

Introduction

WebSockets represent a long-awaited evolution in client/server web technology. They allow a long-held single TCP socket connection to be established between the client and server which allows for bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connection.

sgcWebSockets is a complete package providing access to WebSockets protocol, allowing to create WebSockets Clients and Server and implement a lot of APIs based on this protocol

- Fully functional multithreaded WebSocket server according to RFC 6455.
- Supports Windows 32/64.
- Supports MacOSX 32/64.
- Supports LINUX.
- Supports Message Compression using PerMessage_Deflate extension RFC 7692.
- Supports Text and Binary Messages.
- WatchDog and HeartBeat built-in support.
- Built-in sub-protocols: WebRTC, MQTT (3.1.1 and 5.0) and more.
- Client Built-in API: SignalR Core, Kraken, STOMP, SocketIO and more.
- Built-in Javascript libraries to support browser clients.
- Easy to setup
- Javascript Events for full control
- SSL/TLS Support for Server / Client Components.

Find below a list of current console applications supported:

Application	Description
sgcWebSocketServer	websocket server which can handle multiple threaded client connections.
sgcWebSocketClient	websocket client which can connect to websocket servers.
sgcSocketIOClient	socket.IO client which is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers.
sgcSignalRCoreClient	SignalRCore client for ASP.NET Core SignalR which is an open-source library that simplifies adding real-time web functionality to apps.
sgcKrakenClient	kraken WebSockets Public API offers real-time market data updates.
sgcMQTTClient	is a Client for MQTT publish/subscribe messaging transport protocol.
sgcWebRTCServer	WebRTC (Web Real-Time Communication) is an API definition being drafted by the World Wide Web Consortium (W3C) to enable browser to browser applications for voice calling, video chat and P2P file sharing without plugins
sgcSTOMPClient	STOMP is the Simple (or Streaming) Text Orientated Messaging Protocol.

TCP Server Input/Output

All console applications can run an internal **TCP Server** that allows to send and receive commands as if were written directly on the console application.

Just start the application with the command `-server` and a new server will be started listening on port 8080.
Example: if you want enable the server on WebSocket client, start the console application as follows:

```
sgcWebSocketClient.exe -server
```

Find below the full TCP Server parameters:

```
-server: starts the internal server on port 8080  
-server.port: sets the listening port  
-server.ip: sets the binding ip (useful if there are multiple ips).
```

If the server is enabled, a client connected to this server can send any JSON message and will receive any response as if was written directly from console application.

Example: if you want enable server on IP 192.168.1.1 and Port 5412

```
sgcWebSocketClient.exe -server -server.ip 192.168.1.1 -server.port 5412
```

Console | sgcWebSocketServer

This guide describes the use of the **sgcWebSocketServer** command-line tool.

Features

1. **Fully Featured Application:** Supports Full RFC6455 and widely tested by thousands of users.
2. **Multiple Threads support:** Supports thousands of concurrent users.
3. **Easy to Use:** Simple but very powerful.
4. **Watchdog and HeartBeat:** Automatic recovery after a disconnection and heartbeat support.

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcWebSocketServer help	Shows a list of supported commands with examples
exit	sgcWebSocketServer exit	exits application
start server	{"message":"start", "params":{"host": "127.0.0.1", "port": 80}}	start server listening on port 80 and ip 127.0.0.1
broadcast message	{"message":"broadcast", "params":{"text": "My First Message"}}	broadcast "My First Message" to

		all clients connected
send message	{"message": "write", "params": {"guid": "EDKSKJDFAAJDJ38437", "text": "My First Message"}}	sends a message to a single client
stop server	{"message": "stop"}	stops server
enable ssl	{"message": "ssl", "params": {"ssl": {"enabled": true, "certfile": "sgc.pem", "keyfile": "sgc.pem", "rootcertfile": "sgc.pem", "password": "", "port": 443}}}	enable ssl connections using sgc.pem as certificate and port 443
enable heartbeat	{"message": "heartbeat", "params": {"heartbeat": {"enabled": true, "interval": 30}}}	enable heartbeats every 30 seconds
enable watchdog	{"message": "watchdog", "params": {"watchdog": {"enabled": true, "interval": 10}}}	automatically reconnects after 10 seconds

