Table of Contents

Namespace Firesplash.UnityAssets.SocketIO

Classes

SIOAuthPayload

Creates an object to be sent while connecting to the server. This can be used to authenticate against the server.

SocketIOCommunicator

SocketIOInstance

Enums

SocketIOInstance.SIOStatus

DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect

Delegates

SocketIOInstance.SocketIOCatchallEvent

This is the callback type for Socket.IO "Any" events

SocketIOInstance.SocketIOEvent

This is the callback type for Socket.IO events

Class SIOAuthPayload

Creates an object to be sent while connecting to the server. This can be used to authenticate against the server.

Inheritance

System.Object
SIOAuthPayload

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public class SIOAuthPayload
```

Methods

AddElement(String, Boolean)

Adds a boolean typed value to the payload.

Declaration

```
public void AddElement(string key, bool value)
```

Parameters

Type	Name	Description
System.String	key	The name of this object (on the server side this will go socket.handshake.auth.**HERE**
System.Boolean	value	The value of this object

AddElement(String, Double)

Adds a double typed value to the payload.

Declaration

```
public void AddElement(string key, double value)
```

Parameters

Type	Name	Description
System.String	key	The name of this object (on the server side this will go socket.handshake.auth.**HERE**
System.Double	value	The value of this object

AddElement(String, Int32)

Adds an integer typed value to the payload.

Declaration

```
public void AddElement(string key, int value)
```

Parameters

Type	Name	Description
System.String	key	The name of this object (on the server side this will go socket.handshake.auth.**HERE**
System.Int32	value	The value of this object

AddElement(String, Single)

Adds a float typed value to the payload.

Declaration

```
public void AddElement(string key, float value)
```

Parameters

Type	Name	Description
System.String	key	The name of this object (on the server side this will go socket.handshake.auth.**HERE**
System.Single	value	The value of this object

AddElement(String, String)

Adds a string typed value to the payload.

Declaration

```
public void AddElement(string key, string value)
```

Parameters

Type	Name	Description
System.String	key	The name of this object (on the server side this will go socket.handshake.auth.**HERE**
System.String	value	The value of this object

Clear()

Clears out all previously set payload data from this object

Declaration

```
public void Clear()
```

RemoveElement(String)

Creates an object to be sent while connecting to the server. This can be used to authenticate against the server.

Declaration

```
public bool RemoveElement(string key)
```

Parameters

Type	Name	Description
System.String	key	

Returns

Type	Description
System.Boolean	

Class SocketIOCommunicator

Inheritance

System.Object
SocketIOCommunicator

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOCommunicator : MonoBehaviour
```

Fields

autoConnect

If set to true, the behavior will connect to the server within Start() method. If set to false, you will have to call Connect() on the behavior. WARNING: If autoConnect is enabled, you can not change the target server address at runtime.

Declaration

```
public bool autoConnect
```

Field Value

Type	Description
System.Boolean	

autoReconnect

If set to true, the behavior will connect to the server within Start() method. If set to false, you will have to call Connect() on the behavior. WARNING: If autoConnect is enabled, you can not change the target server address at runtime.

Declaration

```
public bool autoReconnect
```

Field Value

Type	Description
System.Boolean	

secureConnection

If set to true, the connection will use wss/https WARNING: If you need to change this at runtime, make sure to do it BEFORE connecting, else the change will have no effect.

Declaration

```
public bool secureConnection
```

Field Value

Type	Description
System.Boolean	

socketIOAddress

The Address of the SocketIO-Server If you specify a path, it has to be the complete absolute path to the service (the default is /socket.io/) WARNING: If you need to change this at runtime, make sure to do it BEFORE connecting, else the change will have no effect.

