



# sgcWebSockets 2026.6

June 2026

Dokumentacja dla CBuilder

**Copyright © 2012-2026 eSeGeCe Software**  
**info@esegece.com**  
**[www.esegece.com](http://www.esegece.com)**

# Spis treści

|                                             |     |
|---------------------------------------------|-----|
| Wprowadzenie .....                          | 14  |
| Przegląd.....                               | 19  |
| Edycje .....                                | 19  |
| Instalacja .....                            | 21  |
| Instalacja .....                            | 27  |
| Install Setup .....                         | 27  |
| Instalacja pakietu .....                    | 35  |
| Błędy instalacji.....                       | 38  |
| Konfiguracja instalacji .....               | 43  |
| Instalacja pakietu sgcIndy .....            | 46  |
| Konfiguruj ZLib.....                        | 49  |
| Szybki start.....                           | 50  |
| Przegląd.....                               | 50  |
| QuickStart WebSockets.....                  | 53  |
| QuickStart HTTP .....                       | 56  |
| Przepływ wątków .....                       | 58  |
| Kompilacja.....                             | 60  |
| Kompilacja aplikacji OSX .....              | 61  |
| Kompilacja aplikacji Android .....          | 63  |
| Kompilowanie aplikacji iOS.....             | 64  |
| Serwer o wysokiej wydajności.....           | 66  |
| Menedżer pamięci .....                      | 71  |
| OpenSSL .....                               | 74  |
| OpenSSL Windows .....                       | 78  |
| OpenSSL OSX .....                           | 81  |
| OpenSSL Android .....                       | 83  |
| OpenSSL iOS .....                           | 85  |
| OpenSSL Own CA Certificates.....            | 88  |
| Certyfikaty P12 OpenSSL.....                | 90  |
| OpenSSL Weryfikacja certyfikatu .....       | 92  |
| Ładowanie dodatkowych funkcji OpenSSL ..... | 93  |
| Indy .....                                  | 95  |
| Tematy.....                                 | 98  |
| Zdarzenia WebSocket.....                    | 98  |
| Parametry połączenia WebSocket .....        | 100 |
| Użycie wewnątrz biblioteki DLL.....         | 101 |
| Usługa Windows .....                        | 102 |
| Test przeglądarki internetowej .....        | 103 |
| Niestandardowe subprotokołu .....           | 104 |
| Uwierzytelnianie .....                      | 105 |
| Bezpieczne połączenia .....                 | 108 |
| HeartBeat .....                             | 111 |
| WatchDog.....                               | 113 |
| Logi .....                                  | 115 |
| HTTP.....                                   | 117 |
| Broadcast i kanały .....                    | 118 |
| Powiązania .....                            | 119 |
| Wysyłanie dużych plików .....               | 120 |
| Kompresja .....                             | 122 |
| Flash.....                                  | 123 |

|                                                     |     |
|-----------------------------------------------------|-----|
| Obiekty niestandardowe.....                         | 125 |
| Grupy .....                                         | 127 |
| IOCP .....                                          | 129 |
| EPOLL .....                                         | 131 |
| ALPN .....                                          | 133 |
| Przekazywanie żądań HTTP .....                      | 135 |
| Quality Of Service .....                            | 137 |
| Kolejki.....                                        | 140 |
| Transakcje .....                                    | 142 |
| TCP Connections .....                               | 144 |
| SubProtocol.....                                    | 146 |
| Throttle.....                                       | 148 |
| Server-Sent Events.....                             | 149 |
| LoadBalancing .....                                 | 153 |
| Pliki .....                                         | 154 |
| Proxy .....                                         | 155 |
| Fragmentowane wiadomości .....                      | 156 |
| Komponenty.....                                     | 158 |
| TsgcWebSocketClient .....                           | 158 |
| Połącz z serwerem WebSocket.....                    | 160 |
| Klient otwiera połączenie .....                     | 161 |
| Zamknięcie połączenia przez klienta .....           | 164 |
| Klient utrzymuje otwarte połączenie .....           | 165 |
| Porzucone rozłączenia .....                         | 167 |
| Połącz serwer TCP .....                             | 169 |
| Połączenia TIME_WAIT.....                           | 170 |
| Przekierowania WebSocket.....                       | 171 |
| Połącz bezpieczny serwer .....                      | 172 |
| Certyfikaty OpenSSL.....                            | 173 |
| Certyfikaty SChannel.....                           | 174 |
| SChannel — pobieranie informacji o połączeniu ..... | 176 |
| Wysyłanie wiadomości tekstowej przez klienta.....   | 177 |
| Klient wysyła wiadomość binarną.....                | 178 |
| Klient wysyła wiadomości tekstowe i binarne.....    | 179 |
| Odbieranie wiadomości tekstowych .....              | 180 |
| Odbieranie wiadomości binarnych .....               | 181 |
| Uwierzytelnianie klienta .....                      | 182 |
| Wyjątki klienta .....                               | 184 |
| Uzgadnianie WebSocket po stronie klienta.....       | 185 |
| Client Register Protocol .....                      | 186 |
| Client Proxies.....                                 | 187 |
| TsgcTCPClient.....                                  | 188 |
| TsgcTCPClient .....                                 | 189 |
| TsgcWebSocketServer .....                           | 190 |
| Uruchomienie serwera .....                          | 192 |
| Powiązania serwera.....                             | 193 |
| Uruchamianie i zatrzymywanie serwera .....          | 194 |
| Utrzymywanie aktywności serwera .....               | 195 |
| SSL serwera .....                                   | 196 |
| Server SSL SChannel.....                            | 199 |
| Weryfikacja certyfikatu serwera .....               | 202 |
| Utrzymywanie połączeń po stronie serwera .....      | 203 |
| Serwer zwykłego TCP .....                           | 204 |
| Zamknięcie połączenia przez serwer .....            | 205 |

