Unity Asset Documentation

Thank you for your interest in our assets!

Please note, that our assets always include PDF-Files as this is a requirement from Unity.

We have an up-to-date online documentation that is much more convenient to use, available here.

Namespace Firesplash.GameDevAssets.Socket IOPlus

Classes

DataTypes

Contains simple DataTypes used for Socket.IO communication and states

DataTypes.SocketIOErrorPayload

A Payload template to mimic JS "Error" events. Error messages are contained in the "message" property

DataTypes.SocketIOException

This is the base class all Socket.IO-Exceptions derive from

DataTypes.SocketIOProtocolViolationException

This exception is thrown, when the application is trying to violate a protocol constraint. There are only rare possibilities to do so, though. This exception is NOT thrown for incoming packages!

DefaultParser

This implemented the default Socket.IO parser to parse string typed EngineIO messages into Socket.IO and encode packets vice versa.

Parser

This is a skeleton class for writing Socket.IO / Engine.IO transcoders

SocketIOClient

This MonoBehavior derives from EngineIOClient and implements the main Socket.IO Manager logic on top of Engine.IO. It implements Socket.IO protocol version 5 (which is used by Socket.IO v3 and v4)

SocketIOEvent

SocketIONamespace

This class represents a (usually connected) Socket.IO namespace and implements the EventEmitter and the EventReceiver. We try to keep our API as near to the official Socket.IO V4 API as possible within this class. https://socket.io/docs/v4/emitting-events/https://socket.io/docs/v4/listening-to-events/#eventemitter-methods

SocketIOPacket

This class represents a lowlevel SocketIO packet in its parsed state.

Enums

DataTypes.ConnectionState

DataTypes.PacketType

Delegates

DataTypes.SocketIOAuthPayloadCallback

This delegate gets called when a Socket.IO namespace is being connected. Your function must return null if not auht payload is required for the namespace, or an object that can be serialized by Json.Net if a payload must be provided

DataTypes.ThreadedSocketIOEvent

This delegate gets called on a namespace for any Socket.IO event (received or internally generated). The delegate is invoked from a Thread so it is not safe to access Unity functions from it.

Class DataTypes

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

Contains simple DataTypes used for Socket.IO communication and states

```
public static class DataTypes
```

Inheritance

object ← DataTypes

Enum DataTypes.ConnectionState

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public enum DataTypes.ConnectionState
```

Fields

`CONNECTED = 2`

`CONNECTING = 1`

`DISCONNECTED = 3`

`NONE = 0`

Enum DataTypes.PacketType

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public enum DataTypes.PacketType
```

Fields

`ACK = 3`

`BINARY_ACK = 6`

`BINARY_EVENT = 5`

`CONNECT = 0`

`CONNECT_ERROR = 4`

`DISCONNECT = 1`

`EVENT = 2`

Delegate DataTypes.SocketIOAuthPayload Callback

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This delegate gets called when a Socket.IO namespace is being connected. Your function must return null if not auht payload is required for the namespace, or an object that can be serialized by Json.Net if a payload must be provided

```
public delegate object DataTypes.SocketIOAuthPayloadCallback(string namespacePath)
```

Parameters

`namespacePath` string

The Socket.IO namespace path (e.g. "/") for which authentication data is requested

Returns

object

This delegate gets called when a Socket.IO namespace is being connected. Your function must return null if not auht payload is required for the namespace, or an object that can be serialized by Json.Net if a payload must be provided

Class DataTypes.SocketIOErrorPayload

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

A Payload template to mimic JS "Error" events. Error messages are contained in the "message" property

```
[Serializable]
public class DataTypes.SocketIOErrorPayload
```

Inheritance

object ← DataTypes.SocketIOErrorPayload

Constructors

SocketIOErrorPayload(string)

Creates a payload template to mimic JS "Error" events

```
public SocketIOErrorPayload(string message)
```

Parameters

`message` string

The message contained in the object

Fields

message

```
public string message
```

Field Value

string

Class DataTypes.SocketIOException

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This is the base class all Socket.IO-Exceptions derive from

```
public class DataTypes.SocketIOException : Exception
```

