SGCWEBSOCKETS · TECHNICAL DOCUMENT

Apple Push Notification Service

APNs HTTP/2 client — send iOS and macOS push notifications from Delphi using token-based or certificate-based authentication.

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcHTTP2Client implements Client HTTP/2 Component and can connect to HTTP/2 servers.

At a glance

COMPONENT CLASS

`TsgcHTTP2Client`

STANDARDS / SPEC

Sending notification requests to APNs

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	Sending notification requests to APNs · HTTP/2 — RFC 9113
Component class	`TsgcHTTP2Client` (unit `sgcHTTP2_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.

`Authentication`	Configures the credentials used to authenticate HTTP/2 requests, including OAuth2 and JWT tokens.
`TLSOptions`	Configures certificates, TLS version, ALPN, IOHandler and other secure-connection details used for HTTP/2 over TLS.
`Active`	Opens or closes the HTTP/2 connection to the remote server.
`Host`	IP address or DNS name of the HTTP/2 server the client will connect to.
`Port`	TCP port used to connect to the HTTP/2 server.
`TLS`	Enables a secure TLS connection, which is normally required by HTTP/2 servers.
`Proxy`	Routes the HTTP/2 connection through an HTTP CONNECT tunnel or SOCKS proxy server.
`HeartBeat`	Sends periodic HTTP/2 PING frames to keep the connection alive.
`WatchDog`	Automatically reconnects to the HTTP/2 server after an unexpected disconnection.
`Throttle`	Limits the number of bits per second sent or received by the HTTP/2 socket.

Main methods

The principal public methods exposed by the component.

`ConnectAsync()`	Opens the HTTP/2 connection and issues a non-blocking GET; the reply is delivered on OnHTTP2Response.
`Get()`	Sends a synchronous HTTP/2 GET request and returns the response body.
`PostAsync()`	Sends a non-blocking POST; the reply arrives on OnHTTP2Response.
`PutAsync()`	Sends a non-blocking PUT; the reply arrives on OnHTTP2Response.
`DeleteAsync()`	Sends a non-blocking DELETE; the reply arrives on OnHTTP2Response.
`PatchAsync()`	Sends a non-blocking PATCH; the reply arrives on OnHTTP2Response.
`Connect()`	Opens the HTTP/2 connection and issues a synchronous GET to the supplied URL.
`Disconnect()`	Closes the underlying TCP/TLS socket immediately without sending a GOAWAY frame.
`Close()`	Performs a graceful shutdown by sending a GOAWAY frame with an error code and optional debug text.
`Ping()`	Sends an HTTP/2 PING frame to probe liveness and measure round-trip time.

Public events

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

`OnHTTP2Authorization`	Fires when the server requires authentication so the application can supply credentials or a bearer token.
`OnHTTP2BeforeRequest`	Fires just before request headers are sent so the application can add or modify them.
`OnHTTP2Connect`	Fires just after the client connects successfully to the HTTP/2 server.
`OnHTTP2Disconnect`	property OnHTTP2Disconnect: TsgcHTTP2ClientDisconnectEvent; // TsgcHTTP2ClientDisconnectEvent = procedure(Sender: TObject; const Connection: TsgcHTTP2ConnectionClient) of object __property TsgcHTTP2Cl...

`OnHTTP2Exception`	Fires when an exception is raised on the HTTP/2 connection so the application can handle it.
`OnHTTP2GoAway`	Fires when the server sends a GoAway frame signalling the connection is being shut down.
`OnHTTP2PendingRequests`	Fires after a disconnection when there are pending requests so the application can reconnect or clear the queue.
`OnHTTP2PushPromise`	Fires when the server pushes a resource so the client can accept or cancel it.
`OnHTTP2RSTStream`	property OnHTTP2RSTStream: TsgcHTTP2ClientRSTStreamEvent; // TsgcHTTP2ClientRSTStreamEvent = procedure(Sender: TObject; const Connection: TsgcHTTP2ConnectionClient; const RSTStream: TsgcHTTP2RSTStream...
`OnHTTP2Response`	Fires when the client receives the full response (status, headers and body) from the server.
`OnHTTP2ResponseFragment`	Fires for each streamed response fragment when FragmentedData delivers data as it arrives.
`OnSSLAfterCreateHandler`	Fires after the SSL handler has been created so its properties can be customized.
`OnSSLGetHandler`	Fires before the SSL handler is created so a custom handler instance can be supplied.

