SGCWEBSOCKETS · TECHNICAL DOCUMENT

gRPC Client

TsgcGRPCClient — native gRPC client over HTTP/2: unary and streaming calls, custom metadata, deadlines, compression, interceptors and automatic retry.

eSeGeCe

WWW.ESEGECE.COM

Overview

gRPC is a high-performance remote-procedure-call framework: Protocol Buffers messages are length-prefixed and carried over HTTP/2 streams, request and response metadata travel as HTTP/2 headers, the call deadline is sent as the `grpc-timeout` header, and the final outcome arrives as the `grpc-status` and `grpc-message` trailers. Because it runs on HTTP/2, a single TLS connection multiplexes many concurrent calls and supports streaming in both directions.

The sgcWebSockets `TsgcGRPCClient` component is a native Delphi / C++ Builder gRPC client — no external gRPC runtime, no C library and no extra DLLs. It runs on top of the library's own `TsgcHTTP2Client` transport, frames your serialized Protocol Buffers messages, applies the gRPC headers and timeout, opens the HTTP/2 stream, and parses the response and trailers back into a typed `TsgcGRPCResponse`. All four gRPC call types are supported, along with metadata, deadlines, gzip compression, interceptors, automatic retry and typed status codes. The component is part of the **Enterprise** edition.

At a glance

COMPONENT CLASS `TsgcGRPCClient`	STANDARDS / SPEC **gRPC over HTTP/2 — RFC 9113**
TRANSPORT **HTTP/2 over TLS (OpenSSL / SChannel)**	PLATFORMS **Windows, macOS, Linux, iOS, Android**
FRAMEWORKS **VCL, FireMonkey, Lazarus / FPC**	EDITION **Enterprise**

Features

- **Four call types** — unary, server-streaming, client-streaming and bidirectional streaming, each synchronous or asynchronous.
- **Built on native HTTP/2** — uses `TsgcHTTP2Client` for ALPN `h2`, TLS via OpenSSL or SChannel, and stream multiplexing; no external gRPC runtime.

- **Protocol Buffers agnostic** — calls take and return raw `TBytes`, so you can pair the client with any Protocol Buffers library.
- **Custom metadata** — `DefaultMetadata` sends authorization or tracing headers on every call, or per call.
- **Deadlines and cancellation** — a per-call timeout maps to the `grpc-timeout` header; `CancelCall` aborts an in-flight stream.
- **Channel options** — gzip `Compression`, `ContentType` (proto or JSON), `MaxMessageSize` and `MaxMetadataSize`.
- **Interceptors, retry and service config** — an `Interceptors` chain wraps every call, `RetryPolicy` retries on configurable status codes, and `ServiceConfig` plus `MetricsCollector` add per-method policy and counters.
- **Typed status and trailers** — `grpc-status` and `grpc-message` are parsed into the standard `TsgcGRPCStatusCode` set and exposed on the response.
- **Google Cloud gRPC clients** — typed clients for Pub/Sub, Speech-to-Text, Translation, Vision, Natural Language, Cloud Storage, BigQuery and Vertex AI ship on top.

Technical specification

Standards & specs	gRPC over HTTP/2 · HTTP/2 — RFC 9113 · Protocol Buffers
Component class	`TsgcGRPCClient` (unit `sgcGRPC_Client`)
Transport	`TsgcHTTP2Client` (HTTP/2 over TLS, OpenSSL or SChannel)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android
Edition	Enterprise

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`Client`	The `TsgcHTTP2Client` transport that carries the gRPC calls; set its Host, Port and TLS before making a call.
`ChannelOptions`	Channel-wide settings: gzip Compression, ContentType (proto/JSON), MaxMessageSize and MaxMetadataSize.
`DefaultMetadata`	Key/value metadata sent on every call, the usual place for an authorization token or a tracing header.
`Interceptors`	A chain of interceptors that wraps every call for cross-cutting logic such as logging or token refresh.
`RetryPolicy`	Automatic retry configuration: Enabled, MaxAttempts, backoff and the retryable status codes.
`ServiceConfig`	Per-service and per-method policy applied to calls.
`MetricsCollector`	Collects per-call counters and latencies for observability.