Inheritance

object ← Exception ← DataTypes.SocketIOException

Derived

DataTypes.SocketIOProtocolViolationException

Constructors

SocketIOException(string)

```
public SocketIOException(string message)
```

Parameters

`message` string

Class DataTypes.SocketIOProtocolViolation Exception

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This exception is thrown, when the application is trying to violate a protocol constraint. There are only rare possibilities to do so, though. This exception is NOT thrown for incoming packages!

```
public class DataTypes.SocketIOProtocolViolationException : DataTypes.SocketIOException
```

Inheritance

object ← Exception ← DataTypes.SocketIOException ← DataTypes.SocketIOProtocolViolationException

Constructors

SocketIOProtocolViolationException(string)

```
public SocketIOProtocolViolationException(string message)
```

Parameters

`message` string

Delegate DataTypes.ThreadedSocketIOEvent

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This delegate gets called on a namespace for any Socket.IO event (received or internally generated). The delegate is invoked from a Thread so it is not safe to access Unity functions from it.

```
public delegate void DataTypes.ThreadedSocketIOEvent(SocketIOEvent sioEvent)
```

Parameters

`sioEvent` SocketIOEvent

Class DefaultParser

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This implemented the default Socket.IO parser to parse string typed EngineIO messages into Socket.IO and encode packets vice versa.

```
public class DefaultParser : Parser
```

Inheritance

object ← Parser ← DefaultParser

Methods

Parse(EngineIOPacket, SocketIOClient)

This creates a Socket.IO packet from the string types Engine.IO message Binary payloads can then be added to the package.

```
public override SocketIOPacket Parse(EngineIOPacket eioPacket, SocketIOClient client)
```

Parameters

`eioPacket` EngineIOPacket

`client` SocketIOClient

Returns

SocketIOPacket

Class Parser

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This is a skeleton class for writing Socket.IO / Engine.IO transcoders

```
public class Parser
```

Inheritance

object ← Parser

Derived

DefaultParser

Methods

Parse(EngineIOPacket, SocketIOClient)

This creates a Socket.IO packet from the string types Engine.IO message Binary payloads can then be added to the package.

```
public virtual SocketIOPacket Parse(EngineIOPacket eioPacket, SocketIOClient client)
```

Parameters

`eioPacket` EngineIOPacket

The Engine.IO packet to parse

`client` SocketIOClient

A reference to the Scoket.IO client

Returns

SocketIOPacket

A Socket.IO packet instance

Class SocketIOClient

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This MonoBehavior derives from EngineIOClient and implements the main Socket.IO Manager logic on top of Engine.IO. It implements Socket.IO protocol version 5 (which is used by Socket.IO v3 and v4)

```
[DisallowMultipleComponent]
[AddComponentMenu("Networking/Socket.IO/Socket.IO Client")]
public class SocketIOClient : EngineIOClient
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← EngineIOClient ←
SocketIOClient

Inherited Members

EngineIOClient.serverAddress , EngineIOClient.volatileOperationMode , EngineIOClient.State , EngineIOClient.OnEngineIOMessageReceived , EngineIOClient.OnEngineIOConnectionReady , EngineIOClient.OnEngineIODisconnect , EngineIOClient.OnEngineIOError , EngineIOClient.OnEngineIOMessageReceivedThreaded , EngineIOClient.OnEngineIOConnectionReadyThreaded , EngineIOClient.Awake() , EngineIOClient.Connect(string, bool) , EngineIOClient.SendEngineIOMessage(string) , EngineIOClient.SendEngineIOMessage(byte[]) , EngineIOClient.SendEngineIOPacket(EngineIOPacket) , EngineIOClient.SendEngineIOPackets(EngineIOPacket[]) , EngineIOClient.SendQueueLength , EngineIOClient.ClearSendQueue()

Fields

maxConnectAttempts

If the connection (or a reconnect) is not successful after n attempts, cancel trying to (re)connect. A value of zero means infinitely.