Quick Start

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

> **About this scenario.** Configure the JWT Client with the following values:

Delphi (VCL / FireMonkey)

```
oHTTP := TsgcHTTP2Client.Create(nil);
oHTTP.TLSOptions.IOHandler := iohOpenSSL;
<br/>

oJWT := TsgcHTTP_JWT_Client.Create(nil);
oHTTP.Authentication.Token.JWT := oJWT;
oJWT.JWTOptions.Header.alg := jwtES256;
oJWT.JWTOptions.Header.kid := 'apple key id';
oJWT.JWTOptions.Payload.iss := 'issuer';
oJWT.JWTOptions.Payload.iat := StrToInt64(GetDateTimeUnix(Now, False));
oJWT.JWTOptions.Algorithms.ES.PrivateKey.LoadFromFile('AuthKey_**.p8');
oJWT.JWTOptions.RefreshTokenAfter := 60*40;
<br/>

oHTTP.Request.CustomHeaders.Clear;
oHTTP.Request.CustomHeaders.Add('apns-topic: com.example.application');
```

C++ Builder

```
TsgcHTTP2Client *oHTTP = new TsgcHTTP2Client(NULL);
oHTTP->TLSOptions->IOHandler = iohOpenSSL;
<br/>

TsgcHTTP_JWT_Client *oJWT = new TsgcHTTP_JWT_Client(NULL);
oHTTP->Authentication->Token->JWT = oJWT;
oJWT->JWTOptions->Header->alg = jwtES256;
oJWT->JWTOptions->Header->kid = "apple key id";
oJWT->JWTOptions->Payload->iss = "issuer";
oJWT->JWTOptions->Payload->iat = StrToInt64(GetDateTimeUnix(Now, False));
oJWT->JWTOptions->Algorithms->ES->PrivateKey->LoadFromFile("AuthKey_**.p8");
oJWT->JWTOptions->RefreshTokenAfter = 60*40;
<br/>

oHTTP->Request->CustomHeaders->Clear();
oHTTP->Request->CustomHeaders->Add("apns-topic: com.example.application");
```

.NET (C#)

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
oHTTP.TLSOptions.IOHandler = TwsTLSIOHandler.iohOpenSSL;
<br/>

TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();
oHTTP.Authentication.Token.JWT = oJWT;
oJWT.JWTOptions.Header.alg = TsgcHTTP_JWT_Algorithm.jwtES256;
oJWT.JWTOptions.Header.kid = "apple key id";
oJWT.JWTOptions.Payload.iss = "issuer";
oJWT.JWTOptions.Payload.iat = unix time;
oJWT.JWTOptions.Algorithms.ES.PrivateKey.LoadFromFile("AuthKey_**.p8");
oJWT.JWTOptions.RefreshTokenAfter = 60*40;
<br/>

oHTTP.Request.CustomHeaders = "apns-topic: com.example.application";
```

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 · OpenSSL

If you use OpenSSL, you must deploy the OpenSSL libraries with your application. Before setting the certificate with the TsgcHTTP2Client, this certificate must first be converted to PEM format because OpenSSL doesn't allow importing P12 certificates directly.

Delphi (VCL / FireMonkey)

```
oHTTP := TsgcHTTP2Client.Create(nil);
oHTTP.TLSOptions.IOHandler := iohOpenSSL;
oHTTP.TLSOptions.CertFile := 'certificate_file.pem';
oHTTP.TLSOptions.KeyFile := 'private_key.pem';
oHTTP.TLSOptions.Password := 'certificate password';
oHTTP.TLSOptions.Version := tls1_2;
```

C++ Builder

```
TsgcHTTP2Client *oHTTP = new TsgcHTTP2Client(NULL);
oHTTP->TLSOptions->IOHandler = iohOpenSSL;
oHTTP->TLSOptions->CertFile = "certificate_file.pem";
oHTTP->TLSOptions->KeyFile = "private_key.pem";
oHTTP->TLSOptions->Password = "certificate password";
oHTTP->TLSOptions->Version = tls1_2;
```

.NET (C#)

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
oHTTP.TLSOptions.IOHandler = TwsTLSIOHandler.iohOpenSSL;
oHTTP.TLSOptions.CertFile = "certificate_file.pem";
oHTTP.TLSOptions.KeyFile = "private_key.pem";
oHTTP.TLSOptions.Password = "certificate password";
oHTTP.TLSOptions.Version = TwsTLSVersions.tls1_2;
```

2 · Sample Code

Create a new instance of TsgcHTTP2Client and call the method POST to send a notification to APNs.

Delphi (VCL / FireMonkey)

```
oHTTP := TsgcHTTP2Client.Create(nil);
Try
  // ... requires authorization code
  oStream := TStringStream.Create('{"aps":{"alert":"Alert from sgcWebSockets!"}}',
    TEncoding.UTF8);
  Try
    oHTTP.Post('https://api.push.apple/3/device/device_token', oStream);
    if oHTTP.Response.Status = 200 then
      ShowMessage('Notification Sent Successfully')
    else
      ShowMessage('Notification error');
  Finally
    oStream.Free;
  End;
Finally
  oHTTP.Free;
End;
```

