SGCWEBSOCKETS · TECHNICAL DOCUMENT

WebRTC Protocol

WebRTC signalling server for browser-paired peer-to-peer audio, video and data channels — Delphi server pairs with the standard W3C WebRTC JavaScript API.

eSeGeCe

WWW.ESEGECE.COM

Overview

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. The RTC in WebRTC stands for Real-Time Communications, a technology that enables audio/video streaming and data sharing between browser clients (peers). As a set of standards, WebRTC provides any browser with the ability to share application data and perform teleconferencing peer to peer, without the need to install plug-ins or third-party software.

At a glance

COMPONENT CLASS `TsgcWSPServer_WebRTC`	STANDARDS / SPEC **WebRTC 1.0 — W3C Recommendation**
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

Standards & specs	WebRTC 1.0 — W3C Recommendation · SDP — RFC 8866 · ICE — RFC 8445
Component class	`TsgcWSPServer_WebRTC` (unit `sgcWebSocket_Protocol_WebRTC_Server`)
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.

`Server`	References the TsgcWebSocketServer that delivers the WebRTC signalling subprotocol over WebSocket connections.
`Broker`	References a raw-TCP broker server so signalling peers can exchange SDP and ICE payloads over a plain socket instead of a WebSocket.
`WebRTC`	Signalling configuration: the list of STUN/TURN ICE server URIs pushed to joining peers and the hang-up policy for closed sessions.
`Guid`	Unique identifier that tags subscription channels owned by this WebRTC signalling handler.
`Version`	Read-only string with the sgcWebSockets build version of the WebRTC signalling server subprotocol component.

Main methods

The principal public methods exposed by the component.

`WriteData()`	Sends a text payload to a single connected peer identified by its connection GUID.
`Broadcast()`	Sends a text payload to every peer subscribed to the signalling channel, optionally excluding or targeting specific connections.

Public events

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

`OnBeforeSubscription`	Fires when a peer asks to join a signalling channel, giving the server the chance to accept or reject the subscription.
`OnBinary`	Fires when a peer sends a binary WebSocket frame that is not part of the JSON-text signalling protocol.
`OnConnect`	Fires on the server once a peer finishes the WebSocket handshake and attaches to the WebRTC signalling subprotocol.
`OnDisconnect`	Fires on the server when a signalling peer closes its transport or the server drops it.
`OnError`	Fires when the server detects a transport or protocol-level error on a signalling connection.
`OnException`	Fires when a Delphi exception is raised inside the server's worker threads or user event handlers.
`OnFragmented`	Fires for each fragment of a multi-frame WebSocket message received from a peer before the payload is reassembled.
`OnMessage`	Fires for incoming text frames that the signalling decoder did not route to a dedicated subscription or WebRTC relay event.
`OnRawMessage`	Fires before signalling decoding, letting the server inspect every incoming text frame and optionally suppress further processing.
`OnSubscription`	Fires after a peer subscription has been accepted and the signalling server has registered the peer on the channel.
`OnUnSubscription`	Fires when a peer leaves a signalling channel and the server has removed it from the subscriber list.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Client AMQP Connect — Basic Usage** configuration sourced from the online help.

> **About this scenario.** Connect to AMQP server without authentication. Define the AMQPOptions property values, virtual host and then set in the TsgcWebSocketClient the Host and Port of the server.

Delphi (VCL / FireMonkey)

```
oAMQP := TsgcWSPClient_AMQP.Create(nil);
oAMQP.AMQPOptions.Locale := 'en_US';
oAMQP.AMQPOptions.MaxChannels := 100;
oAMQP.AMQPOptions.MaxFrameSize := 16384;
oAMQP.AMQPOptions.VirtualHost := '/';
oAMQP.HeartBeat.Enabled := true;
oAMQP.HeartBeat.Interval := 60;

oClient := TsgcWebSocketClient.Create(nil);
oAMQP.Client := oClient;
oClient.Specifications.RFC6455 := false;
oClient.Host := 'www.esegece.com';
oClient.Port := 5672;
oClient.Active := True;
```