```
public int maxConnectAttempts
```

Field Value

int

raisingReconnectDelay

After a failure, (re)connection attempts will be delayed by initially one second. On every failure the delay is raised by 50% and ceiled (1, 2, 3, 5, 8, 12, ...) up to 60 seconds. On a successful (re)connect this delay is reset to one second. If you set this value to false, the delay will always be one second +/-20% jitter.

```
public bool raisingReconnectDelay
```

Field Value

bool

Properties

D

This is a shorthand to access the default namespace without having to write the whole namespace call every time for simple applications.

```
public SocketIONamespace D { get; }
```

Property Value

SocketIONamespace

DefaultNamespace

This is a shorthand to access the default namespace without having to write the whole namespace call every time for simple applications. Want it even shorter? Use "D" :o)

```
public SocketIONamespace DefaultNamespace { get; }
```

Property Value

SocketIONamespace

Methods

Connect(string)

Connect the client to the server

```
public override void Connect(string pServerAddress = null)
```

Parameters

`pServerAddress` string

Disconnect()

Disconnect the Engine.IO client

```
public override void Disconnect()
```

GetNamespace(string, bool)

Returns the API of the Socket.IO Client for the given namesapce and connects to the namespace if it is not already connected. If the underlaying transport is not completely connected yet, the connect to the namespace is delayed until the transport is ready. Namespaces will always reconnect when the manager reconnects, unless you directly call "Disconnect()" on the namespace itself. A Namespace that has been disconnected directly, can be reconnected by calling "Connect()" on the namespace.

```
public SocketIONamespace GetNamespace(string namespacePath, bool connectIfNotExists = true)
```

Parameters

`namespacePath` string

`connectIfNotExists` bool

Connect to this namespace if it is not connected (returns null if false and not existing)

Returns

SocketIONamespace

GetParser()

```
protected virtual Parser GetParser()
```

Returns

Parser

LateUpdate()

```
protected void LateUpdate()
```

Off(string, UnityAction<object>)

Unregisters a previously registered manager event callback

```
public void Off(string eventName, UnityAction<object> callback)
```

Parameters

`eventName` string

The event name - For valid values see On(...)

`callback` UnityAction<object>

The callback to unregister

Exceptions

DataTypes.SocketIOException

Thrown, if the given eventName is not a valid Manager event

On(string, UnityAction<object>)

Allows registering to "low level" manager events. This is NOT a namespaced event listener!

```
public void On(string eventName, UnityAction<object> callback)
```

Parameters

`eventName` string

The event name (one of error, reconnect, reconnect_attempt, reconnect_error, reconnect_failed)

`callback` UnityAction<object>

The callback to be called. The parameter contains values according to the official Socket.IO documentation. The Error event has a string. For events having no payload, the value is null.

Exceptions

DataTypes.SocketIOException

Thrown, if the given eventName is not a valid Manager event

SetAuthPayloadCallback(SocketIOAuthPayloadCallback)

This callback will be called whenever a namespace connects. If the callback returns a value other than null, it will be sent as authentication payload while connecting the namespace. The function is called from GetNamespace, so if you call this method from a thread, the callback also runs on a thread. Internally generated connect sequences always call the callback from the main thread.

```
public void SetAuthPayloadCallback(DataTypes.SocketIOAuthPayloadCallback callback)
```

Parameters

`callback` DataTypes.SocketIOAuthPayloadCallback

A SocketIOAuthPayloadCallback delegate

SetParser(Parser)

You can override the used parser using this method. You can implement your own (or a publicly available) parser and message format. Please remember, that server and clients need to use the same parser (or better said the same message format).

