Socket.IO V3 / V4 Client Asset Documentation

Thank you for buying our asset.

Unfortunately, sometimes customers leave negative ratings and complain about things not working without contacting us so we cannot help or fix the issue. We want to provide good support and have our customers having a great time developing their content, so:

If you happen to have any issues, please write an email to assets@firesplash.de and we will be happy to help!

If you send a mail to this address, you will receive an automatic confirmation mail within a few minutes. Please check your spam/junk folder!

This asset is used to connect a unity project to a server (or multiple servers) using Socket.IO middleware. The library provides the most important methods for this. It was not and will never be a try to implement every single aspect but the library is extendable as we provide the full source code.

What we sell and deliver is everything you need for **most** applications.

After all we would love to get your honest review on this asset on the asset store when you got some experience with it. This helps us out a LOT!

Socket.IO V3 / V4 Client Asset Documentation

Contents of this documentation

Technical overview

Socket.IO Protocol version compatibility

For general version compatibility please referrer to this link: https://socket.io/docs/v4/client-installation/index.html

Compatibility matrix from our side:

Server Version -> v Asset Version v	v1.x	v2.x	v3.0.x	v3.1.x	v4.x
Asset V2					
Asset V3/V4					

Green = Full compatibility (within the general limitations of this asset according to this documentation)

Yellow = Compatible via 'allowEIO3' flag and **NOT officially supported** – Might break at any time.

Red = incompatible

Backwards Compatibility Note / Warning

One might ask: If your Socket.IO asset v2 is compatible to Socket.IO Server v4, why was this still a good investment? Why the higher price?

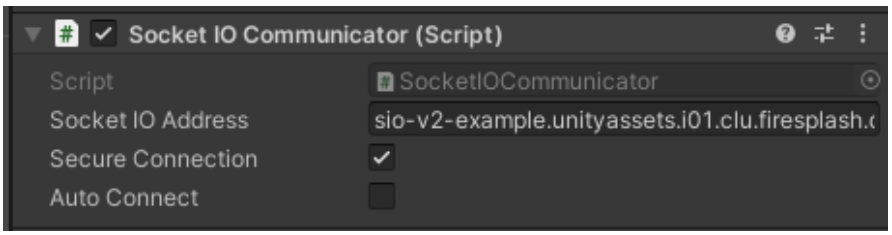
The Socket.IO v2 asset only covers the very basic portions of the protocol in native mode. For example reconnects are not automated, most of the status events are not emitted to your application and after all the compatibility layer is not a future-safe investment. This Asset version implements the most important events (reconnect, connect_error, reconnect_error and so on) as well as auto reconnecting. Further Socket.IO v3/v4 has a way better timeout/ping handling and is considered more stable. This is only the case when both sides are v3/v4

The higher price is applied as v3/v4 support needed huge changes in the code base and lots of research. The undelaying protocol has significantly changed since v2 which is also the reason why v2 and v3/v4 are not compatible (without the compatibility flag which actually degrades v3/v4 servers to v2)

After all we provide our customers a discounted upgrade/downgrade path, so at any time customers will only pay the extra work, and do not have to buy a complete second asset.

Project / Asset Set Up

Our asset works quite straight-forward. Just create a GameObject and add the “Socket IO Communicator” component to it. Enter all detail used for connecting into the fields. The Address has to be provided without protocol prefix or any folders.
Socket.IO-Namespaces are not supported by this library.



This dialog does now look more straight forward and contains more settings.

The Socket.IO-Address has to be entered in an URI-Format without protocol: server.address[:port][/path/][?Query-Parameters]
If using standard ports (Insecure: tcp/80, secure: tcp/443) you do not need to enter a port.
If using the standard socket.io path (/socket.io/) you do not need to specify a path. WARNING: Paths are not the same as namespaces! A path is what you enter in path-option on server side.

Whenever possible you should not use the “Auto Connect” feature as most likely you will want to **setup the listeners** (“On”) **before connecting**. See the example for a best practice setup.

