SGCWEBSOCKETS · TECHNICAL DOCUMENT

TURN Server

TsgcTURNServer — RFC 8656 TURN relay server for sgcWebSockets; relays peer-to-peer traffic when ICE cannot establish a direct path.

eSeGeCe

WWW.ESEGECE.COM

Overview

TURN Server supports Long-Term Authentication, Allocation, Permissions, Channel Data and more.

At a glance

COMPONENT CLASS `TsgcTURNServer`	STANDARDS / SPEC **TURN — RFC 8656**
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	TURN — RFC 8656
Component class	`TsgcTURNServer` (unit `sgcTURN_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 TURN server; set to True to begin accepting ALLOCATE, Binding and relay requests.
`Host`	Local IP address or host name the TURN server binds to when Active is set to True.
`Port`	UDP port the TURN server listens on; defaults to 3478 as reserved by RFC 5766.
`STUNOptions`	STUN-level options inherited from TsgcSTUNServer: FINGERPRINT, SOFTWARE, Authentication and Binding-Response attributes.
`TURNOptions`	TURN-specific options: default Allocation lifetime, port range, Relay IP and long-term credentials.
`LogFile`	Appends every STUN/TURN 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 TURN server without stopping the bindings already in place.
`RemoveBinding()`	Removes a previously added listening endpoint and closes its socket without stopping the TURN 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/TURN message.
`OnSTUNRequestAuthorization`	Raised when an authenticated request arrives; supply the password associated with the incoming Username/Realm.
`OnSTUNRequestError`	Raised before the server sends a STUN/TURN error response; allows the handler to inspect or suppress the reply.
`OnSTUNRequestSuccess`	Raised before the server sends a successful STUN/TURN response so the handler can inspect or veto the reply.
`OnTURNBeforeAllocate`	Raised before a new Allocation is created; inspect the relayed IP/port and set Reject to refuse the ALLOCATE request.
`OnTURNBeforeRelayChannelData`	Raised before the server relays a ChannelData payload to a peer; set Accept to False to drop the packet.
`OnTURNBeforeRelayIndication`	Raised before the server relays a Send-Indication payload to a peer; set Accept to False to drop the packet.
`OnTURNChannelDataDiscarded`	Raised when a ChannelData message is dropped because the channel number is invalid or has no permission.
`OnTURNCreateAllocation`	Raised after a new Allocation has been successfully created for a client.
`OnTURNDeleteAllocation`	Raised after an Allocation has been removed, either by client refresh-to-zero or because its lifetime expired.

`OnTURNMessageDiscarded`	Raised when a TURN message received by the server is discarded before any response is produced.

Quick Start

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

> **About this scenario.** Usually TURN servers run on UDP port 3478 and require Long-Term credentials, so in order to configure a TURN server, set the listening port (by default 3478) and start the server.

Delphi (VCL / FireMonkey)

```
oTURN := TsgcTURNServer.Create(nil);
    oTURN.Port := 3478;
    oTURN.TURNOptions.Authentication.Enabled := True;
    oTURN.TURNOptions.Authentication.LongTermCredentials.Enabled := True;
    oTURN.TURNOptions.Authentication.LongTermCredentials.Realm := 'esegece.com';
    oTURN.Active := True;
    procedure OnSTUNRequestAuthorization(Sender: TObject; const aRequest: TsgcSTUN_Message;
     const aUsername, aRealm: string; var Password: string);
    begin
    if (aUsername = 'user') and (aRealm = 'esegece.com') then
    Password := 'password';
    end;
```

C++ Builder

```
TsgcTURNServer oTURN = new TsgcTURNServer();
    oTURN->Port = 3478;
    oTURN->TURNOptions->Authentication->Enabled = true;
    oTURN->TURNOptions->Authentication->LongTermCredentials->Enabled = true;
    oTURN->TURNOptions->Authentication->LongTermCredentials->Realm = "esegece.com";
    oTURN->Active = true;
    void OnSTUNRequestAuthorization(TObject *Sender, const TsgcSTUN_Message *aRequest,
     const string aUsername, const string aRealm, ref string Password)
    {
    if ((aUsername == "user") && (aRealm == "esegece.com"))
    {
    Password = "password";
    }
    }
```