```
public void SetParser(Parser newParser)
```

Parameters

`newParser` Parser

Class SocketIOEvent

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public class SocketIOEvent
```

Inheritance

object ← SocketIOEvent

Derived

SocketIOPacket

Fields

Namespace

Contians a reference to the SocketIONamespace this packet was received on. It is null on Packets generated locally for sending.

```
public SocketIONamespace Namespace
```

Field Value

SocketIONamespace

eventName

The event name, this event was received under

```
public string eventName
```

Field Value

string

Properties

Length

Returns the number of payloads in this packet. Only valid for messages.

```
public int Length { get; }
```

Property Value

int

acknowledgementID

```
public int acknowledgementID { get; }
```

Property Value

int

callback

If this is an INCOMING acknowledgement, this action triggers sending the acknowledgement. Invoke it using callback.Invoke(payload) where the payload follows the same rules as for an emit. If this Packet is not an incoming acknowledgement, the callback is null.

```
public Action<object[]> callback { get; }
```

Property Value

Action<object[]>

namespacePath

```
public string namespacePath { get; }
```

Property Value

string

payloads

The payloads of this event. Every payload is eighter a byte[] (for binary payloads) or a JToken - which can be a JValue, JObject or JArray IT is recommended to access the paloadsy using GetPayload(...)

```
public List<object> payloads { get; }
```

Property Value

List<object>

type

```
public DataTypes.PacketType type { get; }
```

Property Value

DataTypes.PacketType

Methods

GetPayload<T>(int, bool)

Returns the payload at a specific position. The payload is checked and only returned, when it exists and the type is valid (castable). You can decide the behaviour, if the actual payload does not match the type. This method should work for most Object and Array types as well as binary (byte[]). It might not work for some enumerables. You can always directly access the payloads field.

```
public T GetPayload<T>(int position, bool throwOnError = true)
```

Parameters

`position` int

The position of the payload (zero-based)

`throwOnError` bool

If true or unset, an exception will be thrown if the payload does not exist or does not match the type. If false, the method returns the type's default and a warning is logged instead.

Returns

T

The payload casted into the requested type

Type Parameters

`T`

The type of the payload

Exceptions

IndexOutOfRangeException

Throws IndexOutOfRangeException if throwOnError is true and the event does not contain a payload at the specified position

InvalidCastException

Throws InvalidCastException if throwOnError is true and the requested payload is of an incompatible type

JsonException

Throws InvalidCastException if throwOnError is true and and the requested payload could not be deserialized by Json.Net

Class SocketIONamespace

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This class represents a (usually connected) Socket.IO namespace and implements the EventEmitter and the EventReceiver. We try to keep our API as near to the official Socket.IO V4 API as possible within this class. https://socket.io/docs/v4/emitting-events/https://socket.io/docs/v4/listening-to-events/#eventemitter-methods

```
public class SocketIONamespace
```

Inheritance

object ← SocketIONamespace

Fields

OnSocketIOEventThreaded

This event allows you to receive any Socket IO event (received or internally generated) without additional delay. This callback is executed in a Thread so you may not directly access unity engine functions from it! When timing is not critical, you should use the "On" method instead to register a thread safe, dispatched callback.

```
public DataTypes.ThreadedSocketIOEvent OnSocketIOEventThreaded
```

Field Value

DataTypes.ThreadedSocketIOEvent

Properties

namespacePath

```
public string namespacePath { get; }
```

Property Value

string

socketID

```
public string socketID { get; }
```

Property Value

string

state

```
public DataTypes.ConnectionState state { get; }
```

Property Value

DataTypes.ConnectionState

Methods

BugWorkaroundForReconnect()

This is a temporary workaround for a bug where namespaces are not reconnected correctly. Call this method before a reconnect to override the reconnect logic.

```
public void BugWorkaroundForReconnect()
```

Connect()

A namesapce will automatically connect unless it has manually been disconnected by calling "Disconnect()" on it. A manually disconnected namespace can be reconnected by calling this method.

```
public void Connect()
```

Disconnect()

Calling this method will disconnect the namespace and disable reconnecting to it. To reconnect, you must manually call "Connect()" on it.

```
public void Disconnect()
```

Emit(string, object[], UnityAction<object[]>)

Emits an event to the server

```
public void Emit(string eventName, object[] payloads = null, UnityAction<object[]>
acknowledgementCallback = null)
```

Parameters

`eventName` string

The name of the emitted event

`payloads` object[]

An optional array of payload objects (Any objects supported by Json.Net OR byte[]). Every array element is transmitted as an individual payload. An array of three strings is the equivalent to JS io.emit("someEvent", "string1", "string2", "string3")

`acknowledgementCallback` UnityAction<object[]>

An optional callback. If provided, the emit will be an acknowledgement. This requires a payload.