Main methods

The principal public methods exposed by the component.

`Call()`	Makes a synchronous unary call and returns a TsgcGRPCResponse with the status, message, data and trailers.
`CallAsync()`	Makes a non-blocking unary call; the reply arrives on OnGRPCResponse.
`CallString()`	Convenience unary call that sends and returns string payloads.
`ServerStreamingCall()`	Sends one request and receives a stream of messages through OnGRPCStreamMessage / OnGRPCStreamEnd.
`OpenClientStream()`	Opens a client-streaming call and returns the stream id.
`SendStreamMessage()`	Sends one message on an open client stream.
`CloseClientStream()`	Half-closes a client stream and returns the single server response.
`OpenBidiStream()`	Opens a bidirectional stream for full-duplex messaging and returns the stream id.
`SendBidiMessage()`	Sends one message on an open bidirectional stream.
`CloseBidiStream()`	Closes a bidirectional stream.
`CancelCall()`	Cancels an in-flight call or stream by its stream id.

Public events

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

`OnGRPCConnect`	Fires when the underlying HTTP/2 transport connects to the gRPC server.
`OnGRPCDisconnect`	Fires when the connection to the gRPC server is closed.
`OnGRPCResponse`	Fires when an asynchronous unary call (CallAsync) receives its response.
`OnGRPCStreamMessage`	Fires for each message received on a server-streaming or bidirectional call.
`OnGRPCStreamEnd`	Fires once a stream completes, carrying the final status.
`OnGRPCError`	Fires when a call returns a non-OK gRPC status code.
`OnGRPCException`	Fires when a transport or connection exception is raised on a call.
`OnGRPCBeforeRequest`	Fires just before a request is sent so headers or metadata can be adjusted.

Quick Start

Drop a **TsgcHTTP2Client** and a **TsgcGRPCClient** on a form, assign the transport, add any default metadata, then make a unary call. The request and response payloads are your serialized Protocol Buffers messages as `TBytes`.

> **About this scenario.** TsgcGRPCClient runs gRPC over the native HTTP/2 transport, so a gRPC channel is simply an HTTP/2 connection configured through the Client property.

Delphi (VCL / FireMonkey)

```
var
  HTTP2: TsgcHTTP2Client;
  GRPC: TsgcGRPCClient;
  oResponse: TsgcGRPCResponse;
begin
  HTTP2 := TsgcHTTP2Client.Create(nil);
  HTTP2.Host := 'grpc.example.com';
  HTTP2.Port := 443;
  HTTP2.TLS  := True;

  GRPC := TsgcGRPCClient.Create(nil);
  GRPC.Client := HTTP2;
  GRPC.DefaultMetadata.Add('authorization', 'Bearer eyJ...');

  oResponse := GRPC.Call('helloworld.Greeter', 'SayHello', RequestBytes);
  if oResponse.StatusCode = grpcOK then
    Memo1.Text := oResponse.DataString;
end;
```

C++ Builder

```
TsgcHTTP2Client *HTTP2 = new TsgcHTTP2Client(NULL);
HTTP2->Host = "grpc.example.com";
HTTP2->Port = 443;
HTTP2->TLS  = true;

TsgcGRPCClient *GRPC = new TsgcGRPCClient(NULL);
GRPC->Client = HTTP2;
GRPC->DefaultMetadata->Add("authorization", "Bearer eyJ...");

TsgcGRPCResponse *oResponse = GRPC->Call("helloworld.Greeter", "SayHello", RequestBytes);
if (oResponse->StatusCode == grpcOK)
  Memo1->Text = oResponse->DataString;
```

Common scenarios

Each scenario shows the configuration and method calls needed to drive the component through a specific gRPC flow.

