a stream)

## AwaitCustomStream<T>()

Saves a custom stream to be used later when the Broadcast window is opened. Useful if you want to start a stream without the transformation playing.

```
public static UniTask AwaitCustomStream<T>() where T : ExtNgoStream
```

Returns

UniTask

Type Parameters

T

The custom stream to load later on.

Remarks

Will only be saved for one upcoming stream only. (it will reset after a stream)

## AwaitDarkAngelTransform()

Sets the Dark Angel transformation scene to play instead of the normal transformation scene the next time the Transform! window appears.

```
public static void AwaitDarkAngelTransform()
```

Remarks

Will only play for one upcoming Transform! scene only. (it will reset after a stream)

## EndCustomStream()

Ends a custom stream. Calculates stats if an ExtActionNgoStream is playing, otherwise just closes the Stream window.

```
public static void EndCustomStream()
```

## InitializeFirstAnim(string)

Caches the first animation to be used in a later stream. Recommended if KAngel is the first one to speak in a stream.

```
public static UniTask InitializeFirstAnim(string anim)
```

Parameters

`anim` [string](#)

Returns

`UniTask`

Remarks

`Action_HaishinStart` already does this process.

## StartCustomStream(ExtNgoStream, bool, bool)

Starts a custom stream, based on the contents from your `ExtNgoStream`.

Make sure the `ExtNgoStream` you use isn't static.

```
public static void StartCustomStream(ExtNgoStream stream, bool isDarkUI = false, bool isDarkAnim = false)
```

Parameters

`stream` [ExtNgoStream](#)

The `ExtNgoStream` data used to load your custom stream. Make sure the `ExtNgoStream` you use isn't static.

`isDarkUI` [bool](#)

If true, loads in the Dark UI for the stream, otherwise uses the default stream UI.

`isDarkAnim` [bool](#)

If true, uses the Dark Angel transformation before a stream, otherwise uses the default KAngel transformation.

## StartCustomStream<T>(bool, bool)

Starts a custom stream, based on the contents from your [ExtNgoStream](#).

Make sure the [ExtNgoStream](#) you use isn't static.

```
public static void StartCustomStream<T>(bool isDarkUI = false, bool isDarkAnim = false)  
where T : ExtNgoStream
```

### Parameters

[isDarkUI](#) [bool](#)

If true, loads in the Dark UI for the stream, otherwise uses the default stream UI.

[isDarkAnim](#) [bool](#)

If true, uses the Dark Angel transformation before a stream, otherwise uses the default KAngel transformation.

### Type Parameters

[T](#)

The [ExtNgoStream](#) data used to load your custom stream.

# Class StreamSettings

Namespace: [NGOEventExtender](#)

Assembly: NGOEventExtender.dll

```
public class StreamSettings
```

## Inheritance

[object](#) ← StreamSettings

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### StreamSettings()

```
public StreamSettings()
```

## Methods

### HasEndSplashScreen(bool)

Sets whether the splash screen after a stream ends shows up or not.

```
public static void HasEndSplashScreen(bool hasScreen)
```

## Parameters

hasScreen [bool](#)

If false, the end screen will not appear after a stream.

## Remarks

Note: This only affects the end screen if it is planned to show up after a stream, as some endings affect how the end screen is shown.

## SetCustomReactAnim(string)

Sets the animation for when KAngel starts reading super chat comments.

```
public static void SetCustomReactAnim(string reactAnim)
```

### Parameters

**reactAnim** [string](#)

The name of the animation to play.

## SetExtStreamBG(StreamBackground)

Sets a custom background for a custom stream.

```
public static void SetExtStreamBG(StreamBackground bg)
```

### Parameters

**bg** [StreamBackground](#)

The in-game sprite used to set the image.

### Exceptions

[ArgumentException](#)

Throws if the sprite is null.

## SetExtStreamBG(Sprite)

Sets a custom background for a custom stream.

```
public static void SetExtStreamBG(Sprite sprite)
```

## Parameters

**sprite** Sprite

The custom sprite used to set the image.

## Exceptions

[ArgumentException](#)

Throws if the sprite is null.

## SetExtWatchingNum(int)

Sets the base number of people that are watching a custom stream.

Number cannot be lower than 192, however it can be set to 0.

```
public static void SetExtWatchingNum(int watchNum)
```

## Parameters

**watchNum** [int](#)

The base number of people that are watching a custom stream.

## Remarks

If set to 0, the number will not randomly refresh throughout the stream.

## SetStreamChatType(StreamChatType)

Sets the behaviour of stream chat comments in a stream.

```
public static void SetStreamChatType(StreamChatType chatType)
```

## Parameters

## chatType StreamChatType

The StreamChatType of the stream.