C++ Builder

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP->AMQPOptions->Locale = "en_US";
oAMQP->AMQPOptions->MaxChannels = 100;
oAMQP->AMQPOptions->MaxFrameSize = 16384;
oAMQP->AMQPOptions->VirtualHost = "/";
oAMQP->HeartBeat->Enabled = true;
oAMQP->HeartBeat->Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 5672;
oClient->Active = true;
```

.NET (C#)

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP.AMQPOptions.Locale = "en_US";
oAMQP.AMQPOptions.MaxChannels = 100;
oAMQP.AMQPOptions.MaxFrameSize = 16384;
oAMQP.AMQPOptions.VirtualHost = "/";
oAMQP.HeartBeat.Enabled = true;
oAMQP.HeartBeat.Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP.Client = oClient;
oClient.Specifications.RFC6455 = false;
oClient.Host = "www.esegece.com";
oClient.Port = 5672;
oClient.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 · Client AMQP1 Connect — Basic Usage

Connect to an AMQP 1.0.0 server without authentication. Define the AMQPOptions property values, virtual host and then set in the TsgcWebSocketClient the Host and Port of the server.

Delphi (VCL / FireMonkey)

```
// Creating AMQP client
oAMQP := TsgcWSPClient_AMQP1.Create(nil);
// Creating WebSocket client
oClient := TsgcWebSocketClient.Create(nil);
// Setting WebSocket specifications
oClient.Specifications.RFC6455 := False;
// Setting WebSocket client properties
oClient.Host := 'amqp_host_address';
oClient.Port := 5672;
// Assigning WebSocket client to AMQP client
oAMQP.Client := oClient;
// Activating WebSocket client
oClient.Active := True;
```

C++ Builder

```
// Creating AMQP client
oAMQP = new TsgcWSPClient_AMQP1(this);
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient->Specifications->RFC6455 = false;
// Setting WebSocket client properties
oClient->Host = L"amqp_host_address";
oClient->Port = 5672;
// Assigning WebSocket client to AMQP client
oAMQP->Client = oClient;
// Activating WebSocket client
oClient->Active = true;
```

.NET (C#)

```
oAMQP = new TsgcWSPClient_AMQP1(this);
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient.Specifications.RFC6455 = false;
// Setting WebSocket client properties
oClient.Host = "amqp_host_address";
oClient.Port = 5672;
// Assigning WebSocket client to AMQP client
oAMQP.Client = oClient;
// Activating WebSocket client
oClient.Active = true;
```

2 · Client MQTT Connect — Basic Usage

Connect to Mosquitto MQTT server using websocket protocol. Subscribe to topic: "topic1" after connect.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oClient.Host := 'test.mosquitto.org';
oClient.Port := 8080;
oMQTT := TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client := oClient;
oClient.Active := True;

procedure OnMQTTConnect(Connection: TsgcWSConnection; const Session: Boolean; const ReasonCode:
 const ReasonName: string; const ConnectProperties: TsgcWSMQTTCONNACKProperties);
begin
oMQTT.Subscribe('topic1');
end;
```

C++ Builder

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
 const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
oMQTT->Subscribe("topic1");
}
```

.NET (C#)

```
oClient = new TsgcWebSocketClient();
oClient.Host = "test.mosquitto.org";
oClient.Port = 8080;
oMQTT = TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client = oClient;
oClient.Active = true;

void OnMQTTConnect(TsgcWSConnection Connection, bool Session, int ReasonCode,
 string ReasonName, TsgcWSMQTTCONNACKProperties ConnectProperties);
{
oMQTT.Subscribe("topic1");
}
```

3 · Subscribe QoS = At Least Once

You can Subscribe to a Topic using method Subscribe from TsgcWSPClient_MQTT. This method has the following parameters:

Delphi (VCL / FireMonkey)