1 · Asynchronous unary call

Use CallAsync to keep the UI responsive; the reply is delivered on the OnGRPCResponse event.

Delphi (VCL / FireMonkey)

```
GRPC.OnGRPCResponse := GRPCResponse;
GRPC.CallAsync('helloworld.Greeter', 'SayHello', RequestBytes);

procedure TForm1.GRPCResponse(Sender: TObject; const Response: TsgcGRPCResponse);
begin
  if Response.StatusCode = grpcOK then
    Memo1.Text := Response.DataString;
end;
```

2 · Server streaming

Send one request and receive a stream of messages; each message raises OnGRPCStreamMessage and OnGRPCStreamEnd fires at the close.

Delphi (VCL / FireMonkey)

```
GRPC.OnGRPCStreamMessage := GRPCStreamMessage;
GRPC.OnGRPCStreamEnd := GRPCStreamEnd;
GRPC.ServerStreamingCall('chat.Feed', 'Subscribe', RequestBytes);

procedure TForm1.GRPCStreamMessage(Sender: TObject; aStreamId: Integer;
  const aData: TBytes);
begin
  Memo1.Lines.Add(DecodeMessage(aData));
end;
```

3 · Client streaming

Open the stream, push each message, then close it to read the single server reply.

Delphi (VCL / FireMonkey)

```
var
  vStreamId: Integer;
  oResponse: TsgcGRPCResponse;
begin
  vStreamId := GRPC.OpenClientStream('upload.Service', 'Send');
  GRPC.SendStreamMessage(vStreamId, ChunkBytes1);
  GRPC.SendStreamMessage(vStreamId, ChunkBytes2);
  oResponse := GRPC.CloseClientStream(vStreamId);
  if oResponse.StatusCode = grpcOK then
    ShowMessage('Upload accepted');
end;
```

4 · Bidirectional streaming

Run a full-duplex exchange over a single HTTP/2 stream; both sides send whenever they like.

Delphi (VCL / FireMonkey)

```
var
  vStreamId: Integer;
begin
  vStreamId := GRPC.OpenBidiStream('chat.Room', 'Connect');
  GRPC.SendBidiMessage(vStreamId, EncodeMessage('hello'));
  // incoming messages arrive on OnGRPCStreamMessage
  GRPC.CloseBidiStream(vStreamId);
end;
```

5 · Metadata and channel options

Configure authorization metadata, gzip compression and message limits for the channel.

Delphi (VCL / FireMonkey)

```
GRPC.DefaultMetadata.Add('authorization', 'Bearer eyJ...');
GRPC.ChannelOptions.Compression := grpcGzip;
GRPC.ChannelOptions.ContentType := grpcProto;
GRPC.ChannelOptions.MaxMessageSize := 16 * 1024 * 1024;
```

6 · Automatic retry

Enable RetryPolicy and list the status codes that should trigger a retry, typically transient grpcUNAVAILABLE failures.

Delphi (VCL / FireMonkey)

```
GRPC.RetryPolicy.Enabled := True;
GRPC.RetryPolicy.MaxAttempts := 4;
GRPC.RetryPolicy.InitialBackoff := 200;
GRPC.RetryPolicy.BackoffMultiplier := 2.0;
GRPC.RetryPolicy.RetryableStatusCodes :=
  GRPC.RetryPolicy.RetryableStatusCodes + [grpcUNAVAILABLE, grpcRESOURCE_EXHAUSTED];
```

Resources & references

gRPC over HTTP/2 protocol	github.com/grpc/grpc/blob/master/doc/PROTOCOL-HTTP2.md
HTTP/2 — RFC 9113	datatracker.ietf.org/doc/html/rfc9113
Protocol Buffers	protobuf.dev
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/HTTP/GRPC/Client/TsgcGRPCClient.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\21.GRPC\01.GRPC_Client`
Component page	www.esegece.com/products/websockets/http/grpc-client/
Product page	www.esegece.com/products/websockets/

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