Emit<T>(string, T, UnityAction<object[]>)

Emits an event to the server that has only one payload (for example a string, byte[] **or a - through Json.Net - serializable object**)

```
public void Emit<T>(string eventName, T payload, UnityAction<object[]>
acknowledgementCallback = null)
```

Parameters

`eventName` string

The name of the emitted event

`payload` T

The payload

`acknowledgementCallback` UnityAction<object[]>

An optional callback. If provided, the emit will be an acknowledgement. This requires a payload. The callback received an object[] where every elemtn is eighter a byte[] or a JToken depending on what the server sent.

Type Parameters

`T`

The type of the primitive payload

Off(string, UnityAction<SocketIOEvent>)

Unregisters a callback for a specific event.

```
public bool Off(string eventName, UnityAction<SocketIOEvent> callback)
```

Parameters

`eventName` string

The name of the event

`callback` UnityAction<SocketIOEvent>

The callback which should be removed

Returns

bool

True if the callback was removed, false otherwise

OffAny()

Unregisters all callbacks for the catch-all event.

```
public void OffAny()
```

OffAny(UnityAction<SocketIOEvent>)

Unregisters a callback for the catch-all event.

```
public bool OffAny(UnityAction<SocketIOEvent> callback)
```

Parameters

`callback` UnityAction<SocketIOEvent>

The callback which should be removed

Returns

bool

True if the callback was removed, false otherwise

On(string, UnityAction)

Registers a callback for a specific event that delivers NO payload. **It will NOT invoke if a payload is contained in the received message!** For any more advanced payload handling, use the "On" method without type assignment. **Warning:** You can not unregister this listener using "Off"! The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it!

```
public void On(string eventName, UnityAction callback)
```

Parameters

`eventName` string

The EventName to subscribe to

`callback` UnityAction

The Callback to invoke on receiption

On(string, UnityAction<SocketIOEvent>)

Registers a callback for a specific event. The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it!

```
public void On(string eventName, UnityAction<SocketIOEvent> callback)
```

Parameters

`eventName` string

The EventName to subscribe to

`callback` UnityAction<SocketIOEvent>

The Callback to invoke on receiption

OnAny(UnityAction<SocketIOEvent>)

Registers a callback for ANY event (catch-all) The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it!

```
public void OnAny(UnityAction<SocketIOEvent> callback)
```

Parameters

`callback` UnityAction<SocketIOEvent>

The Callback to invoke on receiption of any event

OnAny<T>(UnityAction<string, T>)

```
public void OnAny<T>(UnityAction<string, T> callback)
```

Parameters

`callback` UnityAction<string, T>

Type Parameters

`T`

On<T>(string, UnityAction<T>)

This is a wrapper included for convenience in simple projects. It has some limitations. Registers a callback for a specific event which only has ONE payload of a GIVEN TYPE. If the received event has more than one payload, the additional payloads will be ignored. If the first payload is not of the correct type, the callback will not fire. For any more advanced payload handling, use the "On" method without type assignment. **Warning:** You can not unregister this listener using "Off"! The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it!

```
public void On<T>(string eventName, UnityAction<T> callback)
```

Parameters

`eventName` string

The EventName to subscribe to

`callback` UnityAction<T>

The Callback to invoke on receiption

Type Parameters

`T`

The expected type of the first payload (JObject, JArray or a primitive type supported by JValue)

See Also
JValue

Once(string, UnityAction<SocketIOEvent>)

Registers a callback for a specific event which is only called once and then destroyed. The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it! Once-Callbacks are called before registered permanent handlers

```
public void Once(string eventName, UnityAction<SocketIOEvent> callback)
```

