SGCWEBSOCKETS · TECHNICAL DOCUMENT

# UDP Server

TsgcUDPServer — low-level UDP datagram server for sgcWebSockets P2P stack; foundation for STUN, TURN and ICE.

eSeGeCe

WWW.ESEGECE.COM

## Overview

TsgcUDPServer implements the UDP Server based on Indy library.

## At a glance

COMPONENT CLASS

`TsgcUDPServer`

STANDARDS / SPEC

**UDP — RFC 768**

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** | UDP — RFC 768 |
| **Component class** | `TsgcUDPServer` (unit `sgcUDP_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.

| | |
|---|---|
| `DTLSOptions` | Certificate, verification and OpenSSL settings applied when DTLS is enabled on the server. |
| `Active` | Starts or stops the UDP listener; read to query whether the server is currently bound and accepting datagrams. |
| `Host` | Local IP address the server listens on when Bindings is empty; leave blank to accept on every interface. |
| `Port` | Local UDP port the server listens on when Bindings is empty. |
| `DTLS` | Enables Datagram TLS (DTLS) to encrypt the UDP traffic received and sent by this server. |
| `WatchDog` | Automatic listener-recovery monitor that restarts the server after unexpected shutdowns or missed heartbeats. |
| `Bindings` | Collection of local IP/port endpoints that the server listens on for incoming UDP datagrams. |
| `LogFile` | Writes every datagram received and sent by the server to a plain-text log file for diagnostics. |
| `NotifyEvents` | Controls how received-datagram, exception and lifecycle events are dispatched to the main thread. |

| | |
|---|---|
| `IPVersion` | Selects the IP stack (IPv4 or IPv6) used by the UDP listener when a binding is created from Host and Port. |

## Main methods

The principal public methods exposed by the component.

| | |
|---|---|
| `Start()` | Starts the server from a secondary thread so the calling thread is not blocked while bindings are opened. |
| `Stop()` | Stops the server from a secondary thread so the calling thread is not blocked while the listener is torn down. |
| `ReStart()` | Stops and then restarts the server from a secondary thread, useful after changing bindings or ports at runtime. |
| `WriteData()` | Sends a single UDP datagram from the server to a specific peer address and port, optionally from a chosen local binding. |
| `AddBinding()` | Adds a new listening endpoint (IP/port) to the server without restarting the currently active bindings. |
| `RemoveBinding()` | Removes the listening endpoint matching the given IP/port and closes its socket without stopping the server. |
| `ClearDTLS()` | Discards the cached DTLS session state so the next datagram triggers a fresh DTLS handshake with every peer. |

## Public events

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

| | |
|---|---|
| `OnBeforeWatchDog` | Fires before each WatchDog cycle so the application can inspect state and optionally suppress the automatic restart. |
| `OnDTLSVerifyPeer` | Fires during the DTLS handshake so the server can inspect and accept or reject the client certificate. |
| `OnShutdown` | Fires once the UDP listener has stopped and no further datagrams will be accepted. |
| `OnStartup` | Fires once the UDP listener is bound and ready to accept datagrams. |

| | |
|---|---|
| `OnUDPException` | Fires when the UDP listener thread catches an unhandled exception while receiving or sending datagrams. |
| `OnUDPRead` | Fires once per incoming UDP datagram, exposing the raw payload and the sender's IP/port. |

# Quick Start

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

> **About this scenario.** TsgcUDPServer implements the UDP Server based on Indy library.

### Delphi (VCL / FireMonkey)

```
oClient := TsgcUDPServer.Create(nil);
    oClient.Port := 80;
```

### C++ Builder

```
oClient = new TsgcUDPServer();
    oClient->Port = 80;
```

### .NET (C#)

```
oClient = new TsgcUDPServer();
    oClient.Port = 80;
```

## 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 · TsgcUDPClient

TsgcUDPClient implements the UDP Client based on Indy library.

Delphi (VCL / FireMonkey)

```
oClient := TsgcUDPClient.Create(nil);
    oClient.Host := '127.0.0.1';
    oClient.Port := 80;
```

C++ Builder

```
oClient = new TsgcUDPClient();
    oClient->Host = "127.0.0.1";
    oClient->Port = 80;
```

.NET (C#)

```
oClient = new TsgcUDPClient();
    oClient.Host = "127.0.0.1";
    oClient.Port = 80;
```

# 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 — UDP — RFC 768** | datatracker.ietf.org/doc/html/rfc768 |
| **Online help — component page** | www.esegece.com/help/sgcWebSockets/Components/P2P/UDP/TsgcUDPServer.htm |
| **Delphi demo project (in the sgcWebSockets package)** | `Demos\35.P2P\01.UDP_Server_Client` |
| **Component page** | www.esegece.com/products/websockets/p2p/udp-server/ |
| **Product page** | www.esegece.com/products/websockets/ |

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