Events

Event	Format	Comments
connect	{"event": "connect", "params": {"guid": "SDFKSJDFJ33223"}}	New client connects
disconnect	{"event": "disconnect", "code": 0, "params": {"guid": "SDFKSJDFJ33223"}}	Client disconnects
error	{"event": "error", "description": "authentication error", "params": {"guid": "SDFKSJDFJ33223"}}	Error in a connection
message	{"event": "message", "text": "Message from server", "params": {"guid": "SDFKSJDFJ33223"}}	Message received from client
startup	{"event": "startup"}	Server starts
shutdown	{"event": "shutdown"}	Server stops

Console | sgcWebSocketClient

This guide describes the use of the **sgcWebSocketClient** command-line tool.

Features

1. **Fully Featured Application:** Supports Full RFC6455 and widely tested by thousands of users.
2. **TLS Support:** Supports secure websocket connections without external libraries (OSX and Linux requires openssl 1.0.2).
3. **Easy to Use:** Simple but very powerful.
4. **Watchdog and HeartBeat:** Automatic recovery after a disconnection and heartbeat support.

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcWebSocketClient help	Shows a list of supported commands with examples
exit	sgcWebSocketClient exit	exits application
connect to websocket server	{"message":"open", "params": {"url": "ws://echo.websocket.org"}}	connects to echo.websocket.org websocket server
connect to tcp server	{"message":"open", "params": {"url": "tcp://echo.websocket.org"}}	connects to echo.websocket.org websocket server
send message	{"message":"write", "params": {"text": "My First Message"}}	sends a message to server

broadcast message	<code>{"message": "broadcast", "params": {"text": "My First Message"}}</code>	sends a message to the server, and the server broadcast the message to all connected clients.
close connection	<code>{"message": "close"}</code>	close connection
enable heartbeat	<code>{"message": "heartbeat", "params": {"heartbeat": {"enabled": true, "interval": 30}}}</code>	enable heartbeats every 30 seconds
enable watchdog	<code>{"message": "watchdog", "params": {"watchdog": {"enabled": true, "interval": 10}}}</code>	automatically reconnects after 10 seconds

Events

Event	Format	Comments
connect	<code>{"event": "connect"}</code>	client connects
disconnect	<code>{"event": "disconnect", "code": 0}</code>	Client disconnects
error	<code>{"event": "error", "description": "authentication error"}</code>	Error in connection
message	<code>{"event": "message", "text": "Message from server"}</code>	Message received from server

Console | sgcSocketIOClient

This guide describes the use of the **sgcSocketIOClient** command-line tool.

Features

1. **Fully Featured Application:** Socket.IO is a JavaScript library for realtime web applications. It enables realtime, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event-driven.
2. **Supported APIs:** Client component supports APIs: 0.*, 1.* and 2.*

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcSocketIOClient help	Shows a list of supported commands with examples
exit	sgcSocketIOClient exit	exits application
connect to socket.io server	{"message":"open", "params":{"url": "wss://socket-io-chat.now.sh"}}	connects to socket-io-chat.now.sh socket.io server
send a message	{"message":"write", "params":{"text": "42[\"add user\", \"John\"]"}}	sends a message

close connection	{"message": "close"}	close connection
---------------------	----------------------	---------------------

Events

Event	Format	Comments
connect	{"event": "connect"}	client connects
disconnect	{"event": "disconnect", "code": 0}	Client disconnects
error	{"event": "error", "description": "authentication error"}	Error in connection
message	{"event": "message", "text": "Message from server"}	Message received from server

Console | sgcSignalRCoreClient

This guide describes the use of the **sgcSignalRCoreClient** command-line tool.