Parameters

`eventName` string

The EventName to subscribe to

`callback` UnityAction<SocketIOEvent>

The Callback to invoke ONCE on receiption

Once<T>(string, UnityAction<T>)

This is a wrapper included for convenience in simple projects. It has some limitations. Registers a callback for a specific event which only has ONE payload of a GIVEN TYPE. This callback will only fire the first time, this event is received after registering the ccallback. If the received event has more than one payload, the additional payloads will be ignored. If the first payload is not of the correct type, the callback will not fire. If the event is received and the payloads are not compatible, **the callback is still removed from the list.** For any more advanced payload handling, use the "Once" method without type assignment. The callback is dispatched, so it will always call from the main thread and you can safely access Unity functions from it!

```
public void Once<T>(string eventName, UnityAction<T> callback)
```

Parameters

`eventName` string

The EventName to subscribe to

`callback` UnityAction<T>

The Callback to invoke on receiption

Type Parameters

`T`

The expected type of the first payload (JObject, JArray or a primitive type supported by JValue)

See Also
JValue

RemoveAllListeners()

Unregisters all callbacks (once and permanent) for all events - including catchall.

```
public void RemoveAllListeners()
```

RemoveAllListeners(string)

Unregisters all callbacks (once and permanent) for a specific event.

```
public void RemoveAllListeners(string eventName)
```

Parameters

`eventName` string

The name of the event

Class SocketIOPacket

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This class represents a lowlevel SocketIO packet in its parsed state.

```
public class SocketIOPacket : SocketIOEvent
```

Inheritance

object ← SocketIOEvent ← SocketIOPacket

Inherited Members

SocketIOEvent.type , SocketIOEvent.Namespace , SocketIOEvent.namespacePath , SocketIOEvent.acknowledgementID , SocketIOEvent.callback , SocketIOEvent.payloads , SocketIOEvent.GetPayload<T>(int, bool) , SocketIOEvent.eventName , SocketIOEvent.Length

Namespace Firesplash.GameDevAssets.Socket IOPlus.EngineIO

Classes

DataTypes

EngineIOClient

This component allows creating or accessing a "low level" EngineIO connection. It is created as a subset of our Socket.IO implementation but if required, you can directly access it for example to create your own protocol on top of Engine.IO It does not implement 100% of Engine.IO API but is enough for All-Day usage. The implementation of BINARY Engine.IO messages is untested and provided without warranty. Feel free to report bugs to us though.

EngineIOPacket

Class used to create a packet to be sent via WebSocket to a server using the Engine.IO protocol

Structs

DataTypes.ConnectionParameters

Enums

DataTypes.ConnectionState

DataTypes.EIOPacketType

Delegates

DataTypes.EngineIOConnectErrorEvent

This event fires when the connection throws an error

DataTypes.EngineIOConnectionReadyEvent

This event fires, when the connection is established and ready to be used This event is fired from a thread. You may not access Unity Engine functions directly from the callback.

DataTypes.EngineIODisconnectEvent

This event fires when the connection gets disconnected

DataTypes.EngineIOMessageReceivedEvent

The event raised, when a message is received by the client This event is fired from a thread. You may not access Unity Engine functions directly from the callback.

Class DataTypes

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public static class DataTypes
```

Inheritance

object ← DataTypes

Struct DataTypes.ConnectionParameters

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
[Serializable]
public struct DataTypes.ConnectionParameters
```

Fields

pingInterval

```
public int pingInterval
```

Field Value

int

pingTimeout

```
public int pingTimeout
```

Field Value

int

sid

```
public string sid
```

Field Value

string

Enum DataTypes.ConnectionState

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public enum DataTypes.ConnectionState
```

Fields

`Aborted = 6`

`CloseReceived = 4`

`CloseSent = 3`

`Closed = 5`

`Connecting = 1`

`Handshake = 255`

`None = 0`

`Open = 2`

Enum DataTypes.EIOPacketType

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

```
public enum DataTypes.EIOPacketType
```

Fields

`Close = 1`

`Message = 4`

`Open = 0`

`Ping = 2`

`Pong = 3`

Delegate DataTypes.EngineIOConnectError Event

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This event fires when the connection throws an error

```
public delegate void DataTypes.EngineIOConnectErrorEvent(Exception e)
```

Parameters

e Exception

This event fires when the connection throws an error

Delegate DataTypes.EngineIOConnectionReady Event

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This event fires, when the connection is established and ready to be used This event is fired from a thread. You may not access Unity Engine functions directly from the callback.