## SetStreamTitle(string)

Sets the stream title.

```
public static string SetStreamTitle(string id)
```

Parameters

### id string ↗

The id associated with a TenComment param object.

Returns

### string ↗

# Namespace NGOExtraSettings

## Classes

[ExtraSettings](#)

# Class ExtraSettings

Namespace: [NGOExtraSettings](#)

Assembly: NGOExtraSettings.dll

```
public class ExtraSettings
```

## Inheritance

[object](#) ← ExtraSettings

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## ExtraSettings()

```
public ExtraSettings()
```

# Methods

## LoadFromExtSettings<T>(string)

```
public static T LoadFromExtSettings<T>(string key)
```

## Parameters

key [string](#)

## Returns

T

## Type Parameters

T

### SaveToExtSettings<T>(string, T)

```
public static void SaveToExtSettings<T>(string key, T data)
```

## Parameters

key [string](#)

data T

## Type Parameters

T

# Namespace NGOTxtExtender

## Classes

[EgosaExtender](#)

[ExtTextManager](#)

[JineExtender](#)

[KRepExtender](#)

[KitsuneExtender](#)

[KusoComExtender](#)

[MobComExtender](#)

[SysTxtExtender](#)

[TenComExtender](#)

[TooltxtExtender](#)

[TweetExtender](#)

# Class EgosaExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class EgosaExtender
```

## Inheritance

[object](#) ← EgosaExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## EgosaExtender()

```
public EgosaExtender()
```

# Methods

## IsCustomSearch(bool, string, string)

Sets if a Vanity Search Thread is custom or not.

```
public static void IsCustomSearch(bool isCustom, string type = null, string jouken = null)
```

### Parameters

**isCustom** [bool](#)

If the thread is custom.

## **type** [string](#)

The type used to search any matching Egosamaster.Params. (optional)

## **jouken** [string](#)

The condition used to search for any matching Egosamaster.Params, in addition to type (optional).

# OpenALotOfEyeWindows()

Creates a lot of Eye windows with custom search results made by the user. Search results will only appear with Egosamaster.Params that has a **Type** of "eyes".

```
public static UniTask OpenALotOfEyeWindows()
```

Returns

UniTask

Remarks

Only 22 Eye windows can be loaded per save instance. The counter will only reset when a save is loaded through the Login screen or the Load window.

# OpenCustomSearchWindow(string, string)

Creates a Window with custom search results made by the user. These search results are pre-set and are not randomized.

```
public static UniTask OpenCustomSearchWindow(string type, string jouken = null)
```

Parameters

## **type** [string](#)

The type used to search any matching Egosamaster.Params.

## **jouken** [string](#)

The condition used to search for any matching Egosamaster.Params, in addition to type (optional).

Returns

UniTask

Remarks

Note: Search results load the first param at the bottom of the result list, and vice-versa.

**IsCustomSearch** also sets to false after using this method.

## OpenEyeSearchWindow(string)

Creates an Eye window with a custom search result made by the user.

```
public static void OpenEyeSearchWindow(string id)
```

Parameters

**id** [string](#)

The EgosMaster.Param's Id used to load the search result.

Remarks

Only 22 Eye windows can be loaded per save instance. The counter will only reset when a save is loaded through the Login screen or the Load window.

Exceptions

[NullReferenceException](#)

# Class ExtTextManager

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class ExtTextManager
```

## Inheritance

[object](#) ← ExtTextManager

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ExtTextManager()

```
public ExtTextManager()
```

## Methods

### AddToExtList<T>(List<T>)

Adds a specific type of list to the corresponding [ExtList](#). Type must be a valid Master.Param.

```
public static void AddToExtList<T>(List<T> list) where T : class
```

## Parameters

**list** [List](#)<T>

The list to be added to the corresponding [ExtList](#).

## Type Parameters

T

The type of list to add. Must be a valid Master.Param

## Remarks

This method is required to load custom text into the game.

## GetUniqueIdNum<T>(string)

Gets the unique Id Number at the end of a param's Id, and converts it to a supported enumtype.

- JineType
- KusoRepType
- TooltipType
- TweetType

```
public static T GetUniqueIdNum<T>(string id) where T : Enum
```

## Parameters

id [string](#)

The param's Id to be used for conversion.

## Returns

T

## Type Parameters

T

The enumtype to convert to.

## Exceptions

[ArgumentException](#)

Thrown if the type is either not supported or is not an enumtype.

## [InvalidOperationException](#)

Thrown if the unique Id Number failed to parse.

# Class JineExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class JineExtender
```

## Inheritance

[object](#) ← JineExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### JineExtender()

```
public JineExtender()
```

## Methods

### ResetExtReplyList()

Re-initializes custom Jine Replies.