```
MQTT.Subscribe('topic1', mtqsAtLeastOnce);
```

C++ Builder

```
MQTT->Subscribe("topic1", mtqsAtLeastOnce);
```

.NET (C#)

```
MQTT.Subscribe("topic1", TmqttQoS.mtqsAtLeastOnce);
```

4 · Subscribe Topic

Subscribe to Topic "topic1" after a successful connection.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oClient.Host := 'test.mosquitto.org';
oClient.Port := 8080;
oMQTT := TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client := oClient;
oClient.Active := True;

procedure OnMQTTConnect(Connection: TsgcWSConnection; const Session: Boolean; const ReasonCode:
 const ReasonName: string; const ConnectProperties: TsgcWSMQTTCONNACKProperties);
begin
oMQTT.Subscribe('topic1');
end;
```

C++ Builder

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
 const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
oMQTT->Subscribe("topic1");
}
```

.NET (C#)

```
oClient = new TsgcWebSocketClient();
oClient.Host = "test.mosquitto.org";
oClient.Port = 8080;
oMQTT = TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client = oClient;
oClient.Active = true;

void OnMQTTConnect(TsgcWSConnection Connection, bool Session, int ReasonCode, string ReasonName,
 TsgcWSMQTTCONNACKProperties ConnectProperties);
{
oMQTT→Subscribe("topic1");
}
```

5 · Publish Messages

The method PublishMessages is used to send a message to the AMQP server.

Delphi (VCL / FireMonkey)

```
AMQP.PublishMessage('channel_name', 'exchange_name', 'routing_key', 'Hello from sgcWebSockets!!!

procedure OnAMQPBasicReturn(Sender: TObject; const aChannel: string;
 const aReturn: TsgcAMQPFramePayload_Method_BasicReturn;
 const aContent: TsgcAMQPMessageContent);
begin
DoLog('#AMQP_basic_return: ' + aChannel + ' ' + IntToStr(aReturn.ReplyCode) + ' ' + aReturn.Repl
end;
```

C++ Builder

```
AMQP→PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!

private void OnAMQPBasicReturn(TObject *Sender, const string aChannel,
 const TsgcAMQPFramePayload_Method_BasicReturn *aReturn,
 const TsgcAMQPMessageContent *aContent)
{
DoLog("#AMQP_basic_return: " + aChannel + " " + IntToStr(aReturn→ReplyCode) + " " + aReturn→Re
}
```

.NET (C#)

```
AMQP.PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!!

private void OnAMQPBasicReturn(TObject Sender, const string aChannel,
  const TsgcAMQPFramePayload_Method_BasicReturn aReturn,
  const TsgcAMQPMessageContent aContent)
{
DoLog("#AMQP_basic_return: " + aChannel + " " + aReturn.ReplyCode.ToString() + " " + aReturn.Rep
}
```

6 · Publish a simple message

You can publish messages to all subscribers of a Topic using Publish method, which has the following parameters:

Delphi (VCL / FireMonkey)

```
MQTT.Publish('topic1', 'Hello Subscribers topic1');
```

C++ Builder

```
MQTT->Publish("topic1", "Hello Subscribers topic1");
```

.NET (C#)

```
MQTT.Publish("topic1", "Hello Subscribers topic1");
```

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.

Primary standard / spec — WebRTC 1.0 — W3C Recommendation	www.w3.org/TR/webrtc/
Primary standard / spec — SDP — RFC 8866	datatracker.ietf.org/doc/html/rfc8866
Primary standard / spec — ICE — RFC 8445	datatracker.ietf.org/doc/html/rfc8445
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/WebRTC/Protocol_WebRTC.htm
Online help — class reference (TsgcWSPServer_WebRTC)	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/WebRTC/TsgcWSPServer_WebRTC.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\30.WebRTC_Protocol\03.WebRTC`
Component page	www.esegece.com/products/websockets/protocols/webrtc/
Product page	www.esegece.com/products/websockets/

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