```
public delegate void DataTypes.EngineIOConnectionReadyEvent(DataTypes.ConnectionParameters
connectionParams)
```

Parameters

`connectionParams` DataTypes.ConnectionParameters

Delegate DataTypes.EngineIODisconnectEvent

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This event fires when the connection gets disconnected

```
public delegate void DataTypes.EngineIODisconnectEvent(bool serverInitiated, string reason)
```

Parameters

`serverInitiated` bool

true, if the server intentionally disconnected us

`reason` string

A textual reason

Delegate DataTypes.EngineIOMessageReceivedEvent

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

The event raised, when a message is received by the client This event is fired from a thread. You may not access Unity Engine functions directly from the callback.

```
public delegate void DataTypes.EngineIOMessageReceivedEvent(EngineIOPacket packet)
```

Parameters

`packet` EngineIOPacket

The received packet

Class EngineIOClient

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

This component allows creating or accessing a "low level" EngineIO connection. It is created as a subset of our Socket.IO implementation but if required, you can directly access it for example to create your own protocol on top of Engine.IO It does not implement 100% of Engine.IO API but is enough for All-Day usage. The implementation of BINARY Engine.IO messages is untested and provided without warranty. Feel free to report bugs to us though.

```
[DisallowMultipleComponent]
[AddComponentMenu("Networking/Socket.IO/Low-Level Engine.IO Client")]
public class EngineIOClient : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← EngineIOClient

Derived

SocketIOClient

Fields

OnEngineIOConnectionReady

```
[HideInInspector]
public UnityEvent<DataTypes.ConnectionParameters> OnEngineIOConnectionReady
```

Field Value

UnityEvent<DataTypes.ConnectionParameters>

OnEngineIOConnectionReadyThreaded

This native C# callback is invoked immediately when an Engine.IO connection has been established and the handshake is done. Warning: If using Threaded and dispatched events, UnityEvents may be invoked out of order compared to only one kind of events. (You might receive Threaded Event 1, 2, 3 before actually receiving UnityEvent 2 for example) **This callback is invoked from a thread!**

```
public DataTypes.EngineIOConnectionReadyEvent OnEngineIOConnectionReadyThreaded
```

Field Value

DataTypes.EngineIOConnectionReadyEvent

OnEngineIODisconnect

```
[HideInInspector]
public UnityEvent<bool, string> OnEngineIODisconnect
```

Field Value

UnityEvent<bool, string>

OnEngineIOError

```
[HideInInspector]
public UnityEvent<Exception> OnEngineIOError
```

Field Value

UnityEvent<Exception>

OnEngineIOMessageReceived

This UnityEvent is fired on the main thread after an Engine.IO message packet has been received on the websocket. Due to dispatching, it can be slightly delayed.

```
[HideInInspector]
public UnityEvent<EngineIOPacket> OnEngineIOMessageReceived
```

Field Value

UnityEvent<EngineIOPacket>

OnEngineIOMessageReceivedThreaded

This native C# callback is invoked immediately when an Engine.IO message packet is received on the websocket. Warning: If using Threaded and dispatched events, UnityEvents may be invoked out of order compared to only one kind of events. (You might receive Threaded Event 1, 2, 3 before actually receiving UnityEvent 2 for example) **This callback is invoked from a thread!**

```
public DataTypes.EngineIOMessageReceivedEvent OnEngineIOMessageReceivedThreaded
```