Connecting at runtime / from code

Instead of configuring the connection parameters in the component using the inspector, you can slo specify the target address at runtime using the Connect() method:

```
void Instance.Connect( [ string targetAddress, bool enableReconnect, ] [ SIOAuthPayload authPayload ] )
```

This is described later in this documentation.

Multiple Socket.IO Connections

If you want to connect to more than one Socket.IO server, you have to use on GameObject per connection and you need to name them different. If you use two GameObjects identically named with SocketIOCommunicator Component in WebGL builds, it will result in unexpected behavior because the Socket.IO system depends on unique names from javascript (JSLib) side.
This is a Unity limitation.

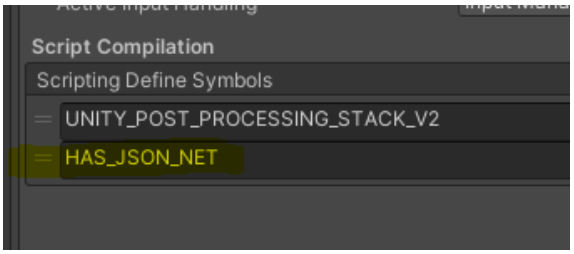
Json.NET for better Json parsing

Why Json.NET

Unfortunately Unity's own JSON "Skills" (JsonUtility) are very limited and everything except reliable when it comes to complex objects – but that's fine as that is not the scope of those methods. There is a very great alternative out there: Newtonsoft Json / JSON.NET – The latter is a fork which has been customized to support Unity's IL2CPP builds (e.g. WebGL).

How do I integrate Json.NET?

To utilize that and keep the ease of use of "Emit(eventName, data)" without the third parameter, install the JSON-NET package as described here: https://github.com/jilleJr/Newtonsoft.Json-for-Unity#installation



Afterwards head to your project's player settings and set the "Scripting Define Symbol" HAS_JSON_NET:

Do not set the flag, if you did not install JSON.NET into the project. This would cause compiler errors.

Recent Unity versions bring their own copy of Json.Net. Their package is identical to the „official" Json.Net implementations so you can use whichever your prefer´but installing Json.Net while having unity's variant in the project will cause compiler errors regarding duplicate assemblies.
Unity's variant is currently delivered with the perforce / collab plugin which is enabled by default.
https://docs.unity3d.com/Packages/com.unity.nuget.newtonsoft-json@2.0/manual/index.html

Using the asset / Code Referene

We provide a well-documented example in our asset. Please review the code to get a closer understanding.

The SocketIOCommunicator Component provides access to a singleton "Instance" which contains the actual Socket.IO implementation specific to the platform. You use this to interact with the library.
Further in runtime it will create an object called "SIODispatcher" which is a helper for running everything thread safe. This component is used to enqueue actions to be run on the main thread.

The following methods and fields are available:

`string Instance.`**`SocketID`** (readonly)

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.

```
void Instance.On(string eventName, Callback(string payloadData));
```

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.

Example:

```
sioCom.Instance.On("WhosThere", (string payload) =>
  {
    Debug.Log("Data received: " + payload);
  }
);
```

```
void Instance.Off(string eventName, Callback);
```

Used to remove the subscription to an event.

Example:

```
sioCom.Instance.Off("WhosThere", WhosthereHandlerMethod);
```

```
void Instance.Emit(string eventName [[, string payloadData], bool DataIsPlainText]);
```

Used to send an event to the server containaing am optional payload.
With third parameter: 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.
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.

Examples:

```
sioCom.Instance.Emit("ItsMe", "Hello World", true);

sioCom.Instance.Emit("ItsMe", "{\"msg\": \"Hello World\"}", false);


// The next example is not recommended without Json.Net.
// If possible, always supply the third parameter!

sioCom.Instance.Emit("ItsMe", "Hello World");
```

`void Instance.Connect( [ string targetAddress, bool enableReconnect, ] [ SIOAuthPayload authPayload ] )`

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.

