Table of Contents

ExampleScript ExampleScript

Namespace Firesplash.GameDevAssets.SocketIOPlus

Classes

DataTypes

Contains simple DataTypes used for Socket.IO communication and states

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.

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

Contains simple DataTypes used for Socket.IO communication and states

Inheritance

System.Object
DataTypes

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public static class DataTypes
```

Enum DataTypes.ConnectionState

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public enum ConnectionState
```

Fields

Name	Description
CONNECTED	
CONNECTING	
DISCONNECTED	
NONE	

Enum DataTypes.PacketType

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public enum PacketType
```

Fields

Name	Description
ACK	
BINARY_ACK	
BINARY_EVENT	
CONNECT	
CONNECT_ERROR	
DISCONNECT	
EVENT	

Delegate 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

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public delegate object SocketIOAuthPayloadCallback(string namespacePath);
```

Parameters

Type	Name	Description
System.String	namespacePath	The Socket.IO namespace path (e.g. "/") for which authentication data is requested

Returns

Type	Description
System.Object	

Class DataTypes.SocketIOException

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

Inheritance

System.Object
System.Exception
DataTypes.SocketIOException
DataTypes.SocketIOProtocolViolationException

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOException : Exception, ISerializable
```

Constructors

SocketIOException(String)

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

Declaration

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

Parameters

Type	Name	Description
System.String	message	

Class 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!

Inheritance

System.Object
System.Exception
DataTypes.SocketIOException
DataTypes.SocketIOProtocolViolationException

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

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

Constructors

SocketIOProtocolViolationException(String)

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!

Declaration

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

Parameters

Type	Name	Description
System.String	message	

Delegate 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.

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public delegate void ThreadedSocketIOEvent(SocketIOEvent sioEvent);
```

Parameters

Type	Name	Description
SocketIOEvent	sioEvent	

Class DefaultParser

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

Inheritance

System.Object
Parser
DefaultParser

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class DefaultParser : Parser
```

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.

Declaration

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

Parameters

Type	Name	Description
EngineIOPacket	eioPacket	
SocketIOClient	client	

Returns

Type	Description
SocketIOPacket	

Overrides

Parser.Parse(EngineIOPacket, SocketIOClient)

Class Parser

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

Inheritance

System.Object
Parser
DefaultParser

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class Parser
```

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.

Declaration

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

Parameters

Type	Name	Description
EngineIOPacket	eioPacket	The Engine.IO packet to parse
SocketIOClient	client	A reference to the Scoket.IO client

Returns

Type	Description
SocketIOPacket	A Socket.IO packet instance

Class 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)

Inheritance

System.Object
SocketIOClient

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOClient : EngineIOClient
```

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.

Declaration

```
public int maxConnectAttempts
```

Field Value

Type	Description
System.Int32	

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.

Declaration

```
public bool raisingReconnectDelay
```

Field Value

Type	Description
System.Boolean	

Properties

D

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

Declaration

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

Property Value

Type	Description
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)

Declaration

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

Property Value

Type	Description
SocketIONamespace	

Methods

Connect(String)

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)

Declaration

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

Parameters

Type	Name	Description
System.String	pServerAddress	

Disconnect()

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)

Declaration

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

GetNamespace(String, Boolean)

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.

Declaration

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

Parameters

Type	Name	Description
System.String	namespacePath	
System.Boolean	connectIfNotExists	Connect to this namespace if it is not connected (returns null if false and not existing)

Returns

Type	Description
SocketIONamespace	

GetParser()

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)

Declaration

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

Returns

Type	Description
Parser	

LateUpdate()

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)

Declaration

```
protected void LateUpdate()
```

Off(String, UnityAction<Object>)

Unregisters a previously registered manager event callback

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The event name - For valid values see On(...)
UnityAction<System.Object>	callback	The callback to unregister

On(String, UnityAction<Object>)

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

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The event name (one of error, reconnect, reconnect_attempt, reconnect_error, reconnect_failed)
UnityAction<System.Object>	callback	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.

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.

Declaration

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

Parameters

Type	Name	Description
SocketIOAuthPayloadCallback	callback	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).

Declaration

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

Parameters

Type	Name	Description
Parser	newParser	

Class SocketIOEvent

Inheritance

System.Object
SocketIOEvent
SocketIOPacket

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOEvent
```

Fields

eventName

The event name, this event was received under

Declaration

```
public string eventName
```

Field Value

Type	Description
System.String	

Namespace

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

Declaration

```
public SocketIONamespace Namespace
```

Field Value

Type	Description
SocketIONamespace	

Properties

acknowledgementID

Declaration

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

Property Value

Type	Description
System.Int32	

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.

Declaration

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

Property Value

Type	Description
System.Action<System.Object[]>	

Length

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

Declaration

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

Property Value

Type	Description
System.Int32	

namespacePath

Declaration

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

Property Value

Type	Description
System.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(...)

