SGCWEBSOCKETS · TECHNICAL DOCUMENT

WebAuthn Server

TsgcWSAPIServer_WebAuthn — W3C WebAuthn Level 2 server-side relying-party for sgcWebSockets HTTP servers.

eSeGeCe

WWW.ESEGECE.COM

Overview

The TsgcWSAPIServer_WebAuthn component provides a simple but powerful solution to implement the WebAuthn Relying Party server, enabling passwordless authentication in your web application. A WebAuthn application consists of a WebAuthn server that handles the server-side registration and authentication and a client-side application that usually is a javascript application.

At a glance

COMPONENT CLASS

`TsgcWSAPIServer_WebAuthn`

STANDARDS / SPEC

Web Authentication Level 2 — W3C

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	Web Authentication Level 2 — W3C · FIDO Alliance — WebAuthn overview
Component class	`TsgcWSAPIServer_WebAuthn` (unit `sgcWebSocket_API_WebAuthn_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.

`EndpointsOptions`	URL routes served by the component for WebAuthn registration, authentication, the bundled JavaScript library and the test page.
`Server`	HTTP server component the WebAuthn API is attached to (TsgcWebSocketHTTPServer or TsgcWebSocketServer_HTTPAPI).
`WebAuthnOptions`	Relying Party configuration: RPName, RPID, Origins, supported Algorithms, Attestation formats, User Verification, Timeout and AuthenticatorSelection.
`Version`	Read-only string with the sgcWebSockets library version.

Main methods

The principal public methods exposed by the component.

`ValidateAuthenticationOptions()`	Parses the authenticator data, client data and signature and authenticates the user against the stored credential.
`IsWebAuthnTokenValid()`	Checks whether the WebAuthn bearer token supplied by the client is still valid for the given connection.
`IsWebAuthnUnauthorized()`	Returns True when the connection is not authorized and a 401 response must be sent.

`IsWebAuthnRequest()`	Returns True when the incoming URL path matches one of the configured WebAuthn endpoints.
`ValidateRegistrationOptions()`	Parses the client attestation payload and validates a new credential registration per the WebAuthn specification.
`AddCredential()`	Registers an existing credential (public key, credential id and user) in the server credential store.
`KeepAlive()`	Sends a keep-alive ping on the given connection to prevent the authenticated session from expiring.
`DoProcessHTTP()`	Entry point of the HTTP pipeline; dispatches registration, authentication and authorization requests.

Public events

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

`OnWebAuthnAuthenticationError`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnAuthenticationError
`OnWebAuthnAuthenticationOptionsRequest`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnAuthenticationOptionsRequest
`OnWebAuthnAuthenticationOptionsResponse`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnAuthenticationOptionsResponse
`OnWebAuthnAuthenticationSuccessful`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnAuthenticationSuccessful
`OnWebAuthnException`	Fires when an unhandled exception is raised while processing a WebAuthn request; lets the application log the error and override the HTTP response code.
`OnWebAuthnHTTPRequest`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnHTTPRequest
`OnWebAuthnHTTPResponse`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnHTTPResponse

`OnWebAuthnMetadata`	Fires when the server needs authenticator metadata for an AAGUID; lets the application return a cached or custom FIDO MDS BLOB entry.
`OnWebAuthnRegistrationError`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationError
`OnWebAuthnRegistrationOptionsRequest`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationOptionsRequest
`OnWebAuthnRegistrationOptionsResponse`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationOptionsResponse
`OnWebAuthnRegistrationSuccessful`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationSuccessful
`OnWebAuthnRegistrationValidateCertificate`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationValidateCertificate
`OnWebAuthnRegistrationValidateCredentialId`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnRegistrationValidateCredentialId
`OnWebAuthnUnauthorized`	TsgcWSAPIServer_WebAuthn › Events › OnWebAuthnUnauthorized

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **WebAuthn Authentication | Request** configuration sourced from the online help.

> **About this scenario.** When a user attempts to log in, the browser sends a request to the server asking for the authentication options (also called "assertion options").

Delphi (VCL / FireMonkey)

```
procedure OnWebAuthnAuthenticationOptionsRequest(
    Sender: TObject; const aRequest:
    TsgcWebAuthn_AuthenticationOptions_Request; var CredentialRecords:
    TsgcWebAuthn_CredentialRecords; var Accept: Boolean);
begin
  if UserExistsInDB(aRequest.Username) then
  begin
    While not EOF do
    begin
      CredentialRecords.AddCredentialRecordFromJSON(RecordFromDB);
      Next;
    end;
  end;
end;
```