There are several overloads of this method: With and without address as well as with and without payload.

Examples:

```
sioCom.Instance.Connect();

sioCom.Instance.Connect("https://example.com:15000/socket.io/", true);

SIOAuthPayload auth = new SIOAuthPayload();
auth.AddElement("id", 1234);   //On Server: socket.handshake.auth.id
auth.AddElement("token", "abc123zyx"); //On Server: socket.handshake.auth.token
sioCom.Instance.Connect(auth);
```

`void Instance.Close()`

Closes the connection to the server

Example:

```
sioCom.Instance.Close();
```

`bool Instance.IsConnected()`

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

Example:

```
while (sioCom.Instance.IsConnected()) {
  sioCom.Instance.Emit("KnockKnock");
  //Please don't do this for real. That's evil 😄
}
```

Common Issues and Frequently Asked Questions

The game is not connecting to my server…

…not even locally (but the example works with the Firesplash Entertainment hosted demo server)

In most cases this is caused by one of the following reasons:

1. **You enabled the "Secure Connection" option in the component but did not correctly configure SSL on your server side – or vice versa!**
   Try to access the Socket.IO server using a Webbrowser: http(s)://your.server-address.here/socket.io/
   Don't forget the trailing "/" It **should** say `{"code":0,"message":"Transport unknown"}`
   If any browser errors appear, this is your issue. Be sure to use the exact same address as in the unity component at the red position, and set **http** for *unchecked* secure connect and **https** for *checked*.
2. **You are using the wrong port**
   Socket.IO assumes port 80 (insecure) or 443 (secure). Make sure to enter the correct port if you are using some other. Test access like in solution1 to make sure everything is right

Now I was able to run a server locally and connect the game with it but… after uploading to the server it is no longer working again…

Well…

1. **Again: SSL might not be configured correctly (you won't communicate unencrypted in production, right?)**
   Check SSL as done in the last question, possible solution 1. But this time using the actual server address.
2. **Your server does not accept the transport "websocket" or some layer in between drops the requests**
   This is a whip of rocket science for most developers: Socket.IO for Unity depends on the "websocket" transport and is NOT able to fallback to long-polling. This is why a standard socket.io application might work while the unity asset is not working.
   Load Balancers, Reverse Proxies, Firewalls and any device/solution between the client and the server could potentially cause issues here. Usually the issue sits on the server side. Most likely – if using one – your load balancer or reverse proxy (like nginx/apache proxying requests through your servers port 80/443 down to a nodejs based server on a different port) is not configured correctly.
   Here are some hints for reverse proxy configuration: https://socket.io/docs/v3/reverse-proxy/
   This Rocket-Science can easily be tested by writing a web based test application. Have a look at our example ServerCode zip archive. It contains such a test html file. The important thing about it is to only try the websocket transport using the options: https://socket.io/docs/v3/client-initialization/#transports

I am using a transport adapter on my server side…

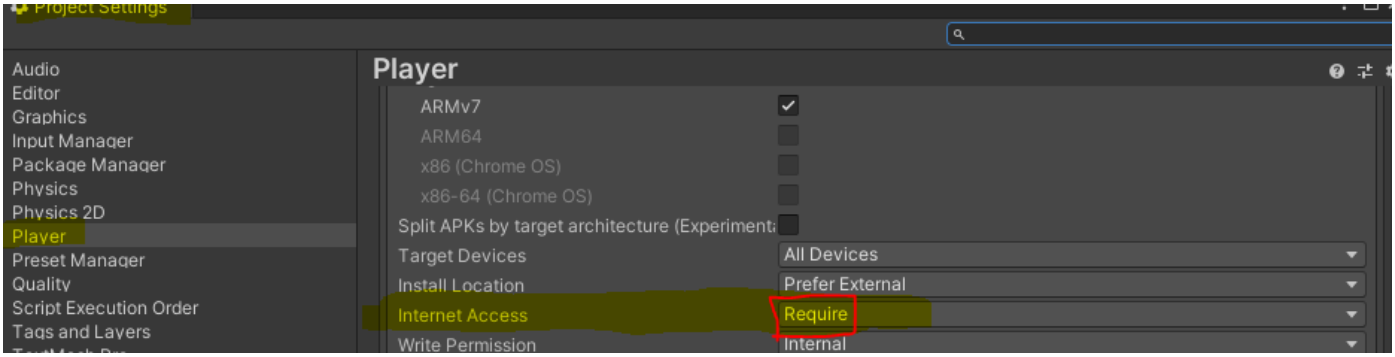
Can I connect to this server using your asset?

