SGCWEBSOCKETS · TECHNICAL DOCUMENT

Proxy Server

TsgcWebSocketProxyServer — reverse-proxy for sgcWebSockets traffic with TLS termination, request rewriting and connection pooling.

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcWebSocketProxyServer implements a WebSocket Server Component that listens for client WebSocket connections and forwards data connections to a normal TCP/IP server.

At a glance

COMPONENT CLASS
`TsgcWebSocketProxyServer`

STANDARDS / SPEC
—

TRANSPORTS
TCP, TLS

PLATFORMS
Windows, macOS, Linux, iOS, Android

FRAMEWORKS
VCL, FireMonkey, Lazarus / FPC

EDITION
Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Component class	`TsgcWebSocketProxyServer` (unit `sgcWebSocket_Proxy`)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`Authentication`	Enables and configures user/password authentication for incoming WebSocket clients connecting to the proxy.
`Active`	Starts or stops the proxy, opening the listening sockets that accept downstream WebSocket clients.
`Port`	TCP port on which the proxy accepts incoming downstream WebSocket connections.
`Proxy`	Identifies the upstream target — the real WebSocket/TCP server to which every accepted client is transparently forwarded.
`SSLOptions`	Holds certificate paths, TLS version selection and OpenSSL tuning for the downstream TLS listener.
`SecurityOptions`	Defines admission rules such as allowed origins for browser WebSocket handshakes reaching the proxy.
`Options`	Bundles miscellaneous proxy behaviour flags: fragment handling, timeouts, HTTP test pages and UTF-8 validation.
`ThreadPoolOptions`	Configures the size and upper bound of the reusable thread pool used when ThreadPool is enabled.
`Throttle`	Caps the bandwidth (bits per second) that the proxy reads from or writes to each downstream connection.
`Bindings`	Collection of IP/Port pairs the proxy listens on for incoming downstream WebSocket clients.

Main methods

The principal public methods exposed by the component.

`Start()`	Starts the proxy server from a secondary thread so the calling thread is not blocked while bindings are opened.
`Stop()`	Stops the proxy server from a secondary thread so the calling thread is not blocked while connections are closed.
`ReStart()`	Stops and then restarts the proxy server from a secondary thread, useful after changing bindings, ports, or the upstream target at runtime.
`DisconnectAll()`	Disconnects every active WebSocket client and its paired upstream TCP link while keeping the proxy listening for new connections.
`WriteData()`	Sends a message to a single WebSocket client on the proxy identified by its connection GUID.
`Ping()`	Sends a WebSocket ping frame to every connected client on the browser-facing side of the proxy.
`Broadcast()`	Sends the same message to all connected WebSocket clients on the proxy, optionally filtered by channel, protocol, or connection GUID list.

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

`OnAuthentication`	Fires when authentication is enabled so the application can check user and password and accept or reject the downstream client connection.
`OnBeforeHeartBeat`	Fires before each HeartBeat ping so the application can implement a custom keep-alive for a downstream WebSocket client.
`OnBinary`	Fires every time a downstream WebSocket client sends a binary message, before the proxy forwards it upstream.
`OnConnect`	Fires every time a WebSocket connection is established with a downstream client and the proxy opens the upstream TCP session.
`OnDisconnect`	Fires every time a WebSocket connection with a downstream client is dropped and the upstream TCP session is closed.

`OnError`	Fires whenever a WebSocket protocol error occurs on a downstream client connection handled by the proxy.
`OnException`	Fires whenever an unhandled exception is raised while processing a downstream connection or its upstream forwarding.
`OnFragmented`	Fires when a fragment of a message is received from a downstream client (only when Options.FragmentedMessages is frgAll or frgOnlyFragmented).
`OnHandshake`	Fires after the handshake with a downstream client is evaluated and before the response is sent.
`OnLoadBalancerConnect`	TsgcWebSocketProxyServer › Events › OnLoadBalancerConnect
`OnLoadBalancerDisconnect`	TsgcWebSocketProxyServer › Events › OnLoadBalancerDisconnect
`OnLoadBalancerError`	Fires when an error occurs communicating with the Load Balancer Server.
`OnMessage`	Fires every time a downstream WebSocket client sends a text message, before the proxy forwards it upstream.
`OnSSLALPNSelect`	Fires during an ALPN-enabled handshake with a downstream client so the application can pick which protocol to negotiate.
`OnSSLAfterCreateHandler`	TsgcWebSocketProxyServer › Events › OnSSLAfterCreateHandler
`OnSSLGetHandler`	Fires before the SSL handler is created so a custom server-side handler instance can be supplied.
`OnSSLVerifyPeer`	Fires when VerifyCertificate is enabled and a downstream client presents a certificate to be accepted or rejected.
`OnShutdown`	Fires after the proxy server has stopped and no more connections are accepted.
`OnStartup`	Fires after the proxy server has started and is ready to accept connections.
`OnTCPConnect`	Fires after a downstream client connects at TCP level and before the WebSocket handshake, so the connection can be accepted or rejected.
`OnUnknownAuthentication`	TsgcWebSocketProxyServer › Events › OnUnknownAuthentication

`OnUnknownProtocol`	Fires when the first message from a downstream client does not match a known protocol so the connection can be accepted or rejected.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **TsgcWebSocketProxyServer.Authentication — Authentication** configuration sourced from the online help.

> **About this scenario.** Enables and configures user/password authentication for incoming WebSocket clients connecting to the proxy.