Features

1. **Fully Featured Application:** ASP.NET Core SignalR is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly.
2. **Good candidates for SignalR:** Apps that require high frequency updates from the server. Examples are gaming, social networks, voting, auction, maps, and GPS apps.
3. **Methods supported:** Client component supports Invocations, Stream Invocations, Completion Result, HeartBeat, Handle Errors, Hubs...
4. **Easy to use:** Simple but very powerful.

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcSignalRCoreClient help	Shows a list of supported commands with examples
exit	sgcSignalRCoreClient exit	exits application
	{"message": "options", "protocol": "signalrcore", "params":{"hub": "/" ChatHub"}}	
connect to SignalRCore server	{"message":"open", "params": {"url": "ws://www.esegece.com/5000"}}	connects to signalrcore server

send message	{"message": "invoke", "protocol": "signalrcore", "params":{"target": "SendMessage", "arguments": ["First", "Second"]}}	sends a message to server
send stream	{"message": "invokestream", "protocol": "signalrcore", "params": {"target": "Counter", "arguments": [10, 50], "id": 2}}	sends stream message
cancel invocation	{"message": "cancelinvocation", "protocol": "signalrcore", "params": {"id": 2}}	cancel invocation with id 2
stream item	{"message": "streamitem", "protocol": "signalrcore", "params": {"item": "Test", "id": 10}}	Stream Item "Test" Invocation "10"
completion result	{"message": "completionresult", "protocol": "signalrcore", "params": {"result": "OK", "id": 20}}	Completion Result "OK" Invocation "20"
completion error	{"message": "completionresult", "protocol": "signalrcore", "params": {"error": "Not Found", "id": 20}}	Completion Error "Not Found" Invocation "20"
close connection	{"message": "close", "protocol": "signalrcore", "params":{"reason": "Closed By Server"}}	Close with error "Closed By Server"
Reconnect	{"message": "reconnect", "protocol": "signalrcore", "params": {"connectionid": 100}}	Reconnect with ConnectionId "100"

Events

Event	Format	Comments
beforeconnect	{"event": "beforeconnect", "protocol": "signalrcore", "params":{"connectionid": "SLDKJF33KJSKDJ"}}	Before client connects
connect	{"event": "connect", "protocol": "signalrcore", "params": {"connectionid": "SLDKJF33KJSKDJ"}}	Client connects
disconnect	{"event": "disconnect", "protocol": "signalrcore", "params": {"closecode": 4000, "closereason": "Connection Closed."}}	Connection closed
error	{"event": "error", "protocol": "signalrcore", "params": {}}	Connection error

	{"description": "Authentication error"}}	
Completion Result	{"event": "completion", "protocol": "signalrcore", "params": {"result": "OK", "id": "2"}}	Completion result successful
Completion Error	{"event": "completion", "protocol": "signalrcore", "params": {"error": "Error", "id": "2"}}	Completion error
Invocation	{"event": "invocation", "protocol": "signalrcore", "params": {"target": "MyMethod", "arguments": "[1,2]", "id": "1"}}	Received invocation from server
Stream Invocation	{"event": "streaminvocation", "protocol": "signalrcore", "params": {"target": "MyStreamMethod", "arguments": "[1, 2]", "id": "2"}}	Received Stream Invocation
Stream Item	{"event": "streamitem", "protocol": "signalrcore", "params": {"id": "ABC1", "item": "1"}}	Received Stream Item
Cancel Invocation	{"event": "cancelinvocation", "protocol": "signalrcore", "params": {"invocationid": "2"}}	Received Cancel Invocation

Console | sgcKrakenClient

This guide describes the use of the **sgcKrakenClient** command-line tool.

Features

1. **Fully Featured Application:** Kraken WebSockets Public API offers real-time market data updates. WebSockets is a bidirectional protocol offering fastest real-time data, helping you build real-time applications.
2. **Supported methods:** Client component supports Ticker, OHLC, Trade, Booke, Spread, Ping and other methods.