```
public static void ResetExtReplyList()
```

### StartAmeJine(JineType)

Create a Jine message sent by Ame.

```
public static UniTask StartAmeJine(JineType jineData)
```

## Parameters

**jineData** JineType

The JineType used to load the message. When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

## Returns

UniTask

## Exceptions

[NullReferenceException](#)

## StartExtAmeJine(string)

Create a custom user-made Jine message sent by Ame.

```
public static UniTask StartExtAmeJine(string id)
```

## Parameters

**id** string

The LineMaster.Param's Id used to load the message.

## Returns

UniTask

## Exceptions

[NullReferenceException](#)

# StartExtPiListAndSet(string, Action, string, Action, string, Action, string, Action, string, Action)

Creates a list of chooseable options on Jine, then sends one custom user-made Jine message from P-chan based on the option chosen. Up to five options can be created. Using async methods for the `Action` parameters are recommended.

```
public static void StartExtPiListAndSet(string optionOneId, Action actionOne, string optionTwoId = null, Action actionTwo = null, string optionThreeId = null, Action actionThree = null, string optionFourId = null, Action actionFour = null, string optionFiveId = null, Action actionFive = null)
```

## Parameters

`optionOneId` [string](#)

The LineMaster.Param's Id used to load the first option.

`actionOne` [Action](#)

The `Action` method related to the first option.

`optionTwoId` [string](#)

The LineMaster.Param's Id used to load the second option. (optional)

`actionTwo` [Action](#)

The `Action` method related to the second option. Required if the second option Id is not null or "".

`optionThreeId` [string](#)

The LineMaster.Param's Id used to load the third option. (optional)

`actionThree` [Action](#)

The `Action` method related to the third option. Required if the third option Id is not null or "".

`optionFourId` [string](#)

The LineMaster.Param's Id used to load the fourth option. (optional)

`actionFour` [Action](#)

The `Action` method related to the fourth option. Required if the fourth option Id is not null or "".

## **optionFiveId** [string ↗](#)

The LineMaster.Param's Id used to load the fifth option. (optional)

## **actionFive** [Action ↗](#)

The **Action** method related to the fifth option. Required if the fifth option Id is not null or "".

## Remarks

Note: If an optional option Id is not null, its related **Action** must be filled, or else the option won't appear on the list. (i.e. filling in the second option Id without including its related **Action**.)

## Exceptions

### [NullReferenceException ↗](#)

## StartExtPiOption(string, Action)

Sends one custom user-made Jine message from P-chan based on the option chosen from **StartExtPiOptionList**. Using async methods for the **Action** parameters are recommended.

```
public static void StartExtPiOption(string id, Action action)
```

## Parameters

### **id** [string ↗](#)

The LineMaster.Param's Id used to load the message.

### **action** [Action ↗](#)

The **Action** method after this message is sent.

## StartExtPiOptionList(string, string, string, string, string)

Creates a list of chooseable options on Jine. Use **SetExtPiOption** after based on the options listed.

```
public static void StartExtPiOptionList(string optionOneId, string optionTwoId = null,  
string optionThreeId = null, string optionFourId = null, string optionFiveId = null)
```

## Parameters

### optionOneId [string](#)

The LineMaster.Param's Id used to load the first option.

### optionTwoId [string](#)

The LineMaster.Param's Id used to load the second option. (optional)

### optionThreeId [string](#)

The LineMaster.Param's Id used to load the third option. (optional)

### optionFourId [string](#)

The LineMaster.Param's Id used to load the fourth option. (optional)

### optionFiveId [string](#)

The LineMaster.Param's Id used to load the fifth option. (optional)

## StartPiListAndSet(JineType, Action, JineType, Action, JineType, Action, JineType, Action, JineType, Action, JineType, Action)

Creates a list of chooseable options on Jine, then sends one custom user-made Jine message from P-chan based on the option chosen. Up to five options can be created. Using async methods for the **Action** parameters are recommended.