.NET (C#)

```
TsgcTURNServer oTURN = new TsgcTURNServer();
    oTURN.Port = 3478;
    oTURN.TURNOptions.Authentication.Enabled = true;
    oTURN.TURNOptions.Authentication.LongTermCredentials.Enabled = true;
    oTURN.TURNOptions.Authentication.LongTermCredentials.Realm = "esegece.com";
    oTURN.Active = true;
    void OnSTUNRequestAuthorization(TObject Sender, const TsgcSTUN_Message aRequest,
      const string aUsername, const string aRealm, ref string Password)
    {
    if ((aUsername == "user") && (aRealm == "esegece.com"))
    {
    Password = "password";
    }
    }
```

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 · TURN Server | Allocations

All TURN operations revolve around allocations and all TURN messages are associated with an Allocation. An allocation consists of:

Delphi (VCL / FireMonkey)

```
procedure OnTURNBeforeAllocate(Sender: TObject; const aSocket: TsgcSocketConnection;
 const aIP: string; aPort: Word; var Reject: Boolean);
begin
if not (your own rules) then
        Reject := false;
end;
```

C++ Builder

```
void OnTURNBeforeAllocate(TObject *Sender, const TsgcSocketConnection *aSocket,
 const string aIP, Word aPort, ref bool Reject)
{
if (your own rules) == false
{
        Reject = false;
}
}
```

.NET (C#)

```
void OnTURNBeforeAllocate(TObject Sender, const TsgcSocketConnection aSocket,
 const string aIP, Word aPort, ref bool Reject)
{
if (your own rules) == false
{
        Reject = false;
}
}
```

2 · TURN Server | Long Term Credentials

Usually TURN Servers are configured WITH Authentication for TURN requests and without Authentication for STUN requests.

Delphi (VCL / FireMonkey)

```
oTURN := TsgcTURNServer.Create(nil);
oTURN.Port := 3478;
oTURN.STUNOptions.Authentication.Enabled := False;
oTURN.TURNOptions.Authentication.Enabled := True;
oTURN.TURNOptions.Authentication.LongTermCredentials.Enabled := True;
oTURN.TURNOptions.Authentication.LongTermCredentials.Realm := 'sgcWebSockets';
oTURN.TURNOptions.Authentication.LongTermCredentials.StaleNonce := 600;
oTURN.Active := True;

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

C++ Builder

```
TsgcTURNServer oTURN = new TsgcTURNServer();
oTURN->STUNOptions->Authentication->Enabled = false;
oTURN->TURNOptions->Authentication->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Realm = "sgcWebSockets";
oTURN->TURNOptions->Authentication->LongTermCredentials->StaleNonce = 600;
oTURN->Port = 3478;
oTURN->Active = true;


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

.NET (C#)

```
TsgcTURNServer oTURN = new TsgcTURNServer();
oTURN.STUNOptions.Authentication.Enabled = false;
oTURN.TURNOptions.Authentication.Enabled = true;
oTURN.TURNOptions.Authentication.LongTermCredentials.Enabled = true;
oTURN.TURNOptions.Authentication.LongTermCredentials.Realm = "sgcWebSockets";
oTURN.TURNOptions.Authentication.LongTermCredentials.StaleNonce = 600;
oTURN.Port = 3478;
oTURN.Active = true;

private void OnSTUNRequestAuthorization(TObject Sender, const TsgcSTUN_Message aRequest,
  const string aUsername, const 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 — TURN — RFC 8656	datatracker.ietf.org/doc/html/rfc8656
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/P2P/TURN/Server/TsgcTURNServer.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\35.P2P\03.TURN`
Component page	www.esegece.com/products/websockets/p2p/turn-server/
Product page	www.esegece.com/products/websockets/

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