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcKrakenClient help	Shows a list of supported commands with examples
exit	sgcKrakenClient exit	exits application
connect to server	{"message": "open"}	connects to kraken server
subscribe ticker	{"message": "subscribeticker", "protocol": "kraken", "pairs": ["XBT/USD"]}	subscribe ticker XBT/USD
unsubscribe ticker	{"message": "unsubscribeticker", "protocol": "kraken", "pairs": ["XBT/USD"]}	unsubscribe ticker XBT/USD

subscribe OHLC	{"message": "subscribeohl", "protocol": "kraken", "pairs":["XBT/USD"], "params":{"interval": 1}}	subscribe OHLC XBT/USD 5min
subscribe Trades	{"message": "subscribetrade", "protocol": "kraken", "pairs":["XBT/USD"]}	subscribe Trades XBT/USD
subscribe book	{"message": "subscribebook", "protocol": "kraken", "pairs":["XBT/USD"], "params":{"depth": 1}}	subscribe Book Depth 25 XBT/USD
subscribe spread	{"message": "subscribespread", "protocol": "kraken", "pairs":["XBT/USD"]}	subscribe spread XBT/USD
subscribe all	{"message": "subscribeall", "protocol": "kraken", "pairs":["XBT/USD"]}	subscribe ALL XBT/USD
close	{"message":"close"}	close connection

Events

Event	Format	Comments
connect	{"event": "connect", "protocol": "kraken", "params": {"connectionid": "12315646515312", "status": "132131254456456", "version": "1.0.0"}}	client connects
data	{"event": "data", "protocol": "kraken", "params":{"data": "[...]"}}	receive data from server
subscribed	{"event": "subscribed", "protocol": "kraken", "params": {"channelid": 123123213, "pair": "XBT/USD", "subscription": "ticker", "channelname": "ticker", "reqid": 411}}	subscribed to channel ticker
subscription error	{"event": "subscriptionerror", "protocol": "kraken", "params": {"errormessage": "Subscription not exists", "pair": "XBT/USD", "subscription": "ticker", "reqid": 322}}	Subscription error
unsubscribed	{"event": "unsubscribed", "protocol": "kraken", "params": {"channelid": 123123213, "pair": "XBT/USD", "subscription": "ticker", "reqid": 411}}	unsubscribed from channel ticker

system status	<pre>{"event": "systemstatus", "protocol": "kraken", "params": {"connectionid": "5451223121", "status": "5451223121", "version": "1.0.0"}}</pre>	update subscription status
------------------	--	----------------------------------

Console | sgcMQTTClient

This guide describes the use of the **sgcMQTTClient** command-line tool.

Features

1. **MQTT publish subscribe architecture:** The MQTT messages are delivered asynchronously (“push”) through publish subscribe architecture.
2. **MQTT 3.1.1 and 5.0:** Component supports MQTT 3.1.1 protocol and latest 5.0 protocol.
3. **Quality of Service (QoS) for MQTT:** Three QoS for message delivery could be achieved using MQTT
4. **Over TCP and WebSockets:** Supports MQTT over WebSockets and over TCP (encrypted and none encrypted).

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcMQTTClient help	Shows a list of supported commands with examples
exit	sgcMQTTClient exit	exits application
connect to tcp mqtt server	{"message": "open", "params": {"url": "tcp://test.mosquitto.org:1883"}}	open a new TCP MQTT connection
connect to ws mqtt server	{"message": "open", "params": {"url": "ws://test.mosquitto.org:8080"}}	open a new WebSocket MQTT connection

subscribe	{"message": "subscribe", "protocol": "mqtt", "params": { "topic": "myTopic" }}	subscribe to topic "myTopic"
publish	{"message": "publish", "protocol": "mqtt", "params": { "topic": "myTopic", "text": "My First Message" }}	publish message "My First Message" in channel "myTopic"
unsubscribe	{"message": "unsubscribe", "protocol": "mqtt", "params": { "topic": "myTopic" }}	unsubscribe from topic "myTopic"
close	{"message": "disconnect", "protocol": "mqtt"}	close connection
mqtt 5.0	{"message": "api", "protocol": "mqtt", "params": { "version": "5.0" }}	enable mqtt 5.0
authentication	{"message": "authentication", "protocol": "mqtt", "params": { "enabled": true, "username": "user", "password": "secret" }}	enable authentication
heartbeat	{"message": "heartbeat", "protocol": "mqtt", "params": { "enabled": true, "interval": 30 }}	enable heartbeat every 30 seconds