C++ Builder

```
void __fastcall TForm1::OnWebAuthnAuthenticationOptionsRequest(
  TObject *Sender,
  TsgcWebAuthn_AuthenticationOptions_Request *aRequest,
  TsgcWebAuthn_CredentialRecords &CredentialRecords,
  bool &Accept)
{
    if (UserExistsInDB(aRequest->Username)) {
        while (!EOF()) {
            CredentialRecords.AddCredentialRecordFromJSON(RecordFromDB());
            Next();
        }
    }
}
```

.NET (C#)

```
public void OnWebAuthnAuthenticationOptionsRequest(
    object sender,
    TsgcWebAuthn_AuthenticationOptions_Request aRequest,
    ref TsgcWebAuthn_CredentialRecords credentialRecords,
    ref bool accept)
{
    if (UserExistsInDB(aRequest.Username))
    {
        while (!EOF())
        {
            credentialRecords.AddCredentialRecordFromJSON(RecordFromDB());
            Next();
        }
    }
}
```

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 · WebAuthn Authorization | WebSocket

Once you have a new token, just send an authorization header with this bearer token. You can use the event OnHandShake to add the Bearer Token to the connection request. Example:

Delphi (VCL / FireMonkey)

```
procedure OnClientHandshake(Connection: TsgcWSConnection; var Headers: TStringList);
begin
        Headers.Add('Authorization: Bearer C760C1C39E3D4E829693A13F18F5CFDE537B516336FC48F7BAB02
end;
```

C++ Builder

```
void __fastcall OnClientHandshake(TsgcWSConnection *Connection, TStringList *Headers)
{
    Headers->Add("Authorization: Bearer C760C1C39E3D4E829693A13F18F5CFDE537B516336FC48F7BAB02761
}
```

.NET (C#)

```
public static async Task<ClientWebSocket> ConnectWithAuthAsync(Uri serverUri)
{
    var client = new ClientWebSocket();
    // Add custom Authorization header
    client.Options.SetRequestHeader("Authorization", "Bearer C760C1C39E3D4E829693A13F18F5CFDE537
    // Connect to WebSocket server
    await client.ConnectAsync(serverUri, CancellationToken.None);
    return client;
}
```

2 · Registration Successful

If the response sent by the client is valid, the event OnWebAuthnRegistrationSuccessful is called and the Credential Record can be safely stored into a database for future logins validations.

Delphi (VCL / FireMonkey)

```
procedure OnWebAuthnRegistrationSuccessful(Sender: TObject; const aRegistration: TsgcWebAuthn_Re
 const aCredentialRecord: TsgcWebAuthn_CredentialRecord; var Accept: Boolean);
begin
  // store in a db
  DB.Credentials.Append;
  DB.Credentials.FieldByName('Credentials').AsString := aCredentialRecord.AsJSON;
  DB.Credentials.Post;
end;
```

C++ Builder

```
void __fastcall OnWebAuthnRegistrationSuccessful(TObject *Sender,
  TsgcWebAuthn_Registration* aRegistration,
  TsgcWebAuthn_CredentialRecord* aCredentialRecord,
  bool &Accept)
{
  // Store in a DB
  DB->Credentials->Append();
  DB->Credentials->FieldByName("Credentials")->AsString = aCredentialRecord->AsJSON;
  DB->Credentials->Post();
}
```

.NET (C#)

```
void OnWebAuthnRegistrationSuccessful(
    object sender,
    WebAuthnRegistration registration,
    WebAuthnCredentialRecord credentialRecord,
    ref bool accept)
{
    // Store in a database
    using (var cmd = dbConnection.CreateCommand())
    {
        cmd.CommandText = "INSERT INTO Credentials (Credentials) VALUES (@json)";
        var param = cmd.CreateParameter();
        param.ParameterName = "@json";
        param.Value = credentialRecord.AsJson(); // or .AsJSON depending on naming
        cmd.Parameters.Add(param);
        cmd.ExecuteNonQuery();
    }
}
```

3 · WebAuthn Authorization | HTTP

Once you have a new token, just send an authorization header with this bearer token. You can use the CustomHeaders property of the TsgcHTTP1Client to configure the Bearer Token. Example:

```
Delphi (VCL / FireMonkey)
```

```
procedure GetHTTPRequest(const aURL: string; const aToken: string): string;
var
  oHTTP: TsgcHTTP1Client;
begin
  oHTTP := TsgcHTTP1Client.Create(nil);
  Try
    oHTTP.Request.CustomHeaders.AddValue('Authorization', 'Bearer ' + aToken);
    result := oHTTP.Get(aURL);
  Finally
    oHTTP.Free;
  End;
end;
```

```
C++ Builder
```

```
String GetHTTPRequest(const String aURL, const String aToken)
{
    // Create the HTTP client dynamically
    std::unique_ptr<TsgcHTTP1Client> oHTTP(new TsgcHTTP1Client(nullptr));
    // Add Authorization header
    oHTTP->Request->CustomHeaders->AddValue("Authorization", "Bearer " + aToken);
    // Perform GET request and return result
    return oHTTP->Get(aURL);
}
```

