SGCWEBSOCKETS · TECHNICAL DOCUMENT

API Key Manager

Centralised API-key rotation, scoping and rate-limit accounting for sgcWebSockets API and HTTP components.

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcWSAPIKeyManager provides full-lifecycle management for API keys issued by sgcWebSockets servers.

At a glance

COMPONENT CLASS `TsgcWSAPIKeyManager`	STANDARDS / SPEC —
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

Component class	`TsgcWSAPIKeyManager` (unit `sgcWebSocket_APIKeyManager`)
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.

`Enabled`	Master switch. When False, ValidateKey and IsRequestAuthorized always return True and server auto-hooks become no-ops.
`Generation`	Controls how plaintext keys are built by IssueKey: KeyPrefix, KeyLength, Charset and optional checksum.
`Hashing`	At-rest hash algorithm (SHA-256, SHA-512 or Bcrypt) with optional Salt and Iterations for key stretching.
`Storage`	Controls where hashed keys and audit log are kept: in memory, in an optionally encrypted file, or in user hooks.
`Scopes`	Catalog of allowed scope strings enforced when keys are issued or checked for a required scope.
`Validation`	How keys are extracted (header, query), transport rules (HTTPS, IP allowlist) and the FailClosed policy.
`Expiration`	Default TTL, enforcement flag and background sweep interval controlling how issued keys age out.
`Rotation`	Grace-period and auto-rotate settings that let a new key replace an old one without downtime.
`RateLimit`	Per-key rate-limit metadata (max requests per window) consumed by TsgcWSRateLimiter.PerAPIKey.
`Audit`	Ring-buffer and file-backed audit log of every key-lifecycle action with configurable retention.

Main methods

The principal public methods exposed by the component.

`IsRequestAuthorized()`	One-shot authorization check: extracts the key from headers or query string and validates it.
`ExtractKeyFromQuery()`	Parses the Validation.QueryParamName value out of a raw query string.
`IsConnectionAllowed()`	Server hook enforcing the optional IPAllowlist when FailClosed is True.
`RegisterConnection()`	Tracks a new connection. Called automatically by the server.
`UnregisterConnection()`	Releases tracking for a connection. Called automatically on disconnect.
`ValidateKey()`	Validates a raw key and optionally enforces a required scope and records the requester's IP.
`ListScopes()`	Returns the scopes attached to a key.
`IssueKey()`	Generates, hashes and stores a new key; returns the plaintext (only time it can be observed).
`RotateKey()`	Issues a fresh key for the same owner and scopes and marks the old one kksRotated.
`RenewKey()`	Extends the key's expiration by the given number of seconds from now.

Public events

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

`OnAuditEvent`	Fired for every audit entry; carries the full TsgcAPIKeyAuditEntry for SIEM forwarding.
`OnKeyExpired`	Fired by the background sweep when a key has expired (or NotifyBeforeExpirySec earlier).
`OnKeyIssued`	Fired when a new key has been issued; carries the owner, plaintext key and scopes.
`OnKeyRevoked`	Fired when a key has been revoked; carries the key and revocation reason.
`OnKeyRotated`	Fired when a key has been rotated; carries both the old and the new key.

`OnKeyValidated`	Fired every time ValidateKey completes; carries the key, an aValid flag and the reason.
`OnValidation`	Final decision hook fired during ValidateKey; set Allow := False to reject a valid key.

Quick Start

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

> **About this scenario.** Server hook enforcing the optional IPAllowlist when FailClosed is True.

Delphi (VCL / FireMonkey)

```
// Custom server loop bridging the manager
if not sgcWSAPIKeyManager1.IsConnectionAllowed(oPeer.IP) then
  oPeer.Disconnect;
```

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 · RegisterConnection

Tracks a new connection. Called automatically by the server.

Delphi (VCL / FireMonkey)

```
// Called automatically by TsgcWebSocketHTTPServer. Custom loops:
sgcWSAPIKeyManager1.RegisterConnection(oPeer.IP);
```

2 · UnregisterConnection

Releases tracking for a connection. Called automatically on disconnect.

Delphi (VCL / FireMonkey)

```
// Called automatically by the server on disconnect. Custom loops:
sgcWSAPIKeyManager1.UnregisterConnection(oPeer.IP);
```

3 · Audit

Ring-buffer and file-backed audit log of every key-lifecycle action with configurable retention.

Delphi (VCL / FireMonkey)

```
// Compliance: 12-month retention, log to file, include IP + payload
sgcWSAPIKeyManager1.Audit.Enabled := True;
sgcWSAPIKeyManager1.Audit.LogFile := 'apikeys-audit.log';
sgcWSAPIKeyManager1.Audit.IncludeIP := True;
sgcWSAPIKeyManager1.Audit.IncludePayload := True;
sgcWSAPIKeyManager1.Audit.RetentionDays := 365;
sgcWSAPIKeyManager1.Audit.MaxMemoryEntries := 50000;
```

4 · ClearAuditLog

Clears the audit log for a specific key or (when empty) for all keys.

Delphi (VCL / FireMonkey)

```
// GDPR erasure for a single customer key
sgcWSAPIKeyManager1.ClearAuditLog(vKey);

// Wipe the entire audit log
sgcWSAPIKeyManager1.ClearAuditLog;
```

5 · Count

Returns the number of keys currently stored.

Delphi (VCL / FireMonkey)

```
LabelTotal.Caption := Format('Total keys in store: %d', [sgcWSAPIKeyManager1.Count]);
```

6 · Enabled

Master switch. When False, ValidateKey and IsRequestAuthorized always return True and server auto-hooks become no-ops.

Delphi (VCL / FireMonkey)

```
// Temporarily bypass API key enforcement during a maintenance window
sgcWSAPIKeyManager1.Enabled := False;
try
  RunMaintenanceTask;
finally
  sgcWSAPIKeyManager1.Enabled := True;
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.

Online help — component page	www.esegece.com/help/sgcWebSockets/Components/TsgcWSAPIKeyManager.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\04.WebSocket_Other_Samples\16.APIKeyManager`
Component page	www.esegece.com/products/websockets/components/resilience/api-key-manager/
Product page	www.esegece.com/products/websockets/

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