SGCWEBSOCKETS · TECHNICAL DOCUMENT

STUN Server

TsgcSTUNServer — RFC 8489 STUN server for sgcWebSockets; lets behind-NAT peers discover their public reflexive address.

eSeGeCe

WWW.ESEGECE.COM

Overview

The STUN server can be configured with or without Authentication, can verify Fingerprint Attribute, send an alternate server and more.

At a glance

COMPONENT CLASS `TsgcSTUNServer`	STANDARDS / SPEC **STUN — RFC 8489**
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	STUN — RFC 8489
Component class	`TsgcSTUNServer` (unit `sgcSTUN_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.

`Active`	Starts or stops the STUN server; set to True to begin listening for Binding Requests.
`Host`	Local IP address or host name the STUN server binds to when Active is set to True.
`Port`	UDP port the STUN server listens on; defaults to 3478 as reserved by RFC 5389.
`STUNOptions`	Server-side STUN options: FINGERPRINT, SOFTWARE, Authentication and the attributes added to Binding Responses.
`LogFile`	Appends every STUN message received or sent by the server to a file for debugging.
`NotifyEvents`	Selects how threaded server events are synchronized with the main VCL/FMX thread.
`IPVersion`	Address family used by the listening socket; defaults to IPv4.
`Version`	Read-only build version of the sgcWebSockets component library.

Main methods

The principal public methods exposed by the component.

`AddBinding()`	Adds an extra listening endpoint (IP/port) to the STUN server without restarting the currently active bindings.
`RemoveBinding()`	Removes a previously added listening endpoint and closes its socket without stopping the STUN server.

Public events

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

`OnSTUNException`	Raised when an unhandled exception is caught while parsing or responding to a STUN message.
`OnSTUNRequestAuthorization`	Raised when a Binding Request requires authentication; supply the password associated with the incoming Username/Realm.
`OnSTUNRequestError`	Raised before the server sends a STUN error response; allows the handler to inspect or suppress the reply.
`OnSTUNRequestSuccess`	Raised before the server sends a successful Binding Response so the handler can inspect or veto the reply.

Quick Start

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

> **About this scenario.** Usually STUN servers run on UDP port 3478 and don't require authentication, so in order to configure a STUN server, set the listening port (by default 3478) and start the server.

Delphi (VCL / FireMonkey)

```
oSTUN := TsgcSTUNServer.Create(nil);
    oSTUN.Port := 3478;
    oSTUN.Active := True;
```

C++ Builder

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
    oSTUN->Port = 3478;
    oSTUN->Active = true;
```

.NET (C#)

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
    oSTUN.Port = 3478;
    oSTUN.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 · STUN Server | Alternate Server

The alternate server represents an alternate transport address identifying a different STUN server that the STUN client should try.

Delphi (VCL / FireMonkey)

```
oSTUN := TsgcSTUNServer.Create(nil);
oSTUN.Port := 3478;
oSTUN.STUNOptions.BindingAttributes.AlternateServer.Enabled := True;
oSTUN.STUNOptions.BindingAttributes.AlternateServer.IPAddress := '80.54.54.1';
oSTUN.STUNOptions.BindingAttributes.AlternateServer.Port := 3478;
oSTUN.Active := True;
```

C++ Builder

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
oSTUN->Port = 3478;
oSTUN->STUNOptions->BindingAttributes->AlternateServer->Enabled = true;
oSTUN->STUNOptions->BindingAttributes->AlternateServer->IPAddress = "80.54.54.1";
oSTUN->STUNOptions->BindingAttributes->AlternateServer->Port = 3478;
oSTUN->Active = true;
```

.NET (C#)

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
oSTUN.Port = 3478;
oSTUN.STUNOptions.BindingAttributes.AlternateServer.Enabled = true;
oSTUN.STUNOptions.BindingAttributes.AlternateServer.IPAddress = "80.54.54.1";
oSTUN.STUNOptions.BindingAttributes.AlternateServer.Port = 3478;
oSTUN.Active = true;
```

2 · STUN Server | Long-Term Credentials

Usually STUN Servers are configured without Authentication, so any STUN client can send a binding request and expect a response from server without Authentication.

Delphi (VCL / FireMonkey)

```
oSTUN := TsgcSTUNServer.Create(nil);
oSTUN.Port := 3478;
oSTUN.STUNOptions.Authentication.Enabled := True;
oSTUN.STUNOptions.Authentication.LongTermCredentials.Enabled := True;
oSTUN.STUNOptions.Authentication.LongTermCredentials.Realm := 'sgcWebSockets';
oSTUN.STUNOptions.Authentication.LongTermCredentials.StaleNonce := 600;
oSTUN.Active := True;

procedure OnSTUNRequestAuthorization(Sender: TObject; const aRequest: TsgcSTUN_Message;
 const aUsername, aRealm: string; var Password: string);
begin
if aUsername = 'my-user' then
Password := 'my-password';
end;
```

C++ Builder

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
oSTUN->STUNOptions->Authentication->Enabled = true;
oSTUN->STUNOptions->Authentication->LongTermCredentials->Enabled = true;
oSTUN->STUNOptions->Authentication->LongTermCredentials->Realm = "sgcWebSockets";
oSTUN->STUNOptions->Authentication->LongTermCredentials->StaleNonce = 600;
oSTUN->Port = 3478;
oSTUN->Active = true;

private void OnSTUNRequestAuthorization(TObject *Sender, const TsgcSTUN_Message *aRequest,
 const string aUsername, const string aRealm, ref string Password)
{
if (aUsername == "my-user")
{
Password = "my-password";
}
}
```

.NET (C#)

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
oSTUN.Port = 3478;
oSTUN.STUNOptions.Authentication.Enabled = true;
oSTUN.STUNOptions.Authentication.LongTermCredentials.Enabled = true;
oSTUN.STUNOptions.Authentication.LongTermCredentials.Realm = "sgcWebSockets";
oSTUN.STUNOptions.Authentication.LongTermCredentials.StaleNonce = 600;
oSTUN.Active = true;

private void OnSTUNRequestAuthorization(Component Sender, TsgcSTUN_Message aRequest,
  string aUsername, string aRealm, ref string Password)
{
if ((aUsername == "my-user") && (aRealm == "sgcWebSockets"))
{
Password = "my-password";
}
}
```

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 — STUN — RFC 8489	datatracker.ietf.org/doc/html/rfc8489
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/P2P/STUN/Server/TsgcSTUNServer.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\35.P2P\02.STUN`
Component page	www.esegece.com/products/websockets/p2p/stun-server/
Product page	www.esegece.com/products/websockets/

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