Delphi (VCL / FireMonkey)

```
oProxy := TsgcWebSocketProxyServer.Create(nil);
oProxy.Authentication.Enabled := true;
oProxy.Authentication.Basic.Enabled := true;
oProxy.Authentication.AuthUsers.Add('user=secret');
oProxy.Proxy.Host := 'upstream.example.com';
oProxy.Proxy.Port := 8080;
oProxy.Active := true;
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · OnAuthentication

Fires when authentication is enabled so the application can check user and password and accept or reject the downstream client connection.

Delphi (VCL / FireMonkey)

```
procedure OnAuthentication(Connection: TsgcWSConnection; aUser, aPassword: string;
  var Authenticated: Boolean);
begin
  if ((aUser = 'user') and (aPassword = 'secret')) then
    Authenticated := True
  else
    Authenticated := False;
end;
```

C++ Builder

```
void OnAuthentication(TsgcWSConnection *Connection, string aUser, string aPassword,
  bool &Authenticated)
{
  if ((aUser == "user") && (aPassword == "secret"))
    Authenticated = true;
  else
    Authenticated = false;
}
```

.NET (C#)

```
void OnAuthentication(TsgcWSConnection Connection, string aUser, string aPassword,
  ref bool Authenticated)
{
  if ((aUser == "user") && (aPassword == "secret"))
    Authenticated = true;
  else
    Authenticated = false;
}
```

2 · OnUnknownAuthentication

Fires when authentication is enabled and the authentication method of a downstream client is not recognized by the server.

Delphi (VCL / FireMonkey)

```
procedure OnUnknownAuthentication(Connection: TsgcWSConnection; AuthType, AuthData: string;
  var aUser, aPassword: string; var Authenticated: Boolean);
begin
  if AuthType = 'Bearer' then
  begin
    if AuthData = 'jwt_token' then
      Authenticated := True
    else
      Authenticated := False;
  end
  else
    Authenticated := False;
end;
```

C++ Builder

```
void OnUnknownAuthentication(TsgcWSConnection *Connection, string AuthType, string AuthData,
  string &aUser, string &aPassword, bool &Authenticated)
{
  if (AuthType == "Bearer")
  {
    if (AuthData == "jwt_token")
      Authenticated = true;
    else
      Authenticated = false;
  }
  else
    Authenticated = false;
}
```

.NET (C#)

```
void OnUnknownAuthenticationEvent(TsgcWSConnection Connection, string AuthType, string AuthData,
  ref string User, ref string Password, ref bool Authenticated)
{
  if (AuthType == "Bearer")
  {
    if (AuthData == "jwt_token")
      Authenticated = true;
    else
      Authenticated = false;
  }
  else
    Authenticated = false;
}
```

3 · DisconnectAll

Disconnects every active WebSocket client and its paired upstream TCP link while keeping the proxy listening for new connections.

Delphi (VCL / FireMonkey)

```
oServer.DisconnectAll;
```

C++ Builder

```
oServer->DisconnectAll();
```

.NET (C#)

```
oServer.DisconnectAll();
```

4 · OnBeforeHeartBeat

Fires before each HeartBeat ping so the application can implement a custom keep-alive for a downstream WebSocket client.

Delphi (VCL / FireMonkey)

```
procedure OnBeforeHeartBeat(Sender: TObject; const Connection: TsgcWSConnection;
  var Handled: Boolean);
begin
  Connection.WriteData('ping');
  Handled := True;
end;
```

C++ Builder

```
void OnBeforeHeartBeat(TObject *Sender, const TsgcWSConnection *Connection,
  bool &Handled)
{
  Connection->WriteData("ping");
  Handled = true;
}
```

.NET (C#)

```
void OnBeforeHeartBeat(TObject Sender, TsgcWSConnection Connection, out bool Handled)
{
  Connection.WriteData("ping");
  Handled = true;
}
```

5 · OnBinary

Fires every time a downstream WebSocket client sends a binary message, before the proxy forwards it upstream.

Delphi (VCL / FireMonkey)

```
procedure OnBinary(Connection: TsgcWSConnection; const Data: TMemoryStream);
var
  oBitmap: TBitmap;
begin
  oBitmap := TBitmap.Create;
  try
    oBitmap.LoadFromStream(Data);
    Image1.Picture.Assign(oBitmap);
  finally
    FreeAndNil(oBitmap);
  end;
end;
```

C++ Builder

```
void OnBinary(TsgcWSConnection *Connection, const TMemoryStream *Data)
{
  TBitmap *oBitmap = new TBitmap();
  oBitmap->LoadFromStream(Data);
  Image1->Picture->Assign(oBitmap);
  delete oBitmap;
}
```

.NET (C#)

```
private void OnBinary(TsgcWSConnection Connection, byte[] Bytes)
{
  MemoryStream stream = new MemoryStream(Bytes);
  pictureBox1.Image = new Bitmap(stream);
}
```

6 · OnConnect

Fires every time a WebSocket connection is established with a downstream client and the proxy opens the upstream TCP session.

Delphi (VCL / FireMonkey)

```
procedure OnConnect(Connection: TsgcWSConnection);
begin
  Log('Client connected: ' + Connection.Guid + ' from ' + Connection.PeerIP);
end;
```

C++ Builder

```
void OnConnect(TsgcWSConnection *Connection)
{
  Log("Client connected: " + Connection->Guid + " from " + Connection->PeerIP);
}
```

.NET (C#)

```
void OnConnect(TsgcWSConnection Connection)
{
  Console.WriteLine("Client connected: " + Connection.Guid + " from " + Connection.PeerIP);
}
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Online help — component page	www.esegece.com/help/sgcWebSockets/Components/TsgcWebSocketProxyServer.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\04.WebSocket_Other_Samples\05.Proxy_Server`
Component page	www.esegece.com/products/websockets/components/scaling/proxy-server/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the Proxy Server component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.