Declaration

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

Property Value

Type	Description
List<System.Object>	

type

Declaration

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

Property Value

Type	Description
PacketType	

Methods

GetPayload<T>(Int32, Boolean)

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.

Declaration

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

Parameters

Type	Name	Description
System.Int32	position	The position of the payload (zero-based)
System.Boolean	throwOnError	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

Type	Description
T	The payload casted into the requested type

Type Parameters

Name	Description
T	The type of the payload

Exceptions

Type	Condition
System.IndexOutOfRangeException	Throws IndexOutOfRangeException if throwOnError is true and the event does not contain a payload at the specified position
System.InvalidCastException	Throws InvalidCastException if throwOnError is true and the requested payload is of an incompatible type

Class 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.

Inheritance

System.Object
SocketIONamespace

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class 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.

Declaration

```
public ThreadedSocketIOEvent OnSocketIOEventThreaded
```

Field Value

Type	Description
ThreadedSocketIOEvent	

Properties

namespacePath

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.

Declaration

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

Property Value

Type	Description
System.String	

socketID

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.

Declaration

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

Property Value

Type	Description
System.String	

state

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.

Declaration

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

Property Value

Type	Description
ConnectionState	

Methods

Disconnect()

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.

Declaration

```
public void Disconnect()
```

Emit(String, Object[], UnityAction<Object[]>)

Emits an event to the server

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The name of the emitted event
System.Object[]	payloads	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")
UnityAction<System.Object[]>	acknowledgementCallback	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**)

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The name of the emitted event
T	payload	The payload

Type	Name	Description
UnityAction<System.Object[]>	acknowledgementCallback	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

Name	Description
T	The type of the primitive payload

Off(String, UnityAction<SocketIOEvent>)

Unregisters a callback for a specific event.

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The name of the event
UnityAction<SocketIOEvent>	callback	The callback which should be removed

Returns

Type	Description
System.Boolean	True if the callback was removed, false otherwise

OffAny()

Unregisters all callbacks for the catch-all event.

Declaration

```
public void OffAny()
```

OffAny(UnityAction<SocketIOEvent>)

Unregisters a callback for the catch-all event.

Declaration

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

Parameters

Type	Name	Description
UnityAction<SocketIOEvent>	callback	The callback which should be removed

Returns

Type	Description

Type	Description
System.Boolean	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!

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The EventName to subscribe to
UnityAction	callback	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!

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The EventName to subscribe to
UnityAction<SocketIOEvent>	callback	The Callback to invoke on receiption

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!

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The EventName to subscribe to
UnityAction<T>	callback	The Callback to invoke on receiption

Type Parameters

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

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!

Declaration

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

Parameters

Type	Name	Description
UnityAction<SocketIOEvent>	callback	The Callback to invoke on receiption of any event

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

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.

Declaration

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

Parameters

Type	Name	Description
UnityAction<System.String, T>	callback	

Type Parameters

Name	Description
T	

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

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The EventName to subscribe to
UnityAction<SocketIOEvent>	callback	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!

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The EventName to subscribe to
UnityAction<T>	callback	The Callback to invoke on receiption

Type Parameters

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

RemoveAllListeners()

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

Declaration

```
public void RemoveAllListeners()
```

RemoveAllListeners(String)

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

Declaration

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

Parameters

Type	Name	Description
System.String	eventName	The name of the event

Class SocketIOPacket

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

Inheritance

System.Object
SocketIOEvent
SocketIOPacket

Inherited Members

SocketIOEvent.type
SocketIOEvent.Namespace
SocketIOEvent.namespacePath
SocketIOEvent.acknowledgementID
SocketIOEvent.callback
SocketIOEvent.payloads
SocketIOEvent.GetPayload<T>(Int32, Boolean)
SocketIOEvent.eventName
SocketIOEvent.Length

Namespace: Firesplash.GameDevAssets.SocketIOPlus

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOPacket : SocketIOEvent
```

Namespace Firesplash.GameDevAssets.SocketIOPlus.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

Inheritance

System.Object
DataTypes

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public static class DataTypes
```

Struct DataTypes.ConnectionParameters

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
[Serializable]
public struct ConnectionParameters
```

Fields

pingInterval

Declaration

```
public int pingInterval
```

Field Value

Type	Description
System.Int32	

pingTimeout

Declaration

```
public int pingTimeout
```

Field Value

Type	Description
System.Int32	

sid

Declaration

```
public string sid
```

Field Value

Type	Description
System.String	

Enum DataTypes.ConnectionState

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public enum ConnectionState
```

Fields

Name	Description
Aborted	
Closed	
CloseReceived	
CloseSent	
Connecting	
Handshake	
None	
Open	

Enum DataTypes.EIOPacketType

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public enum EIOPacketType
```

Fields

Name	Description
Close	
Message	
Open	
Ping	
Pong	

Delegate DataTypes.EngineIOConnectErrorEvent

This event fires when the connection throws an error

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public delegate void EngineIOConnectErrorEvent(Exception e);
```

Parameters

Type	Name	Description
System.Exception	e	

Delegate 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.