Field Value

DataTypes.EngineIOMessageReceivedEvent

serverAddress

```
public string serverAddress
```

Field Value

string

volatileOperationMode

```
[Tooltip("If enabled, the library will drop all packets from the send queue that could not
be transmitted when they were scheduled to be sent. Additionally the send queue is cleared
on (re)connect.\nThis is very similar to the 'Volatile' Socket.IO mode and feels a bit like
UDP - probably the best for realtime data.\n\nWhen disabled, queues are kept across
reconnects and failed transmisions will be re-queued until they succeed, which is similar to
the default Socket.IO mode. This has high reliability but can cause jams.")]
public bool volatileOperationMode
```

Field Value

bool

Properties

SendQueueLength

Returns the length of the current data send queue in transmissions/emits Warning: This is a blocking and quite slow call.

```
public int SendQueueLength { get; }
```

Property Value

int

State

Returns the connection state of the Engine.IO connection

```
public DataTypes.ConnectionState State { get; }
```

Property Value

DataTypes.ConnectionState

Methods

Awake()

```
public void Awake()
```

ClearSendQueue()

This clears the data send queue

```
public void ClearSendQueue()
```

Connect(string)

Connect the client to the server

```
public virtual void Connect(string pServerAddress = null)
```

Parameters

`pServerAddress` string

Connect(string, bool)

Connect the client to the server

```
public virtual void Connect(string pServerAddress, bool volatileMode)
```

Parameters

`pServerAddress` string

`volatileMode` bool

If true, failed transmissions are not retried and queue is cleared on connect. A little like UDP. - If false, failed transmissions are re-queued.

Disconnect()

Disconnect the Engine.IO client

```
public virtual void Disconnect()
```

LateUpdate()

```
protected void LateUpdate()
```

SendEngineIOMessage(byte[])

Sends a binary message to the server using raw Engine.IO protocol

```
public void SendEngineIOMessage(byte[] message)
```

Parameters

`message` byte[]

The message

SendEngineIOMessage(string)

Sends a string message to the server using raw Engine.IO protocol

```
public void SendEngineIOMessage(string message)
```

Parameters

`message` string

The message

SendEngineIOPacket(EngineIOPacket)

Sends a previously built Engine.IO packet without modification

```
public void SendEngineIOPacket(EngineIOPacket packet)
```

Parameters

`packet` EngineIOPacket

The packet

SendEngineIOPackets(EngineIOPacket[])

Sends multiple previously built Engine.IO packets without modification in row

```
public void SendEngineIOPackets(EngineIOPacket[] packets)
```

Parameters

`packets` EngineIOPacket[]

The packet array

Class EngineIOPacket

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: Firesplash.GameDevAssets.SocketIOPlus.dll

Class used to create a packet to be sent via WebSocket to a server using the Engine.IO protocol

```
public class EngineIOPacket
```

Inheritance

object ← EngineIOPacket

Constructors

EngineIOPacket(byte[])

Creates a packet for a binary-typed MESSAGE This can not be used to parse an incoming message!

```
public EngineIOPacket(byte[] messagePayload)
```

Parameters

`messagePayload` byte[]

EngineIOPacket(string)

Creates a packet for a string-typed MESSAGE This can not be used to parse an incoming message!

```
public EngineIOPacket(string messagePayload)
```

Parameters

`messagePayload` string

Methods

GetPacketType()

```
public DataTypes.EIOPacketType GetPacketType()
```

Returns

DataTypes.EIOPacketType

GetPayloadBytes()

```
public byte[] GetPayloadBytes()
```

Returns

byte[]

GetPayloadString()

```
public string GetPayloadString()
```

Returns

string

IsBinaryMessage()

```
public bool IsBinaryMessage()
```

Returns

bool

Parse(bool, byte[])

Used to parse a received byte array from the transport into an Engine.IO packet

```
public static EngineIOPacket Parse(bool isBinaryMessage, byte[] webSocketMessageBytes)
```

Parameters

`isBinaryMessage` bool

Set this true, if the message was received as binary message. Otherwise false.

`webSocketMessageBytes` byte[]

The received byte array

Returns

EngineIOPacket

The parsed package instance