```
public static void StartPiListAndSet(JineType optionOne, Action actionOne, JineType  
optionTwo = JineType.None, Action actionTwo = null, JineType optionThree = JineType.None,  
Action actionThree = null, JineType optionFour = JineType.None, Action actionFour = null,  
JineType optionFive = JineType.None, Action actionFive = null)
```

## Parameters

### optionOne JineType

The JineType used to load the first option. When loading in a custom message, either use [ExtTextExtender.GetUniqueIdNum\(\)](#), or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

### actionOne Action

The **Action** method related to the first option.

#### **optionTwo** JineType

The JineType used to load the second option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

#### **actionTwo** [Action ↗](#)

The **Action** method related to the second option. Required if the second option Id is not null or "".

#### **optionThree** JineType

The JineType used to load the third option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

#### **actionThree** [Action ↗](#)

The **Action** method related to the third option. Required if the third option Id is not null or "".

#### **optionFour** JineType

The JineType used to load the fourth option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

#### **actionFour** [Action ↗](#)

The **Action** method related to the fourth option. Required if the fourth option Id is not null or "".

#### **optionFive** JineType

The JineType used to load the fifth option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

#### **actionFive** [Action ↗](#)

The **Action** method related to the fifth option. Required if the fifth option Id is not null or "".

## Remarks

Note: If an optional option Id is not null, its related **Action** must be filled, or else the option won't appear on the list. (i.e. filling in the second option Id without including its related **Action**.)

## Exceptions

[NullReferenceException ↗](#)

## StartPiOption(JineType, Action)

Sends one custom user-made Jine message from P-chan based on the option chosen from [StartExtPiOptionList](#). Using async methods for the [Action](#) parameters are recommended.

```
public static void StartPiOption(JineType jineData, Action action)
```

### Parameters

**jineData** JineType

The JineType used to load the message. When loading in a custom message, either use [ExtTextExtender.GetUniqueIdNum\(\)](#), or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

**action** [Action ↗](#)

The [Action](#) method after this message is sent.

## StartPiOptionList(JineType, JineType, JineType, JineType, JineType)

Creates a list of chooseable options on Jine. Use [SetPiOption](#) after based on the options listed.

```
public static void StartPiOptionList(JineType optionOne, JineType optionTwo = JineType.None,
JineType optionThree = JineType.None, JineType optionFour = JineType.None, JineType
optionFive = JineType.None)
```

### Parameters

**optionOne** JineType

The JineType used to load the first option. When loading in a custom message, either use [ExtTextExtender.GetUniqueIdNum\(\)](#), or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

### **optionTwo** JineType

The JineType used to load the second option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

### **optionThree** JineType

The JineType used to load the third option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

### **optionFour** JineType

The JineType used to load the fourth option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

### **optionFive** JineType

The JineType used to load the fifth option. (optional)

When loading in a custom message, either use `ExtTextExtender.GetUniqueIdNum()`, or use your unique number at the end of your ID and add 10000 to it, then cast it as a JineType.

## StartTraumaList(Action, Action)

Loads a separate event where you have to choose the right trauma, if applicable.

```
public static void StartTraumaList(Action isRight, Action isWrongOrNull)
```

### Parameters

#### **isRight** [Action](#)

The `Action` that happens if you chose the right trauma.

#### **isWrongOrNull** [Action](#)

The `Action` that happens if you chose the wrong trauma, or if she didn't tell you any of her traumas on Day 15.

### Remarks

Using async **Actions** are recommended here.

## StartWrittenMsg(Action)

Starts a prompt that requires you to write a message to Ame on Jine.

```
public static void StartWrittenMsg(Action eventAfterMsg)
```

Parameters

**eventAfterMsg** [Action](#)

The action that happens after the message has been sent.

## StartWrittenMsg(Func<string, bool>, Action, Action)

Starts a prompt that requires you to write a message to Ame on Jine.

```
public static void StartWrittenMsg(Func<string, bool> checkMsgMatch, Action isRight,
Action isWrong)
```

Parameters

**checkMsgMatch** [Func](#)<[string](#), [bool](#)>

A **Func** bool that does something with the written message.

**isRight** [Action](#)

The **Action** that happens when **checkMsgMatch** returns true.

**isWrong** [Action](#)

The **Action** that happens when **checkMsgMatch** returns false.

Remarks

This version of the method checks the message using **checkMsgMatch**, where the remainder of the event changes depending on what **checkMsgMatch** returns as.

# Class KRepExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class KRepExtender
```

## Inheritance

[object](#) ← KRepExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

KRepExtender()

```
public KRepExtender()
```

## Methods

ResetExtKReps()

Re-initializes custom Tweet Replies.

```
public static void ResetExtKReps()
```

# Class KitsuneExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class KitsuneExtender
```

## Inheritance

[object](#) ← KitsuneExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## KitsuneExtender()

```
public KitsuneExtender()
```

# Methods

## IsCustomKitsuneThread(bool, string, bool)

Sets if an /st/ thread is custom or not.

```
public static void IsCustomKitsuneThread(bool isCustom, string topic = null, bool isCanPost
= true)
```

## Parameters

**isCustom** [bool](#)

It is a custom thread if true, otherwise loads the normal thread.

**topic** [string](#)

The KituneMaster.Param and KitsuneSuretai.Param

**isCanPost** [bool](#)

## OpenCustomKituneWindow(bool, string)

Opens a Window with a custom /st/ thread made by the user. Posting in /st/ is disabled.