Declaration

```
public string socketIOAddress
```

Field Value

Type	Description
System.String	

Properties

Instance

Use this field to access the Socket.IO interfaces

Declaration

```
public SocketIOInstance Instance { get; }
```

Property Value

Type	Description
SocketIOInstance	

Class SocketIOInstance

Inheritance

System.Object
SocketIOInstance

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketIOInstance
```

Properties

SocketID

Contains the SocketID of the current connection. Is null if never connected, still contains the old SocketID after a connection loss until a (re)connect succeeded.

Declaration

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

Property Value

Type	Description
System.String	

Status

Declaration

```
public SocketIOInstance.SIOStatus Status { get; }
```

Property Value

Type	Description
SocketIOInstance.SIOStatus	

Methods

Close()

Closes the connection to the server

Declaration

```
public virtual void Close()
```

Connect()

Connect this Socket.IO instance using the stored parameters from last connect / component configuration

Declaration

```
public virtual void Connect()
```

Connect(SIOAuthPayload)

Connect this Socket.IO instance using the component's set configuration but with (new) auth data

Declaration

```
public virtual void Connect(SIOAuthPayload authPayload)
```

Parameters

Type	Name	Description
SIOAuthPayload	authPayload	An instance of SIOAuthPayload to be sent upon (re-)connection. Can for example be used to send an authentication token.

Connect(String, Boolean, SIOAuthPayload)

When Auto-Connect is disabled(best practice), this call connects to the server.It can also be used to reconnect to a different(or the same) server at runtime. You can optionally specify a targetAddress.If omitted, the system will connect to the server configured in the inspector (or the last target if Connect has already been called before on the instance). If an address is given, you must also specify the enableReconnect Boolean which sets the automatic reconnect function on(true) or off(false). Note: If specified via Connect parameter, the server address must be given as a valid http:// or https:// scheme URI for native and WebGL implementations. The server still has to work using WebSocket transport. Further, the optional authPayload can be given to transmit data(e.g.a token) to the server at connect time which can(and should) be used for authentication purposes.SIOAuthPayload supports bool, string, int, double and float parameters.

Declaration

```
public virtual void Connect(string targetAddress, bool enableReconnect, SIOAuthPayload authPayload)
```

Parameters

Type	Name	Description
System.String	targetAddress	The server / IO address to connect to. Has to start with http:// or https:// (substitute ws with http or wss with https): http[s]://<Hostname>[:<Port>][/<path>]
System.Boolean	enableReconnect	Shall we reconnect automatically on an unexpected connection loss?
SIOAuthPayload	authPayload	Null or an instance of SIOAuthPayload to be sent upon connection. Can for example be used to send an authentication token.

Connect(String, Boolean)

Declaration

```
public virtual void Connect(string targetAddress, bool enableReconnect)
```

Parameters

Type	Name	Description
System.String	targetAddress	
System.Boolean	enableReconnect	

Emit(String, String, Boolean)

Used to send an event to the server containaing am optional payload. If DataIsPlainText is set true, the data will be delivered as a string. Else it will be delivered as a JSON object. If JSON object is sent(DataIsPlainText= false) and the string is not a valid stringified object, unexpected errors might occur.The third parameter is a hard override.

Declaration

```
public virtual void Emit(string EventName, string Data, bool DataIsPlainText)
```

Parameters

Type	Name	Description
System.String	EventName	The name of the event
System.String	Data	The payload (can for example be a serialized object)
System.Boolean	DataIsPlainText	Use this parameter to explicitely state if the data is stringified JSON or a plain text string. Default: false = JSON object

Emit(String, String)

Emits a Socket.IO Event with payload Without third parameter: If the payload is a valid JSON stringified object, the server will receive it as a JSON object. The automatic detection(JSON or PlainText) only works reliably in conjunction with JSON.NET as described above.If you don't use JSON.NET (or if you forgot to set the flag), omitting the third parameter will cause a deprecation warning. If you are using JSON.NET, everything is fine. If not, consider using it (and set the HAS_JSON_NET flag) OR use the third parameter to specify the data type manually.

Declaration

```
[Obsolete("You are sending payload along an Emit without specifying the third parameter. -- This might cause unexpected results for complex objects or some plain text strings. Pl
public virtual void Emit(string EventName, string Data)
```

Parameters

Type	Name	Description
System.String	EventName	The name of the event
System.String	Data	The payload (can for example be a serialized object)

Emit(String)

Emits a Socket.IO Event without payload

Declaration

```
public virtual void Emit(string EventName)
```

Parameters

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

Finalize()

Declaration

```
protected void Finalize()
```

IsConnected()

Returns a Boolean which is true if the library is currently connected to the server.

Declaration

```
public virtual bool IsConnected()
```

Returns

Type	Description
System.Boolean	

Off(String, SocketIOInstance.SocketIOEvent)

Unregisters a specific callback to a given event

Declaration

