SGCWEBSOCKETS · TECHNICAL DOCUMENT

JWT Client

TsgcHTTP_JWT_Client — JSON Web Token client helper for any sgcWebSockets HTTP component.

eSeGeCe

WWW.ESEGECE.COM

Overview

Configure which openSSL libraries you will use when using JWT client.

At a glance

COMPONENT CLASS `TsgcHTTP_JWT_Client`	STANDARDS / SPEC **JSON Web Token — RFC 7519**
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	JSON Web Token — RFC 7519 · JSON Web Signature — RFC 7515 · JSON Web Encryption — RFC 7516
Component class	`TsgcHTTP_JWT_Client` (unit `sgcHTTP_JWT_Client`)
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.

`JWTOptions`	Token configuration: algorithm, signing key material, standard claims and custom header/payload entries.
`Version`	Read-only string exposing the sgcWebSockets library version.

Main methods

The principal public methods exposed by the component.

`Start()`	Signs the configured JWT and delivers it as a Bearer token to the host HTTP / WebSocket client.
`Sign()`	Builds, signs and returns the encoded JWT (header.payload.signature) as a single compact-serialization string.

Quick Start

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

> **About this scenario.** TsgcWebSocketClient allows the use of JWT when connecting to WebSocket servers, just create a new JWT client and assign to Authentication.Token.JWT property.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
    oClient.URL := 'ws://www.esegece.com:2052';
    <br/>

    oJWT := TsgcHTTP_JWT_Client.Create(nil);
    oJWT.JWTOptions.Header.alg := jwtRS256;
    oJWT.JWTOptions.Payload.sub := '1234567890';
    oJWT.JWTOptions.Payload.iat := 1516239022;
    <br/>

    oClient.Authentication.Enabled := True;
    oClient.Authentication.URL.Enabled := False;
    oClient.Authentication.Token.Enabled := True;
    oClient.Authentication.Token.JWT := oJWT;
    oClient.Active := True;
```

C++ Builder

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
    oClient->URL = "ws://www.esegece.com:2052";
    <br/>

    TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
    oJWT->JWTOptions->Header.alg = jwtRS256;
    oJWT->JWTOptions->Payload.sub = "1234567890";
    oJWT->JWTOptions->Payload.iat = 1516239022;
    <br/>

    oClient->Authentication->Enabled = true;
    oClient->Authentication->URL->Enabled = false;
    oClient->Authentication->Token->Enabled = true;
    oClient->Authentication->Token->JWT = oJWT;
    oClient->Active = true;
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
    oClient.URL = "ws://www.esegece.com:2052";
    <br/>

    TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
    oJWT.JWTOptions.Header.alg = jwtRS256;
    oJWT.JWTOptions.Payload.sub = "1234567890";
    oJWT.JWTOptions.Payload.iat = 1516239022;
    <br/>

    oClient.Authentication→Enabled = true;
    oClient.Authentication.URL.Enabled = false;
    oClient.Authentication.Token.Enabled = true;
    oClient.Authentication.Token.JWT = oJWT;
    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 · Custom Headers

The Header and Payload properties contains the most common headers used to generate a JWT, but you can add more headers calling the method AddKeyValue and passing the Key and Value as parameters.

```
Delphi (VCL / FireMonkey)
```

```
Header.AddKeyValue('name', 'John Smith');
```

```
C++ Builder
```

```
Header->AddKeyValue("name", "John Smith");
```

```
.NET (C#)
```

```
Header.AddKeyValue("name", "John Smith");
```

2 · Create JWT Signature

You can create JWT Signatures manually to use on applications that doesn't make use of WebSocket or HTTP Protocol, or if you are using components from third-parties applications and you only need the JWT Token.

```
Delphi (VCL / FireMonkey)
```

```
oJWT := TsgcHTTP_JWT_Client.Create(nil);
    // ... header
    oJWT.JWTOptions.Header.alg := jwtHS256;
    oJWT.JWTOptions.Algorithms.HS.Secret := '79F66F1E-E998-436B-8A0A-3E5DEFA4FD9E';
    // ... payload
    oJWT.JWTOptions.Payload.jti := '9B66FB94-B761-42B1-A1AF-3C44233DBE87';
    oJWT.JWTOptions.Payload.iat := 1630925658;
    oJWT.JWTOptions.Payload.iss := '2886EC7547B7BA6A9009';
    oJWT.JWTOptions.Payload.exp := 1630933158;
    // ... custom payload values
    oJWT.JWTOptions.Payload.ClearKeyValues;
    oJWT.JWTOptions.Payload.AddKeyValue('origin', 'www.yourwebsite.com');
    oJWT.JWTOptions.Payload.AddKeyValue('ip', '69.39.46.178');
    // ... get JWT Token
    ShowMessage(oJWT.Sign);
```

C++ Builder

