SGCWEBSOCKETS · TECHNICAL DOCUMENT

ICE Client

TsgcICEClient — RFC 8445 Interactive Connectivity Establishment client; combines STUN and TURN to pick the best peer-to-peer path.

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcICEClient is the client that implements the ICE protocol and allows you to send allocation requests to TURN servers. The client requires the TsgcTURNClient and a TsgcWebSocketClient.

At a glance

COMPONENT CLASS
`TsgcICEClient`

STANDARDS / SPEC
ICE — RFC 8445

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	ICE — RFC 8445
Component class	`TsgcICEClient` (unit `sgcICE_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.

`TURNClient`	TURN client used to allocate relayed transport addresses and obtain relay candidates during ICE gathering.
`ICEOptions`	ICE agent policies covering candidate gathering sources and the connectivity check list limits.
`Version`	Read-only build version of the sgcWebSockets component library.

Main methods

The principal public methods exposed by the component.

`GatherCandidates()`	Starts ICE candidate gathering; each discovered candidate is reported through OnICECandidate.
`AddRTCIceCandidate()`	Registers a remote ICE candidate received from the peer so it can be paired with the local candidate set.
`ProcessCandidates()`	Runs ICE connectivity checks against the queued candidate pairs and nominates the first pair that succeeds.

Public events

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

`OnICECandidate`	Fires once per local candidate discovered during gathering; forward the candidate to the remote peer via signalling.
`OnICECandidateError`	Fires when candidate gathering fails for a specific STUN or TURN probe; inspect the error details to diagnose.
`OnICECandidatePairFailed`	Fires when a candidate pair fails connectivity checks; the pair is dropped and the next one in the check list is tried.
`OnICECandidatePairNominated`	Fires when a candidate pair passes connectivity checks and is nominated as the selected path for the session.
`OnICEException`	Fires when an unhandled internal exception is raised while processing ICE candidates or connectivity checks.
`OnICEReceiveBindingRequest`	Fires for each STUN binding request received from the peer during connectivity checks; set Accept to allow the probe.

Quick Start

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

> **About this scenario.** ICEOptions.

Delphi (VCL / FireMonkey)

```
oICE := TsgcICEClient.Create(nil);
    oTURN := TsgcTURNClient.Create(nil);
    oTURN.Host := 'www.esegece.com';
    oTURN.Port := 3478;
    oTURN.TURNOptions.Authentication.Credentials := stauLongTermCredential;
    oTURN.TURNOptions.Authentication.Username := 'sgc';
    oTURN.TURNOptions.Authentication.Password := 'secret';
    oICE.GatherCandidates();
```

C++ Builder

```
TsgcICEClient *oICE = new TsgcICEClient();
    TsgcTURNClient *oTURN = new TsgcTURNClient();
    oTURN->Host = "www.esegece.com";
    oTURN->Port = 3478;
    oTURN->TURNOptions->Authentication->Credentials = stauLongTermCredential;
    oTURN->TURNOptions->Authentication->Username = "sgc";
    oTURN->TURNOptions->Authentication->Password = "secret";
    oICE->GatherCandidates();
```

.NET (C#)

```
TsgcICEClient oICE = new TsgcICEClient();
    TsgcTURNClient oTURN = new TsgcTURNClient();
    oTURN.Host = "www.esegece.com";
    oTURN.Port = 3478;
    oTURN.TURNOptions.Authentication.Credentials = stauLongTermCredential;
    oTURN.TURNOptions.Authentication.Username = "sgc";
    oTURN.TURNOptions.Authentication.Password = "secret";
    oICE.GatherCandidates();
```

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 · ICE | Gather Candidates

ICE starts gathering candidates, usually will obtain local IP Addresses, reflexive address using STUN protocol and relayed address using TURN protocol.

Delphi (VCL / FireMonkey)

```
oICE := TsgcICEClient.Create(nil);
oTURN := TsgcTURNClient.Create(nil);
oTURN.Host := 'www.esegece.com';
oTURN.Port := 3478;
oTURN.TURNOptions.Authentication.Credentials := stauLongTermCredential;
oTURN.TURNOptions.Authentication.Username := 'sgc';
oTURN.TURNOptions.Authentication.Password := 'secret';
oICE.GatherCandidates();

procedure OnICECandidate(Sender: TObject; const aCandidate: TsgcICE_Candidate);
begin
  DoLog('[#Candidate] ' + aCandidate.AsString);
end;
```

C++ Builder

```
TsgcICEClient *oICE = new TsgcICEClient();
TsgcTURNClient *oTURN = new TsgcTURNClient();
oTURN->Host = "www.esegece.com";
oTURN->Port = 3478;
oTURN->TURNOptions->Authentication->Credentials = stauLongTermCredential;
oTURN->TURNOptions->Authentication->Username = "sgc";
oTURN->TURNOptions->Authentication->Password = "secret";
oICE->GatherCandidates();

void OnICECandidate(TObject *Sender, const TsgcICE_Candidate *aCandidate)
{
  DoLog("[#Candidate] " + aCandidate->AsString);
}
```

.NET (C#)

```
TsgcICEClient oICE = new TsgcICEClient();
TsgcTURNClient oTURN = new TsgcTURNClient();
oTURN.Host = "www.esegece.com";
oTURN.Port = 3478;
oTURN.TURNOptions.Authentication.Credentials = stauLongTermCredential;
oTURN.TURNOptions.Authentication.Username = "sgc";
oTURN.TURNOptions.Authentication.Password = "secret";
oICE.GatherCandidates();

void OnICECandidate(TsgcICEClient Sender, const TsgcICE_Candidate aCandidate)
{
  DoLog("[#Candidate] " + aCandidate.AsString);
}
```

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 — ICE — RFC 8445	datatracker.ietf.org/doc/html/rfc8445
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/P2P/ICE/Client/TsgcICEClient.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\35.P2P\04.ICE`
Component page	www.esegece.com/products/websockets/p2p/ice-client/
Product page	www.esegece.com/products/websockets/

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