```
public virtual void Off(string EventName, SocketIOInstance.SocketIOEvent Callback)
```

Parameters

Type	Name	Description
System.String	EventName	
SocketIOInstance.SocketIOEvent	Callback	

Off(String)

Unregisters all callbacks to a given event

Declaration

```
public virtual void Off(string EventName)
```

Parameters

Type	Name	Description
System.String	EventName	

OffAny()

Unregisters all CatchAll-Callbacks

Declaration

```
public virtual void OffAny()
```

OffAny(SocketIOInstance.SocketIOCatchallEvent)

Unregisters a specific Catchall-Callback

Declaration

```
public virtual void OffAny(SocketIOInstance.SocketIOCatchallEvent Callback)
```

Parameters

Type	Name	Description
SocketIOInstance.SocketIOCatchallEvent	Callback	

On(String, SocketIOInstance.SocketIOEvent)

Used to subscribe to a specific event. The callback will be executed everytime when the specific event is received. The callback contains a string. This is the data sent from the server, eighter a stringified JSON object (if the data was a json object) or a plain text string. If the server sent no payload, the string will be null.

Declaration

```
public virtual void On(string EventName, SocketIOInstance.SocketIOEvent Callback)
```

Parameters

Type	Name	Description
System.String	EventName	
SocketIOInstance.SocketIOEvent	Callback	

OnAny(SocketIOInstance.SocketIOCatchallEvent)

Registers a callback that will be called on any incoming event

Declaration

```
public virtual void OnAny(SocketIOInstance.SocketIOCatchallEvent Callback)
```

Parameters

Type	Name	Description
SocketIOInstance.SocketIOCatchallEvent	Callback	

Enum SocketIOInstance.SIOStatus

DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public enum SIOStatus
```

Fields

Name	Description
CONNECTED	DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect
DISCONNECTED	DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect
ERROR	DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect
RECONNECTING	DISCONNECTED means a disconnect happened upon request or a connection has never been attempted. CONNECTED is obvious ERROR means that connection should be established but it is not (check log output) RECONNECTING means that connection was established but got disconnected and the system is still trying to reconnect

Delegate SocketIOInstance.SocketIOCatchallEvent

This is the callback type for Socket.IO "Any" events

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public delegate void SocketIOCatchallEvent(string eventName, string data);
```

Parameters

Type	Name	Description
System.String	eventName	The name of the received event
System.String	data	The data payload of the transmitted event. Plain text or stringified JSON object.

Delegate SocketIOInstance.SocketIOEvent

This is the callback type for Socket.IO events

Namespace: Firesplash.UnityAssets.SocketIO

Assembly: cs.temp.dll.dll

Syntax

```
public delegate void SocketIOEvent(string data);
```

Parameters

Type	Name	Description
System.String	data	The data payload of the transmitted event. Plain text or stringified JSON object.

Namespace Firesplash.UnityAssets.SocketIO.MIT

Classes

Decoder

Encoder

Parser

SocketOpenData

Class Decoder

Inheritance

System.Object
Decoder

Namespace: Firesplash.UnityAssets.SocketIO.MIT

Assembly: cs.temp.dll.dll

Syntax

```
public class Decoder
```

Methods

Decode(String)

Declaration

```
public static SocketPacket Decode(string data)
```

Parameters

Type	Name	Description
System.String	data	

Returns

Type	Description
SocketPacket	

Class Encoder

Inheritance

System.Object
Encoder

Namespace: Firesplash.UnityAssets.SocketIO.MIT

Assembly: cs.temp.dll.dll

Syntax

```
public class Encoder
```

Methods

Encode(SocketPacket)

Declaration

```
public static string Encode(SocketPacket packet)
```

Parameters

Type	Name	Description
SocketPacket	packet	

Returns

Type	Description
System.String	

Class Parser

Inheritance

System.Object
Parser

Namespace: Firesplash.UnityAssets.SocketIO.MIT

Assembly: cs.temp.dll.dll

Syntax

```
public class Parser
```

Methods

ParseData(String)

Declaration

```
public string ParseData(string json)
```

Parameters

Type	Name	Description
System.String	json	

Returns

Type	Description
System.String	

Class SocketOpenData

Inheritance

System.Object
SocketOpenData

Namespace: Firesplash.UnityAssets.SocketIO.MIT

Assembly: cs.temp.dll.dll

Syntax

