SGCWEBSOCKETS · TECHNICAL DOCUMENT

Custom Protocol — End-to-End Encryption

End-to-end encrypted custom subprotocol — peer-to-peer keys never reach the server, perfect for sensitive payloads.

eSeGeCe

WWW.ESEGECE.COM

Overview

End-to-End Encryption (E2EE) means that messages are encrypted on the sender device and can be decrypted only on recipient devices. The server routes packets but cannot read plaintext content.

At a glance

COMPONENT CLASS `TsgcWSPClient_E2EE`	STANDARDS / SPEC —
TRANSPORTS TCP, TLS	PLATFORMS Windows, macOS, Linux, iOS, Android
FRAMEWORKS VCL, FireMonkey, Lazarus / FPC, .NET	EDITION Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Component class	`TsgcWSPClient_E2EE` (unit `sgcWebSocket_Protocol_E2EE_Client`)
Frameworks	VCL, FireMonkey, Lazarus / FPC, .NET
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.

`Client`	WebSocket client component used as transport for the E2EE subprotocol.
`Broker`	In-memory broker used for PubSub, RPC and QoS when the E2EE client participates in sgc message routing.
`E2EE_Options`	Client-side end-to-end encryption options: local UserId, key/algorithm settings and acknowledgment flags.
`Guid`	Unique identifier assigned to this protocol instance.
`Version`	Read-only E2EE subprotocol version string.

Main methods

The principal public methods exposed by the component.

`Subscribe()`	Subscribes the client to a channel on the in-memory Broker.
`UnSubscribe()`	Unsubscribes the client from a channel on the in-memory Broker.
`SendDirectMessage()`	Sends an encrypted direct message (text, stream or bytes) to a remote user.
`SendGroupMessage()`	Sends an encrypted message (text, stream or bytes) to all online members of a group.
`DeleteGroup()`	Deletes an existing encrypted group.
`WriteData()`	Sends raw text or a stream through the underlying WebSocket connection.

`CreateGroup()`	Creates a new encrypted group on the server.
`JoinGroup()`	Joins an existing encrypted group to receive membership and key context.
`LeaveGroup()`	Leaves a group the local user is currently a member of.

Public events

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

`OnConnect`	Fired when the underlying WebSocket connection is established.
`OnDisconnect`	property OnDisconnect: TsgcWSDisconnectEvent; // TsgcWSDisconnectEvent = procedure(Connection: TsgcWSConnection; Code: Integer) of object __property TsgcWSDisconnectEvent OnDisconnect; // typedef void...
`OnE2EEError`	Fired when the remote peer or the E2EE layer reports a protocol error.
`OnE2EEGroupCreated`	property OnE2EEGroupCreated: TsgcWSE2EEOnGroupCreated; // TsgcWSE2EEOnGroupCreated = procedure(Sender: TObject; const aGroup: string) of object __property TsgcWSE2EEOnGroupCreated OnE2EEGroupCreated; ...
`OnE2EEGroupDeleted`	property OnE2EEGroupDeleted: TsgcWSE2EEOnGroupDeleted; // TsgcWSE2EEOnGroupDeleted = procedure(Sender: TObject; const aGroup: string) of object __property TsgcWSE2EEOnGroupDeleted OnE2EEGroupDeleted; ...
`OnE2EEGroupJoin`	Fired when the local user joins a group; reports the current member list.
`OnE2EEGroupLeave`	property OnE2EEGroupLeave: TsgcWSE2EEOnGroupLeave; // TsgcWSE2EEOnGroupLeave = procedure(Sender: TObject; const aGroup: string) of object __property TsgcWSE2EEOnGroupLeave OnE2EEGroupLeave; // typedef...
`OnE2EEGroupMemberJoin`	Fired when another user joins a group the local user belongs to.
`OnE2EEGroupMemberLeave`	Fired when another user leaves a group the local user belongs to.
`OnE2EEGroupMessageAck`	Fired when the server or a peer acknowledges a group message.
`OnE2EEGroupMessageBinary`	property OnE2EEGroupMessageBinary: TsgcWSE2EEOnGroupMessageBinary; //

	TsgcWSE2EEOnGroupMessageBinary = procedure(Sender: TObject; const aGroup, aFrom: string; const aBytes: TBytes) of object __propert...
`OnE2EEGroupMessageText`	property OnE2EEGroupMessageText: TsgcWSE2EEOnGroupMessageText; // TsgcWSE2EEOnGroupMessageText = procedure(Sender: TObject; const aGroup, aFrom, aText: string) of object __property TsgcWSE2EEOnGroupMe...
`OnE2EEMessageAck`	property OnE2EEMessageAck: TsgcWSE2EEOnMessageAckEvent; // TsgcWSE2EEOnMessageAckEvent = procedure(Sender: TObject; const aId, aFrom, aTo, aState: string) of object __property TsgcWSE2EEOnMessageAckEv...
`OnE2EEMessageBinary`	Fired when a decrypted direct binary message is received from another user.
`OnE2EEMessageText`	Fired when a decrypted direct text message is received from another user.
`OnE2EEUserCreated`	property OnE2EEUserCreated: TsgcWSE2EEClientOnUserCreated; // TsgcWSE2EEClientOnUserCreated = procedure(Sender: TObject; const aUserId: string) of object __property TsgcWSE2EEClientOnUserCreated OnE2E...
`OnE2EEUserDeleted`	property OnE2EEUserDeleted: TsgcWSE2EEClientOnUserDeleted; // TsgcWSE2EEClientOnUserDeleted = procedure(Sender: TObject; const aUserId: string) of object __property TsgcWSE2EEClientOnUserDeleted OnE2E...
`OnError`	Fired for transport-level errors on the underlying WebSocket connection.
`OnException`	Fired when an unhandled exception is raised while processing E2EE traffic.

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.

Online help — component page	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/E2EE/Protocol_E2EE.htm
Online help — class reference (TsgcWSPClient_E2EE)	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/E2EE/TsgcWSPClient_E2EE.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\02.WebSocket_Protocols\12.E2EE`
.NET demo project (in the sgcWebSockets package)	`.net\demos\02.WebSocket_Protocols\12.E2EE`
Component page	www.esegece.com/products/websockets/protocols/custom-e2ee/
Product page	www.esegece.com/products/websockets/

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