```
public static void OpenCustomKituneWindow(bool isMini, string topic)
```

### Parameters

**isMini** [bool](#)

Is smaller and placed at a random position if true.

Is normal sized otherwise.

**topic** [string](#)

The custom topic of the /st/ thread.

### Remarks

**IsCustomKitsuneThread** is set to false after using this method.

# Class KusoComExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class KusoComExtender
```

## Inheritance

[object](#) ← KusoComExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## KusoComExtender()

```
public KusoComExtender()
```

# Class MobComExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class MobComExtender
```

## Inheritance

[object](#) ← MobComExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## MobComExtender()

```
public MobComExtender()
```

# Class SysTxtExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class SysTxtExtender
```

## Inheritance

[object](#) ← SysTxtExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## SysTxtExtender()

```
public SysTxtExtender()
```

# Methods

## OpenCustomNote(string, string)

Opens a Notepad window that contains custom text.

```
public static void OpenCustomNote(string contextId, string nameId)
```

### Parameters

contextId [string](#)

The SystemTextMaster.Param's Id used to load the text. Can use either in-game or custom params.

## **nameId** [string](#)

The SystemTextMaster.Param's Id used to set the name of the window. Can use either in-game or custom params.

## Remarks

Note: This window won't open if "Ame's Notes" is open, as this uses "Ame's Notes" as a window base. "Ame's Notes" also won't open if this custom window is currently open.

# Class TenComExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class TenComExtender
```

## Inheritance

[object](#) ← TenComExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## TenComExtender()

```
public TenComExtender()
```

# Class TooltxtExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class TooltxtExtender
```

## Inheritance

[object](#) ← TooltxtExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### TooltxtExtender()

```
public TooltxtExtender()
```

## Methods

### ShowExtTooltip(bool, string)

Creates a Tooltip with custom text.

```
public static void ShowExtTooltip(bool isCat, string id)
```

#### Parameters

**isCat** [bool](#)

Shows the Cat Tooltip if true. Otherwise, shows the normal speech bubble tooltip.

**id** [string](#)

The TooltipMaster.Param's Id used to load the tooltip text.

## ShowTooltip(bool, TooltipType)

Creates a Tooltip with either original or custom text.

```
public static void ShowTooltip(bool isCat, TooltipType tooltxtData)
```

### Parameters

**isCat** [bool](#)

Shows the Cat Tooltip if true. Otherwise, shows the normal speech bubble tooltip.

**tooltxtData** TooltipType

The TooltipType used to load text. If loading custom text, either use

[ExtTextExtender.GetUniqueIdNum\(\)](#), or use your unique number at the end of your ID and add 10000 to it, then cast it as a TooltipType.

# Class TweetExtender

Namespace: [NGOTxtExtender](#)

Assembly: NGOTxtExtender.dll

```
[HarmonyPatch]
public class TweetExtender
```

## Inheritance

[object](#) ← TweetExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## TweetExtender()

```
public TweetExtender()
```

# Methods

## StartExtTweet(string)

Creates a custom Tweet on Tweeter/Poketter with no replies attached to it.

```
public static void StartExtTweet(string id)
```

## Parameters

**id** [string](#)

The TweetMaster.Param's Id used to load the tweet.

## Exceptions

[NullReferenceException ↗](#)

## StartExtTweetWithExtReps(string, string)

Creates a custom Tweet on Tweeter/Poketter with custom replies attached to it. These replies are pre-set and are not randomized. Tweet replies are loaded based on their **ParentID**.

```
public static void StartExtTweetWithExtReps(string tweetId, string kusoRepParId)
```

## Parameters

**tweetId** [string ↗](#)

The TweetMaster.Param's Id used to load the tweet.

**kusoRepParId** [string ↗](#)

The KRepMaster.Param's **ParentID** that's used to find matching params.

## Remarks

Note: Tweet Replies loads the first param at the bottom of the reply thread, and vice-versa.

## Exceptions

[NullReferenceException ↗](#)

## StartExtTweetWithPresetReps(string, List<KusoRepType>)

Creates a custom Tweet on Tweeter/Poketter with replies attached to it. These replies are pre-set and are not randomized. Tweet replies are loaded based on their **ParentID**.

```
public static void StartExtTweetWithPresetReps(string tweetId, List<KusoRepType> repList)
```

## Parameters

**tweetId** [string ↗](#)

The TweetMaster.Param's Id used to load the tweet.

### repList [List](#)<KusoRepType>

The list of KusoRepType's used to load in the pre-set tweet replies. You can also place custom tweet replies in the list by either using [ExtTextExtender.GetUniqueIdNum\(\)](#) or by using your unique number at the end of your ID and add 10000 to it, then cast it as a KusoRepType.