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

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

Parameters

Type	Name	Description
DataTypes.ConnectionParameters	connectionParams	

Delegate DataTypes.EngineIODisconnectEvent

This event fires when the connection gets disconnected

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

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

Parameters

Type	Name	Description
System.Boolean	serverInitiated	true, if the server intentionally disconnected us
System.String	reason	A textual reason

Delegate 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.

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public delegate void EngineIOMessageReceivedEvent(EngineIOPacket packet);
```

Parameters

Type	Name	Description
EngineIOPacket	packet	The received packet

Class 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.

Inheritance

System.Object
EngineIOClient

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public class EngineIOClient : MonoBehaviour
```

Fields

OnEngineIOConnectionReady

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.

Declaration

```
public UnityEvent<ConnectionParameters> OnEngineIOConnectionReady
```

Field Value

Type	Description
UnityEvent<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!**

Declaration

```
public EngineIOConnectionReadyEvent OnEngineIOConnectionReadyThreaded
```

Field Value

Type	Description
EngineIOConnectionReadyEvent	

OnEngineIODisconnect

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.

Declaration

```
public UnityEvent<bool, string> OnEngineIODisconnect
```

Field Value

Type	Description
UnityEvent<System.Boolean, System.String>	

OnEngineIOError

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.

Declaration

```
public UnityEvent<Exception> OnEngineIOError
```

Field Value

Type	Description
UnityEvent<System.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.

Declaration

```
public UnityEvent<EngineIOPacket> OnEngineIOMessageReceived
```

Field Value

Type	Description
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!**

Declaration

```
public EngineIOMessageReceivedEvent OnEngineIOMessageReceivedThreaded
```

Field Value

Type	Description
EngineIOMessageReceivedEvent	

serverAddress

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.

Declaration

```
public string serverAddress
```

Field Value

Type	Description
System.String	

Properties

State

Returns the connection state of the Engine.IO connection

Declaration

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

Property Value

Type	Description
ConnectionState	

Methods

Awake()

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.

Declaration

```
public void Awake()
```

Connect(String)

Connect the client to the server

Declaration

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

Parameters

Type	Name	Description
System.String	pServerAddress	

Disconnect()

Disconnect the Engine.IO client

Declaration

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

LateUpdate()

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.

Declaration

```
protected void LateUpdate()
```

SendEngineIOMessage(Byte[])

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

Declaration

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

Parameters

Type	Name	Description
System.Byte[]	message	The message

SendEngineIOMessage(String)

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

Declaration

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

Parameters

Type	Name	Description
System.String	message	The message

SendEngineIOPacket(EngineIOPacket)

Sends a previously built Engine.IO packet without modification

Declaration

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

Parameters

Type	Name	Description
EngineIOPacket	packet	The packet

SendEngineIOPackets(EngineIOPacket[])

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

Declaration

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

Parameters

Type	Name	Description
EngineIOPacket[]	packets	The packet array

Class EngineIOPacket

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

Inheritance

System.Object
EngineIOPacket

Namespace: Firesplash.GameDevAssets.SocketIOPlus.EngineIO

Assembly: cs.temp.dll.dll

Syntax

```
public class EngineIOPacket
```

Constructors

EngineIOPacket(Byte[])

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

Declaration

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

Parameters

Type	Name	Description
System.Byte[]	messagePayload	

EngineIOPacket(String)

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

Declaration

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

Parameters

Type	Name	Description
System.String	messagePayload	

Methods

GetPacketType()

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

Declaration

```
public EIOPacketType GetPacketType()
```

Returns

Type	Description
EIOPacketType	

GetPayloadBytes()

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

Declaration

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

Returns

Type	Description
System.Byte[]	

GetPayloadString()

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

Declaration

```
public string GetPayloadString()
```

Returns

Type	Description
System.String	

IsBinaryMessage()

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

Declaration

```
public bool IsBinaryMessage()
```

Returns

Type	Description
System.Boolean	

Parse(Boolean, Byte[])

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

Declaration

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

Parameters

Type	Name	Description
System.Boolean	isBinaryMessage	Set this true, if the message was received as binary message. Otherwise false.
System.Byte[]	webSocketMessageBytes	The received byte array

Returns

Type	Description
EngineIOPacket	The parsed package instance

Namespace Global

Classes

ExampleScript

Class ExampleScript

Inheritance

System.Object
ExampleScript

Namespace: Global

Assembly: cs.temp.dll.dll

Syntax

```
public class ExampleScript : MonoBehaviour
```

Fields

io

Declaration

```
public SocketIOClient io
```

Field Value

Type	Description
SocketIOClient	

uiGreeting

Declaration

```
public Text uiGreeting
```

Field Value

Type	Description
Text	

uiPodName

Declaration

```
public Text uiPodName
```

Field Value

Type	Description
Text	

uiStatus

Declaration

```
public Text uiStatus
```

Field Value

Type	Description
Text	