C++ Builder

```
TsgcHTTP2Client *oHTTP = new TsgcHTTP2Client();
try
{
  // ... requires authorization code
  TStringStream *oStream = new TStringStream("{\"aps\":{\"alert\":\"Alert from sgcWebSockets!\"}
    TEncoding::UTF8);
  try
  {
    oHTTP->Post("https://api.push.apple/3/device/device_token", oStream);
    if (oHTTP->Response->Status == 200)
    {
      ShowMessage("Notification Sent Successfully");
    }
    else
    {
      ShowMessage("Notification error");
    }
  }
  __finally
  {
    oHTTP->Free();
  }
}
__finally
{
  oHTTP->Free();
}
```

.NET (C#)

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
// ... requires authorization code
oHTTP.Post("https://api.push.apple/3/device/device_token", "{\"aps\":{\"alert\":\"Alert from sgc
if (oHTTP.Response.Status == 200)
{
  MessageBox.Show("Notification Sent Successfully");
}
else
{
  MessageBox.Show("Notification error");
}
```

3 · Errors

If you get the error "missing topic" most probably you are using an universal certificate (certificates that can be used for push notifications, voip...) which requires to set the topic name with the value of your

app's bundle ID/app id (example: com.example.application). Just set the apns-topic header with the correct value in the Request property of the HTTP/2 client.

Delphi (VCL / FireMonkey)

```
oHTTP.Request.CustomHeaders.Clear;
oHTTP.Request.CustomHeaders.Add('apns-topic: com.example.application');
```

C++ Builder

```
oHTTP->Request->CustomHeaders->Clear();
oHTTP->Request->CustomHeaders->Add("apns-topic: com.example.application");
```

.NET (C#)

```
oHTTP.Request.CustomHeaders = "apns-topic: com.example.application";
```

4 · SChannel

If you use SChannel there is no need to deploy any libraries and the certificate downloaded from Apple can be directly imported without the need of a previous conversion to PEM format.

Delphi (VCL / FireMonkey)

```
oHTTP := TsgcHTTP2Client.Create(nil);
oHTTP.TLSOptions.IOHandler := iohSChannel;
oHTTP.TLSOptions.SChannel_Options.UseLegacyCredentials := true;
oHTTP.TLSOptions.CertFile := 'certificate_file.p12';
oHTTP.TLSOptions.Password := 'certificate password';
oHTTP.TLSOptions.Version := tls1_2;
```

C++ Builder

```
TsgcHTTP2Client *oHTTP = new TsgcHTTP2Client(NULL);
oHTTP->TLSOptions->IOHandler = iohSChannel;
oHTTP->TLSOptions->SChannel_Options->UseLegacyCredentials = true;
oHTTP->TLSOptions->CertFile = "certificate_file.p12";
oHTTP->TLSOptions->Password = "certificate password";
oHTTP->TLSOptions->Version = tls1_2;
```

.NET (C#)

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
oHTTP.TLSOptions.IOHandler = TwsTLSIOHandler.iohSChannel;
oHTTP.TLSOptions.IOHandler.SChannel_Options.UseLegacyCredentials = true;
oHTTP.TLSOptions.CertFile = "certificate_file.p12";
oHTTP.TLSOptions.Password = "certificate password";
oHTTP.TLSOptions.Version = TwsTLSVersions.tls1_2;
```

5 · Register your APP with APNs

https://developer.apple.com/documentation/usernotifications/registering_your_app_with_apns

Delphi (VCL / FireMonkey)

```
var
oPushService: TPushService;
oPushConnection: TPushServiceConnection;
begin
oPushService := TPushServiceManager.Instance.GetServiceByName(TPushService.TServiceNames.APS);
oPushConnection := TPushServiceConnection.Create(oPushService);
oPushConnection.Active := True;
oPushConnection.OnChange := OnChangeEvent;
oPushConnection.OnReceiveNotification := OnReceiveNotificationEvent;
vDeviceId := oPushService.DeviceIDValue[TPushService.TDeviceIDNames.DeviceID];
vDeviceToken := oPushService.DeviceTokenValue[TPushService.TDeviceTokenNames.DeviceToken];
end;
<br/>

procedure OnChangeEvent(Sender: TObject; AChange: TPushService.TChanges);
begin
memoLog.Lines.Add('OnChange');
end;
<br/>

procedure OnReceiveNotificationEvent(Sender: TObject; const ANotification: TPushServiceNotificat
begin
memoLog.Lines.Add('DataKey=' + ANotification.DataKey);
memoLog.Lines.Add('JSON=' + ANotification.JSON.ToString);
memoLog.Lines.Add('DataObject=' + ANotification.DataObject.ToString);
end;
```

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 — Sending notification requests to APNs	developer.apple.com/documentation/usernotifications/sending-notification-requests-to-apns
Primary standard / spec — HTTP/2 — RFC 9113	datatracker.ietf.org/doc/html/rfc9113
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/HTTP/HTTP2/Client/TsgcHTTP2Client.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\20.HTTP_Protocol\07.Apple_Push_Notifications`
Component page	www.esegece.com/products/websockets/http/apple-push/
Product page	www.esegece.com/products/websockets/

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