## Exceptions

### [NullReferenceException](#)

## StartExtTweetWithRandomReps(string)

Creates a custom Tweet on Tweeter/Poketter with normal (random) replies attached to it.

```
public static void StartExtTweetWithRandomReps(string id)
```

## Parameters

### id [string](#)

The TweetMaster.Param's Id used to load the tweet.

## Exceptions

### [NullReferenceException](#)

## StartTweet(TweetType)

Creates a Tweet on Tweeter/Poketter with no replies attached to it.

```
public static void StartTweet(TweetType tweet)
```

## Parameters

### tweet TweetType

The TweetType used to load the tweet. If loading custom text, either use `ExtTextExtender.GetUniqueIdNum()` or use your unique number at the end of your ID and add 10000 to it, then cast it as a TweetType.

## StartTweetWithExtReps(TweetType, string)

Creates a normal or custom Tweet on Tweeter/Poketter with custom replies attached to it. These replies are pre-set and are not randomized. Tweet replies are loaded based on their `ParentID`.

```
public static void StartTweetWithExtReps(TweetType tweet, string kusoRepParId)
```

### Parameters

`tweet` TweetType

The TweetType used to load the tweet. If loading custom text, either use `ExtTextExtender.GetUniqueIdNum()` or use your unique number at the end of your ID and add 10000 to it, then cast it as a TweetType.

`kusoRepParId` [string](#)

The KRepMaster.Param's `ParentID` that's used to find matching params.

### Remarks

Note: Tweet Replies loads the first param at the bottom of the reply thread, and vice-versa.

### Exceptions

[NullReferenceException](#)

## StartTweetWithPresetReps(TweetType, List<KusoRepType>)

Creates a normal or custom Tweet on Tweeter/Poketter with custom replies attached to it. These replies are pre-set and are not randomized. Tweet replies are loaded based on their `ParentID`.

```
public static void StartTweetWithPresetReps(TweetType tweet, List<KusoRepType> repList)
```

### Parameters

## **tweet** TweetType

The TweetType used to load the tweet. If loading custom text, either use `ExtTextExtender.GetUniqueIdNum()` or use your unique number at the end of your ID and add 10000 to it, then cast it as a TweetType.

## **repList** [List](#)<KusoRepType>

The list of KusoRepType's used to load in the pre-set tweet replies. You can also place custom tweet replies in the list by either using `ExtTextExtender.GetUniqueIdNum()` or by using your unique number at the end of your ID and add 10000 to it, then cast it as a KusoRepType.

## Remarks

Note: Tweet Replies loads the first param at the bottom of the reply thread, and vice-versa.

## Exceptions

### [NullReferenceException](#)

## StartTweetWithRandomReps(TweetType)

Creates a custom Tweet on Tweeter/Poketter with normal (random) replies attached to it.

```
public static void StartTweetWithRandomReps(TweetType tweet)
```

## Parameters

### **tweet** TweetType

The TweetType used to load the tweet. If loading custom text, either use `ExtTextExtender.GetUniqueIdNum()` or use your unique number at the end of your ID and add 10000 to it, then cast it as a TweetType.

# Namespace NSOMediaExtender

## Classes

[AddressableExtender](#)

[AssetBundleLoader](#)

[AudioExtender](#)

[ExternalFileLoader](#)

[ExternalFileLoader.FileLoadFailedException](#)

The exception that is thrown if an error occurred in loading the requested file.

[PictureExtender](#)

[WebCamExtender](#)

## Enums

[IdleAnimationType](#)

[WebcamType](#)

The idle animations Ame does based on her Task Manager Stats, with the exception of **Horror**.

# Class AddressableExtender

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
public class AddressableExtender
```

## Inheritance

[object](#) ← AddressableExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### AddressableExtender()

```
public AddressableExtender()
```

## Methods

### AddAddressBundle(string)

Loads an Asset Bundle related to an Addressable catalog.

```
public static UniTask AddAddressBundle(string path)
```

#### Parameters

**path** [string](#)

The path to the Addressable Asset Bundle.

#### Returns

## Remarks

This temporarily copies the Asset Bundle into the StreamingAssets folder of the game until the game closes.

Note: The Asset Bundle will not delete itself in the case of the game crashing or force quitted through Task Manager. (However, it will delete itself if the game is closed properly next time.)

## AddExternalCatalog(string)

Loads an external Addressable catalog file into the game. If you're relying on Addressables to load in external assets, this is required.

```
public static UniTask AddExternalCatalog(string path)
```

### Parameters

**path** [string](#)

The path to the external catalog.

### Returns

UniTask

## LoadAddressAsset<T>(string)

Loads an asset from Addressables into the game.

