SGCWEBSOCKETS · TECHNICAL DOCUMENT

JWT Server

TsgcHTTP_JWT_Server — JSON Web Token server-side issuance and validation for sgcWebSockets HTTP and WebSocket servers.

eSeGeCe

WWW.ESEGECE.COM

Overview

The TsgcHTTP_JWT_Server component allows you to decode and validate JWT tokens received in WebSocket Handshake when using WebSocket protocol or as HTTP Header when using HTTP protocol.

At a glance

COMPONENT CLASS
`TsgcHTTP_JWT_Server`

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_Server` (unit `sgcHTTP_JWT_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.

`JWTOptions`	Server-side validator configuration: enabled signing algorithms (HS/RS/ES) with their Secret or PublicKey, plus registered-claim validations (iat, nbf, exp).
`Version`	Read-only string exposing the sgcWebSockets library version.

Main methods

The principal public methods exposed by the component.

`IsJWTUnauthorized()`	Final 401 gate: fires OnJWTUnauthorized so the application can accept the connection (CORS pre-flight, whitelisted endpoints) and returns True when the request must be rejected as Unauthorized.
`Validate()`	Decodes a JWT string and validates its signature against the configured algorithm, returning the decoded Header, Payload and any error text.
`IsJWTTokenValid()`	Validates the Bearer token carried in the request: runs the full pipeline (OnJWTBeforeRequest, signature check, claim validations, OnJWTAfterValidateToken) and returns True when the token is accepted.

Public events

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

`OnJWTAfterValidateToken`	Fired after the signature and claim validations run; inspect Header, Payload and Error, and flip the Valid flag to accept or reject the token.
`OnJWTBeforeRequest`	Fired for every incoming HTTP/WebSocket request before any JWT processing; set Cancel=True to bypass JWT validation for this connection.
`OnJWTBeforeValidateSignature`	TsgcHTTP_JWT_Server › Events › OnJWTBeforeValidateSignature
`OnJWTBeforeValidateToken`	Fired after a Bearer token is extracted and before it is validated; set Cancel=True to skip validation (the request is then treated as authorized).
`OnJWTException`	Fired when an exception is raised while decoding or validating the token so the application can log the error.
`OnJWTResponseError`	Fired just before the Unauthorized HTTP response is sent, allowing the code (default 401), text (default "Unauthorized") and headers to be customized.
`OnJWTUnauthorized`	Fired when the request has no valid JWT and is about to be rejected; set Disconnect=False to still accept it (for example, CORS pre-flight requests).

Quick Start

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

> **About this scenario.** You can configure the following properties in the JWTOptions property of the component:

Delphi (VCL / FireMonkey)

```
oServer := TsgcWebSocketHTTPServer.Create(nil);
    oServer.Port := 80;
    oJWT := TsgcHTTP_JWT_Server.Create(nil);
    oJWT.JWTOptions.Algorithms.RS.PublicKey.Text := 'public key here';
    oServer.Authorization.Enabled := True;
    oServer.Authorization.JWT.JWT := oJWT;
    oServer.Active := True;
```

C++ Builder

```
TsgcWebSocketHTTPServer oServer = new TsgcWebSocketHTTPServer();
    oServer->Port = 80;
    TsgcHTTP_JWT_Server oJWT = new TsgcHTTP_JWT_Server();
    oJWT->JWTOptions->Algorithms->RS->PublicKey->Text = "public key here";
    oServer->Authorization->Enabled = true;
    oServer->Authorization->JWT->JWT = oJWT;
    oServer->Active = true;
```

.NET (C#)

```
TsgcWebSocketHTTPServer oServer = new TsgcWebSocketHTTPServer();
    oServer.Port = 80;
    TsgcHTTP_JWT_Server oJWT = new TsgcHTTP_JWT_Server();
    oJWT.JWTOptions.Algorithms.RS.PublicKey = "public key here";
    oServer.Authorization.Enabled = true;
    oServer.Authorization.JWT.JWT = oJWT;
    oServer.Active = true;
```

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/server/TsgcHTTP_JWT_Server.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\20.HTTP_Protocol\05.JWT`
Component page	www.esegece.com/products/websockets/http/jwt-server/
Product page	www.esegece.com/products/websockets/

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