```
[Serializable]
public class SocketOpenData
```

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	

upgrades

Declaration

```
public string[] upgrades
```

Field Value

Type	Description
System.String[]	

Namespace Firesplash.UnityAssets.SocketIO.MIT.Packet

Classes

SocketPacket

Enums

EnginePacketType

SocketPacketType

Enum EnginePacketType

Namespace: Firesplash.UnityAssets.SocketIO.MIT.Packet

Assembly: cs.temp.dll.dll

Syntax

```
public enum EnginePacketType
```

Fields

Name	Description
CLOSE	
MESSAGE	
NOOP	
OPEN	
PING	
PONG	
UNKNOWN	
UPGRADE	

Class SocketPacket

Inheritance

System.Object
SocketPacket

Namespace: Firesplash.UnityAssets.SocketIO.MIT.Packet

Assembly: cs.temp.dll.dll

Syntax

```
public class SocketPacket
```

Constructors

SocketPacket()

Declaration

```
public SocketPacket()
```

SocketPacket(EnginePacketType, SocketPacketType, Int32, String, Int32, String)

Declaration

```
public SocketPacket(EnginePacketType enginePacketType, SocketPacketType socketPacketType, int attachments, string nsp, int id, string json)
```

Parameters

Type	Name	Description
EnginePacketType	enginePacketType	
SocketPacketType	socketPacketType	
System.Int32	attachments	
System.String	nsp	
System.Int32	id	
System.String	json	

SocketPacket(EnginePacketType)

Declaration

```
public SocketPacket(EnginePacketType enginePacketType)
```

Parameters

Type	Name	Description
EnginePacketType	enginePacketType	

Fields

attachments

Declaration

```
public int attachments
```

Field Value

Type	Description
System.Int32	

enginePacketType

Declaration

```
public EnginePacketType enginePacketType
```

Field Value

Type	Description
EnginePacketType	

id

Declaration

```
public int id
```

Field Value

Type	Description
System.Int32	

json

Declaration

```
public string json
```

Field Value

Type	Description
System.String	

nsp

Declaration

```
public string nsp
```

Field Value

Type	Description
System.String	

socketPacketType

Declaration

```
public SocketPacketType socketPacketType
```

Field Value

Type	Description
SocketPacketType	

Enum SocketPacketType

Namespace: Firesplash.UnityAssets.SocketIO.MIT.Packet

Assembly: cs.temp.dll.dll

Syntax

```
public enum SocketPacketType
```

Fields

Name	Description
ACK	
BINARY_ACK	
BINARY_EVENT	
CONNECT	
CONTROL	
DISCONNECT	
ERROR	
EVENT	
UNKNOWN	

Namespace Global

Classes

[ExampleScript]

[MultiSceneExampleScript]

[PingPongClientSample]

Class ExampleScript

Inheritance

System.Object
ExampleScript

Namespace: Global

Assembly: cs.temp.dll.dll

Syntax

```
public class ExampleScript : MonoBehaviour
```

Fields

sioCom

Declaration

```
public SocketIOCommunicator sioCom
```

Field Value

Type	Description
SocketIOCommunicator	

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	

Class MultiSceneExampleScript

Inheritance

System.Object
MultiSceneExampleScript

Namespace: Global

Assembly: cs.temp.dll.dll

Syntax

```
public class MultiSceneExampleScript : MonoBehaviour
```

Fields

sioCom

Declaration

```
public SocketIOCommunicator sioCom
```

Field Value

Type	Description
SocketIOCommunicator	

Class PingPongClientSample

Inheritance

System.Object
PingPongClientSample

Namespace: Global

Assembly: cs.temp.dll.dll

Syntax

```
public class PingPongClientSample : MonoBehaviour
```

Fields

txtDC

Declaration

```
public Text txtDC
```

Field Value

Type	Description
Text	

txtLosses

Declaration

```
public Text txtLosses
```

Field Value

Type	Description
Text	

txtPing

Declaration

```
public Text txtPing
```

Field Value

Type	Description
Text	

txtPong

Declaration

```
public Text txtPong
```

Field Value

Type	Description
Text	

txtSID

Declaration

```
public Text txtSID
```

Field Value

Type	Description
Text	