```
public static UniTask<T> LoadAddressAsset<T>(string address) where T : Object
```

### Parameters

**address** [string](#)

The address of the asset Addressable.

### Returns

UniTask<T>

## Type Parameters

T

The type of `UnityEngine.Object` to load.

## Remarks

For easy releasing, you can use `ReleaseAddressAsset` or `ReleaseAllHandles`, but only if it was loaded through this method.

## Exceptions

[Exception ↗](#)

## ReleaseAddressAsset(string)

Releases an object Addressable's handle loaded with `LoadAddressAsset()`.

```
public static void ReleaseAddressAsset(string address)
```

## Parameters

address [string ↗](#)

## ReleaseAllHandles()

Releases all handles loaded with `LoadAddressAsset()`.

```
public static void ReleaseAllHandles()
```

# Class AssetBundleLoader

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
public class AssetBundleLoader
```

## Inheritance

[object](#) ← AssetBundleLoader

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### AssetBundleLoader()

```
public AssetBundleLoader()
```

## Methods

### LoadCustomAssets(string)

Loads specific assets from an Asset Bundle, then unloads the rest of the assets.

These specific assets include:

**AnimationClip** assets that starts with "stream\_cho\_" or "stream\_ame\_" .

**AudioClip** assets that starts with "BGM\_" or "SE\_" .

**Sprite** assets that starts with "MyPic\_" , specifically for My Pictures.

```
public static UniTask LoadCustomAssets(string path)
```

## Parameters

**path** [string](#)

The path to the asset bundle.

Returns

UniTask

Exceptions

[NullReferenceException](#)

# Class AudioExtender

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
[HarmonyPatch]
public class AudioExtender
```

## Inheritance

[object](#) ← AudioExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

# Constructors

## AudioExtender()

```
public AudioExtender()
```

# Methods

## NewSoundFromClip(AudioClip)

Creates a new [Sound](#) from an [AudioClip](#), and adds it to the Custom Sound List.

```
public static void NewSoundFromClip(AudioClip clip)
```

## Parameters

[clip](#) AudioClip

The [AudioClip](#) used to make the [Sound](#).

## PlayCustomSound(string)

Plays a custom [Sound](#) from the Custom Sound List.

```
public static void PlayCustomSound(string Id)
```

### Parameters

**Id** [string](#)

The [Sound](#) that you want to play, using their Id.

# Class ExternalFileLoader

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
public class ExternalFileLoader
```

## Inheritance

[object](#) ← ExternalFileLoader

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### ExternalFileLoader()

```
public ExternalFileLoader()
```

## Methods

### LoadAudioFromFile(string, AudioType)

Loads a custom audio file as an Audio Clip.

```
public static UniTask<AudioClip> LoadAudioFromFile(string path, AudioType type)
```

## Parameters

**path** [string](#)

The path to the audio file.

**type** [AudioType](#)

The type of audio file.

Returns

UniTask<AudioClip>

Remarks

Supported audio types include .mp3, .wav, .ogg, and more.

Exceptions

[ArgumentNullException](#)

[ExternalFileLoader.FileLoadFailedException](#)

[FileNotFoundException](#)

## LoadImageFromFile(string)

Loads a custom image file as a Sprite.

```
public static Sprite LoadImageFromFile(string path)
```

Parameters

**path** [string](#)

The path to the image file.

Returns

Sprite

Remarks

Supported image types are jpg, and png.

Exceptions

[ArgumentNullException](#)

[ExternalFileLoader.FileLoadFailedException](#)

[FileNotFoundException](#) ↗

# Class ExternalFileLoader.FileLoadFailedException

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

The exception that is thrown if an error occurred in loading the requested file.

```
public class ExternalFileLoader.FileLoadFailedException : Exception,  
ISerializable, _Exception
```

## Inheritance

[object](#) ← [Exception](#) ← ExternalFileLoader.FileLoadFailedException

## Implements

[ISerializable](#), [Exception](#)

## Inherited Members

[Exception.GetBaseException\(\)](#), [Exception.ToString\(\)](#),  
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#), [Exception.GetType\(\)](#),  
[Exception.Message](#), [Exception.Data](#), [Exception.InnerException](#), [Exception.TargetSite](#),  
[Exception.StackTrace](#), [Exception.HelpLink](#), [Exception.Source](#), [Exception.HResult](#),  
[Exception.SerializeObjectState](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.MemberwiseClone\(\)](#)

## Constructors

### FileLoadFailedException()

```
public FileLoadFailedException()
```

### FileLoadFailedException(string)

```
public FileLoadFailedException(string message)
```

Parameters

message [string](#)

## FileLoadFailedException(string, Exception)

```
public FileLoadFailedException(string message, Exception inner)
```

Parameters

message [string](#)

inner [Exception](#)

# Enum IdleAnimationType

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
public enum IdleAnimationType
```

## Fields

Anxious = 3

Happy = 1

Irritated = 2

Normal = 0

# Class PictureExtender

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
[HarmonyPatch]
public class PictureExtender
```

## Inheritance

[object](#) ← PictureExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### PictureExtender()

```
public PictureExtender()
```

## Methods

### AddExtSpriteForMyPics(Sprite)

Adds a [Sprite](#) to the [Sprite](#) Extlist.

```
public static void AddExtSpriteForMyPics(Sprite sprite)
```

#### Parameters

[sprite](#) Sprite

The sprite to load.