Usually no. Transport adapters need to be implemented on both sides. You would have to rework our code to implement the transport adapter.

My Android build does not connect to the server…

It says “WebSocketException”…

Unity does not cleanly detect when an app needs networking if you are not using unity’s own Web-APIs.
You have to manually set internet access to required in project settings:



How do I Serialize/Deserialize complex objects?

Unity’s builtin “JsonUtility” does not support complex objects. This means it only supports objects and only a single layer of complexity.

Job	Builtin JsonUtility variant / workaround	Better variant (requires Json.Net)
Employee data	{ “firstname”: “John”, “surname”: “Baker”, “job”: “Dentist” }	{ name: { “first”: “John”, “last”: “Baker” }, “job”: “Dentist” }
A list of numbers	{ “a”: 12, “b”: 15, “c”: 20 }	[12, 15, 20]
A dynamic list of employees	*Not possible at all*	[{ name: { “first”: “John”, “last”: “Baker” }, “job”: “Dentist” }, { name: { “first”: “Martina”, “last”: “Carrey” }, “job”: “Student” }, …]

The good news: All those “bad” situations can be solved using Json.Net (we talked about this on page two)

I am using WebGL (on an SSL-secured site)…

…and the game does not connect to the server.

The most common issue is an SSL-misconfiguration. Please make sure that you can access **https://*<your-socketio-address>*/socket.io/** from your browser without any SSL errors or warnings. (A json formatted transport error is fine) Also remember that in modern browsers, a WebGL game delivered via HTTPS may not contact a server using HTTP/WS but only using HTTPS/WSS so this means SSL everywhere or nowhere.

My IL2CPP-Build (WebGL is always IL2CPP) using Json.Net is showing errors…

…like Attempting to call method 'System.Collections.Generic.List[…]::.ctor' for which no AOT code was generated

For some reason using arrays in structures does not work on IL2CPP. Try to use Lists instead.

`struct Foo {public int[] bar;}` could break, `struct Foo {public List<int> bar;}` works.

My WebGL (or other IL2CPP) build using Json.Net is crashing…

…telling me that a constructor or type has not been found

Most likely the linker stripped code that is used by Json.Net. You should add a link.xml file to your project which restricts the linker from doing so. Its content depends on your project but here is an example:
https://github.com/SaladLab/Json.Net.Unity3D/blob/master/src/UnityPackage/Assets/link.xml

I imported Json.Net and now I am getting a lot of "duplicate assembly" errors

Sometime around mid of 2021 Unity started to deliver their "own" Json.Net package with some components. If your project already contains that package, installing Json.Net will cause this behavior. It is up to you to decide what to do. Unity officially states that their package is not meant to be used but on the other hand you got no real choice. Seen from a dev's perspective it should not cause any issues so if your project includes unity's version, remove your imported one and use unity's.
https://docs.unity3d.com/Packages/com.unity.nuget.newtonsoft-json@2.0/manual/index.html

I copied a snippet form this documentation but I am receiving a syntax error…

or the server does not correctly handle the parsed JSON

Sorry. This could be our fault. Sometimes when exporting this documentation to PDF, the " sign gets replaced by better looking unicode characters „ and “. Please have an eye on this and always use the ASCII " sign.