Events

Event	Format	Comments
connect	{"event": "connect", "protocol": "mqtt", "params": { "session": "324234222 ", "reasoncode": 0, "reasonname": "connected" }}	client connects
disconnect	{"event": "disconnected", "protocol": "mqtt", "params": { "reasoncode": 51, "reasonname": "Closed by Server." }}	Client disconnects
subscribe	{"event": "subscribe", "protocol": "mqtt", "params": { "packetidentifier": 55411, "reasoncode": 100, "reasonname": "Channel" }}	subscribed to channel
unsubscribe	{"event": "unsubscribe", "protocol": "mqtt", "params": { "packetidentifier": 1234, "reasoncode": 500, "reasonname": "Channel" }}	Unsubscribed from channel

publish	<code>{"event": "publish", "protocol": "mqtt", "params": {"topic": "MyTopic", "text": "Message From Server"}}</code>	client receives a message from server
pubrec	<code>{"event": "pubrec", "protocol": "mqtt", "params": {"packetidentifier": 1255, "reasoncode": 0, "reasonname": ""}}</code>	client receives a pubrec message
pubcomp	<code>{"event": "pubcomp", "protocol": "mqtt", "params": {"packetidentifier": 1255, "reasoncode": 0, "reasonname": ""}}</code>	client receives a pubcomp message
puback	<code>{"event": "puback", "protocol": "mqtt", "params": {"packetidentifier": 1255, "reasoncode": 0, "reasonname": ""}}</code>	client receives a puback message

Console | sgcWebRTCServer

This guide describes the use of the **sgcWebRTCServer** command-line tool.

Features

1. **WebRTC Protocol:** WebRTC (Web Real-Time Communication) is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.
2. **Wide:** Its mission is to enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices.
3. **Applications:** Build WebRTC Applications for: Secure Video Monitoring, Shared live media streaming, Video conversations with experts, Interactive online gaming...

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

*All versions requires sgcAppRTC.htm file for web-browser clients.

Commands

Command	Format	Comments
help	sgcWebRTCServer help	Shows a list of supported commands with examples
exit	sgcWebRTCServer exit	exits application
start server	{"message":"start", "params":{"host": "127.0.0.1", "port": 80}}	start server listening on port 80 and ip 127.0.0.1

broadcast message	{"message": "broadcast", "params": {"text": "My First Message"}}	broadcast "My First Message" to all clients connected
send message	{"message": "write", "params": {"guid": "EDKSKJDFAAJDJ38437", "text": "My First Message"}}	sends a message to a single client
stop server	{"message": "stop"}	stops server
enable ssl	{"message": "ssl", "params": {"ssl": {"enabled": true, "certfile": "sgc.pem", "keyfile": "sgc.pem", "rootcertfile": "sgc.pem", "password": "", "port": 443}}}	enable ssl connections using sgc.pem as certificate and port 443
enable heartbeat	{"message": "heartbeat", "params": {"heartbeat": {"enabled": true, "interval": 30}}}	enable heartbeats every 30 seconds
enable watchdog	{"message": "watchdog", "params": {"watchdog": {"enabled": true, "interval": 10}}}	automatically reconnects after 10 seconds