## CreateExtResWithAddress(string)

Creates a ResourceLocal object, the reference the game uses to load an image from Tweeter/Poketter, Jine, and MyPictures.

Only use this if you're relying on Addressables to load images.

```
public static void CreateExtResWithAddress(string address)
```

### Parameters

**address** [string](#)

The image's addressable Id, which the game will use to load the image.

## ExtSpriteToResLocal()

Uses the [Sprite](#) ExtList to create related ResourceLocal objects, the reference the game uses to load an image from Tweeter/Poketter, Jine, and MyPictures.

```
public static void ExtSpriteToResLocal()
```

# Class WebCamExtender

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

```
[HarmonyPatch]
public class WebCamExtender
```

## Inheritance

[object](#) ← WebCamExtender

## Inherited Members

[object.ToString\(\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#)

## Constructors

### WebCamExtender()

```
public WebCamExtender()
```

## Methods

### AddCustomAnimClip(List<AnimationClip>)

Adds [AnimationClips](#) to the [AnimationClip](#) ExtList.

```
public static void AddCustomAnimClip(List<AnimationClip> clips)
```

#### Parameters

[clips](#) [List](#)<AnimationClip>

The [AnimationClips](#) to add to the list.

## AddCustomAnimClip(AnimationClip)

Adds an [AnimationClip](#) to the [AnimationClip](#) ExtList.

```
public static void AddCustomAnimClip(AnimationClip clip)
```

### Parameters

**clip** AnimationClip

The [AnimationClip](#) to add to the list.

## AddCustomIdleAnims(List<string>, WebcamType)

Adds new idle animations to Ame.

```
public static void AddCustomIdleAnims(List<string> extAnims, WebcamType type)
```

### Parameters

**extAnims** [List](#)<[string](#)>

The list of animations to add, based on their [AnimationClip](#) name.

**type** [WebcamType](#)

The idle animation list to add to.

## PlayNegativeAme(bool)

Plays an animation of Ame reacting negatively to something in the Webcam. The kind of animation that's played depends on her stats.

```
public static void PlayNegativeAme(bool isScary = false)
```

### Parameters

**isScary** [bool](#)

If true, automatically plays the scary version of the animation, regardless of her stats.

## PlayPositiveAme(bool)

Plays an animation of Ame reacting positively to something in the Webcam. The kind of animation that's played depends on her stats.

```
public static void PlayPositiveAme(bool isScary = false)
```

### Parameters

**isScary** [bool](#)

If true, automatically plays the scary version of the animation, regardless of her stats.

## PlaySpecificIdlingAme(IdleAnimationType, bool, bool)

Plays a specific idling animation of Ame. The kind of animation that's played depends on her stats.

```
public static void PlaySpecificIdlingAme(IdleAnimationType anim, bool setBaseAnim = false,  
bool isScary = false)
```

### Parameters

**anim** [IdleAnimationType](#)

The IdleAnimationType.

**setBaseAnim** [bool](#)

If true, the current animation defaults to this one. Otherwise, only plays this animation once.

**isScary** [bool](#)

If true, automatically plays the scary version of the animation, regardless of her stats.

## ReplaceCustomIdleAnims(List<string>, WebcamType)

Replaces default idle animations of Ame to custom ones.

```
public static void ReplaceCustomIdleAnims(List<string> extAnims, WebcamType type)
```

## Parameters

**extAnims** [List](#) <[string](#)>

The list of animations to replace with, based on their [AnimationClip](#) name.

**type** [WebcamType](#)

The idle animation list to replace.

## Remarks

Note: If two or more different mods use this method, the mod loaded last will take priority on what animations to replace.

# Enum WebcamType

Namespace: [NSOMediaExtender](#)

Assembly: NSOMediaExtender.dll

The idle animations Ame does based on her Task Manager Stats, with the exception of Horror.

```
public enum WebcamType
```

## Fields

Dark = 2

Darker = 3

DarkerLove = 10

Horror = 12

Like = 6

Love = 7

Normal = 0

StressDark = 4

StressDarker = 5

StressDarkerLove = 11

StressLike = 8

StressLove = 9

Stressed = 1