```
TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client(this);
    // ... header
    oJWT->JWTOptions->Header->alg = jwtHS256;
    oJWT->JWTOptions->Algorithms->HS->Secret = "79F66F1E-E998-436B-8A0A-3E5DEFA4FD9E";
    // ... payload
    oJWT->JWTOptions->Payload->jti = "9B66FB94-B761-42B1-A1AF-3C44233DBE87";
    oJWT->JWTOptions->Payload->iat = 1630925658;
    oJWT->JWTOptions->Payload->iss = "2886EC7547B7BA6A9009";
    oJWT->JWTOptions->Payload->exp = 1630933158;
    // ... custom payload values
    oJWT->JWTOptions->Payload->ClearKeyValues;
    oJWT->JWTOptions->Payload->AddKeyValue("origin", "www->yourwebsite->com");
    oJWT->JWTOptions->Payload->AddKeyValue("ip", "69.39.46.178");
    // ... get JWT Token
    ShowMessage(oJWT->Sign());
```

.NET (C#)

```
TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
    // ... header
    oJWT.JWTOptions.Header.alg = jwtHS256;
    oJWT.JWTOptions.Algorithms.HS.Secret = "79F66F1E-E998-436B-8A0A-3E5DEFA4FD9E";
    // ... payload
    oJWT.JWTOptions.Payload.jti = "9B66FB94-B761-42B1-A1AF-3C44233DBE87";
    oJWT.JWTOptions.Payload.iat = 1630925658;
    oJWT.JWTOptions.Payload.iss = "2886EC7547B7BA6A9009";
    oJWT.JWTOptions.Payload.exp = 1630933158;
    // ... custom payload values
    oJWT.JWTOptions.Payload.ClearKeyValues;
    oJWT.JWTOptions.Payload.AddKeyValue("origin", "www.yourwebsite.com");
    oJWT.JWTOptions.Payload.AddKeyValue("ip", "69.39.46.178");
    // ... get JWT Token
    MessageBox.Show(oJWT.Sign());
```

3 · HTTP Clients and JWT

TsgcHTTP2Client and TsgcHTTP1Client allows the use of JWT when connecting to HTTP/2 servers, just create a new JWT client and assign to Authentication.Token.JWT property.

Delphi (VCL / FireMonkey)

```
oHTTP := TsgcHTTP2Client.Create(nil);
    <br/>

    oJWT := TsgcHTTP_JWT_Client.Create(nil);
    oJWT.JWTOptions.Header.alg := jwtRS256;
    oJWT.JWTOptions.Payload.sub := '1234567890';
    oJWT.JWTOptions.Payload.iat := 1516239022;
    <br/>

    oHTTP.Authentication.Token.JWT := oHTTP;
    oHTTP.Get('https://your.api.com');
```

C++ Builder

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
    <br/>

    TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
    oJWT->JWTOptions->Header->alg = jwtRS256;
    oJWT->JWTOptions->Payload->sub = "1234567890";
    oJWT->JWTOptions->Payload->iat = 1516239022;
    <br/>

    oHTTP->Authentication->Token->JWT = oHTTP;
    oHTTP->Get("https://your.api.com");
```

.NET (C#)

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
    <br/>

    TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
    oJWT.JWTOptions.Header.alg = jwtRS256;
    oJWT.JWTOptions.Payload.sub = "1234567890";
    oJWT.JWTOptions.Payload.iat = 1516239022;
    <br/>

    oHTTP.Authentication.Token.JWT = oHTTP;
    oHTTP.Get("https://your.api.com");
```

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 — JSON Web Token — RFC 7519	datatracker.ietf.org/doc/html/rfc7519
Primary standard / spec — JSON Web Signature — RFC 7515	datatracker.ietf.org/doc/html/rfc7515
Primary standard / spec — JSON Web Encryption — RFC 7516	datatracker.ietf.org/doc/html/rfc7516
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/HTTP/Authorization/JWT/client/TsgcHTTP_JWT_Client.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\20.HTTP_Protocol\05.JWT`
Component page	www.esegece.com/products/websockets/http/jwt-client/
Product page	www.esegece.com/products/websockets/

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