.NET (C#)

```
public static async Task<string> GetHTTPRequestAsync(string aURL, string aToken)
{
    using (var client = new HttpClient())
    {
        // Add Authorization header
        client.DefaultRequestHeaders.Add("Authorization", "Bearer " + aToken);
        // Send GET request
        HttpResponseMessage response = await client.GetAsync(aURL);
        // Throw exception if request failed
        response.EnsureSuccessStatusCode();
        // Return the response content as string
        return await response.Content.ReadAsStringAsync();
    }
}
```

4 · WebAuthn Registration | Request

This registration options request is essential to bootstrap WebAuthn registration securely. It asks the server to:

Delphi (VCL / FireMonkey)

```
procedure OnWebAuthnRegistrationOptionsRequest(Sender: TObject; const aRequest: TsgcWebAuthn_Reg
begin
  if aRequest.Username = 'anonymous' then
    Accept := False;
end;
```

C++ Builder

```
void __fastcall TForm1::OnWebAuthnRegistrationOptionsRequest(TObject *Sender, TsgcWebAuthn_Regis
{
    if (aRequest->Username == "anonymous")
        Accept = false;
}
```

.NET (C#)

```
public void OnWebAuthnRegistrationOptionsRequest(object sender, TsgcWebAuthn_RegistrationOptions
{
    if (aRequest.Username == "anonymous")
        accept = false;
}
```

5 · WebAuthn Registration | Response

The server responds to the client's registration options request (e.g., POST /sgcWebAuthn/Registration/Options) with a JSON payload that looks like the following (after base64url-encoding binary fields):

Delphi (VCL / FireMonkey)

```
procedure OnWebAuthnRegistrationOptionsResponse(Sender: TObject; const aRequest: TsgcWebAuthn_Re
begin
  if aRequest.Username = 'esegece.com' then
  begin
    aResponse.ExcludeCredentials.AddCredentialRecordFromJSON('json1.txt');
    aResponse.ExcludeCredentials.AddCredentialRecordFromJSON('json2.txt');
  end;
end;
```

C++ Builder

```
void __fastcall OnWebAuthnRegistrationOptionsResponse(TObject *Sender,
  TsgcWebAuthn_RegistrationOptions_Request* aRequest,
  TsgcWebAuthn_RegistrationOptions_Response* aResponse)
{
  if (aRequest->Username == "esegece.com") {
    aResponse->ExcludeCredentials->AddCredentialRecordFromJSON("json1.txt");
    aResponse->ExcludeCredentials->AddCredentialRecordFromJSON("json2.txt");
  }
}
```

.NET (C#)

```
void OnWebAuthnRegistrationOptionsResponse(object sender,
    WebAuthnRegistrationOptionsRequest request,
    WebAuthnRegistrationOptionsResponse response)
{
    if (request.Username == "esegece.com")
    {
        response.ExcludeCredentials.AddCredentialRecordFromJSON("json1.txt");
        response.ExcludeCredentials.AddCredentialRecordFromJSON("json2.txt");
    }
}
```

6 · Registration Error

If there is any error while validating the client response, the event OnWebAuthnRegistrationError is called and you can access the reason for the error in the parameter aError.

Delphi (VCL / FireMonkey)

```
procedure OnWebAuthnRegistrationError(Sender:
    TObject; const aRequest: TsgcWebAuthn_RegistrationVerify_Request; const
    aRegistration: TsgcWebAuthn_Registration; const aError: string);
begin
  Log('#webauthn_registration_error: ' + aError);
end;
```

C++ Builder

```
void __fastcall OnWebAuthnRegistrationError(TObject *Sender,
  TsgcWebAuthn_RegistrationVerify_Request* aRequest,
  TsgcWebAuthn_Registration* aRegistration,
  const String aError)
{
  Log("#webauthn_registration_error: " + aError);
}
```

.NET (C#)

```
void OnWebAuthnRegistrationError(
    object sender,
    WebAuthnRegistrationVerifyRequest request,
    WebAuthnRegistration registration,
    string error)
{
    Log("#webauthn_registration_error: " + error);
}
```

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 — Web Authentication Level 2 — W3C	www.w3.org/TR/webauthn-2/
Primary standard / spec — FIDO Alliance — WebAuthn overview	fidoalliance.org/specs/fido-v2.2-rd-20230321/fido-client-to-authenticator-protocol-v2.2-rd-20230321.html
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/HTTP/Authorization/WebAuthn/server/TsgcWSAPIServer_WebAuthn.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\20.HTTP_Protocol\12.WebAuthn`
Component page	www.esegece.com/products/websockets/http/webauthn/
Product page	www.esegece.com/products/websockets/

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