Events

Event	Format	Comments
connect	{"event": "connect", "protocol": "webrtc", "params": {"guid": "SDFKSJDFJ33223"}}	New client connects
disconnect	{"event": "disconnect", "protocol": "webrtc", "code": 0, "params": {"guid": "SDFKSJDFJ33223"}}	Client disconnects
error	{"event": "error", "protocol": "webrtc", "description": "authentication error", "params": {"guid": "SDFKSJDFJ33223"}}	Error in a connection
message	{"event": "message", "protocol": "webrtc", "text": "Message from server", "params": {"guid": "SDFKSJDFJ33223"}}	Message received from client
subscription	{"event": "subscribed", "protocol": "webrtc", "params": {"guid": "SDFKSJDFJ33223", "subscription": "MyChannel"}}	subscribed to MyChannel
unsubscription	{"event": "unsubscribed", "protocol": "webrtc",	unsubscribed from MyChannel

<pre>"params":{"guid":"SDFKSJDFJ33223", "subscription": "MyChannel"}}</pre>

Console | sgcSTOMPClient

This guide describes the use of the **sgcSTOMPClient** command-line tool.

Features

1. **STOMP Protocol:** STOMP provides an interoperable wire format so that STOMP clients can communicate with any STOMP message broker to provide easy and widespread messaging interoperability among many languages, platforms and brokers.
2. **STOMP is simple:** Simple (or Streaming) Text Oriented Message Protocol (STOMP), is a simple text-based protocol
3. **Message-oriented Middleware:** Designed for working with message-oriented middleware (MOM)
4. **Interoperable wire format:** It provides an interoperable wire format that allows STOMP clients to talk with any message broker supporting the protocol.

Platforms Supported

Platform	Requirements
Win32	
Win64	
OSX32	libcgunwind.1.0.dylib, libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
OSX64	libcrypto.1.0.0.dylib and libssl.1.0.0.dylib
Linux	openssl 1.0.2 version

Commands

Command	Format	Comments
help	sgcSTOMPClient help	Shows a list of supported commands with examples
exit	sgcSTOMPClient exit	exits application
connect to STOMP server	{"message":"open", "params": {"url": "ws://www.esegece.com:15674/ws"}}	connects to STOMP server
subscribe	{"message":"subscribe", "protocol":"stomp", "params":	subscribe to topic "sgc"

	<code>{"id": "12345", "destination": "/topic/sgc"}}</code>	
send message	<code>{"message": "send", "protocol": "stomp", "params": {"destination": "/topic/sgc", "text": "My First Message"}}</code>	send message "My First Message" in channel "myTopic"
unsubscribe	<code>{"message": "unsubscribe", "protocol": "stomp", "params": {"id": "12345"}}</code>	unsubscribe from topic "sgc"
close	<code>{"message": "disconnect", "protocol": "stomp"}</code>	close connection
authentication	<code>{"message": "authentication", "protocol": "stomp", "params": {"enabled": true, "username": "user", "password": "secret"}}</code>	enable authentication
heartbeat	<code>{"message": "heartbeat", "protocol": "stomp", "params": {"enabled": true, "incoming": 30, "outgoing": 30}}</code>	enable heartbeat every 30 seconds

Events

Event	Format	Comments
connect	<code>{"event": "connect", "protocol": "stomp", "params": {"version": "1.2", "server": "Server", "session": "sdf4sd5f4fs4df", "heartbeat": "30"}}</code>	client connects
disconnect	<code>{"event": "disconnect", "protocol": "stomp", "params": {"code": 0}}</code>	Client disconnects
error	<code>{"event": "error", "protocol": "stomp", "params": {"text": "Error", "content-type": "text/html", "content-length": 5, "receipt-id": "asdjjkj3kj323"}}</code>	Error in connection
message	<code>{"event": "message", "protocol": "stomp", "params": {"text": "Message From SErver", "destination": "/topic/sgc", "id": "1", "subscription": "55113", "ack": ""}}</code>	Message received from server
receipt	<code>{"event": "receipt", "protocol": "stomp", "params": {"receipt-id": "32kj3kj22"}}</code>	receipt from server