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| Połączenia klientów .....                                            | 206 |
| Uwierzytelnianie serwera.....                                        | 207 |
| Serwer wysyła wiadomość tekstową .....                               | 209 |
| Serwer: wysyłanie binarnej wiadomości .....                          | 210 |
| Serwer odbiera wiadomość tekstową.....                               | 212 |
| Server Receive Binary Message .....                                  | 213 |
| Serwer odczytuje nagłówki od klienta.....                            | 214 |
| TsgcWebSocketHTTPServer .....                                        | 215 |
| Żądania serwera HTTP .....                                           | 217 |
| HTTP Dispatch Files.....                                             | 218 |
| Serwer HTTP/2 .....                                                  | 219 |
| HTTP/2 Server Push .....                                             | 220 |
| HTTP/2 Alternate Service.....                                        | 222 |
| Wątki serwera HTTP/2 .....                                           | 223 |
| Błąd HTTP 404 bez treści odpowiedzi .....                            | 226 |
| Sesje serwera HTTP.....                                              | 227 |
| HTTP Server Stream Video .....                                       | 229 |
| Server SSL SChannel.....                                             | 230 |
| TsgcWebSocketServer_HTTPAPI .....                                    | 233 |
| Rezerwacja adresu URL HTTPAPI.....                                   | 236 |
| HTTPAPI FineTune .....                                               | 238 |
| HTTPAPI Server SSL .....                                             | 241 |
| Certyfikaty z podpisem własnym.....                                  | 243 |
| HTTPAPI — wyłączenie HTTP/2 .....                                    | 244 |
| Niestandardowe nagłówki HTTPAPI.....                                 | 245 |
| HTTPAPI — wysyłanie odpowiedzi tekstowej .....                       | 246 |
| HTTPAPI Wysyłanie odpowiedzi z plikiem .....                         | 248 |
| HTTPAPI OnDisconnect not fired.....                                  | 250 |
| TsgcWebSocketClient_WinHTTP .....                                    | 251 |
| TsgcWebSocketFirewall.....                                           | 252 |
| Zapora: czarna lista i lista dozwolonych.....                        | 261 |
| Zapora sieciowa: Ochrona przed atakami brute force .....             | 263 |
| Zapora sieciowa: wykrywanie SQL Injection i XSS .....                | 265 |
| Firewall: Ograniczanie szybkości i ochrona przed atakami flood ..... | 268 |
| Zapora: Wykrywanie botów.....                                        | 271 |
| TsgcWSRateLimiter.....                                               | 275 |
| TsgcWSCircuitBreaker .....                                           | 289 |
| TsgcWSAPIKeyManager.....                                             | 302 |
| TsgcWebSocketLoadBalancerServer .....                                | 317 |
| TsgcWebSocketProxyServer .....                                       | 319 |
| TsgcIWWebSocketClient .....                                          | 320 |
| TsgcWSConnection.....                                                | 322 |
| Protokoły.....                                                       | 325 |
| Protokoły JavaScript .....                                           | 328 |
| Protokół MQTT .....                                                  | 332 |
| TsgcWSPClient_MQTT.....                                              | 334 |
| Client MQTT Connect .....                                            | 336 |
| Łączenie z serwerami Mosquitto MQTT .....                            | 338 |
| Sesje klienta MQTT .....                                             | 339 |
| Wersja MQTT klienta .....                                            | 340 |
| MQTT Publish Subscribe .....                                         | 341 |
| Tematy MQTT .....                                                    | 342 |
| MQTT Subscribe .....                                                 | 344 |
| Publikowanie wiadomości MQTT .....                                   | 345 |

|                                                          |     |
|----------------------------------------------------------|-----|
| Odbieranie wiadomości MQTT .....                         | 346 |
| MQTT Publish i oczekiwanie na odpowiedź .....            | 348 |
| MQTT Usuń zachowane wiadomości .....                     | 349 |
| Protokół AMQP .....                                      | 350 |
| TsgcWSPClient_AMQP .....                                 | 352 |
| Połączenie AMQP klienta .....                            | 356 |
| Rozłączenie klienta AMQP .....                           | 358 |
| Kanały AMQP .....                                        | 359 |
| Wymiana AMQP .....                                       | 361 |
| Kolejki AMQP .....                                       | 363 |
| Publikowanie wiadomości AMQP .....                       | 368 |
| AMQP Odbieranie wiadomości .....                         | 369 |
| Odbieranie wiadomości AMQP .....                         | 371 |
| QoS AMQP .....                                           | 372 |
| Transakcje AMQP .....                                    | 373 |
| Protocol Kafka .....                                     | 375 |
| TsgcWSPClient_Kafka .....                                | 376 |
| Client Kafka Connect .....                               | 378 |
| Client Kafka Disconnect .....                            | 379 |
| Kafka Produce Messages .....                             | 380 |
| Kafka Consume Messages .....                             | 381 |
| Kafka Consumer Groups .....                              | 382 |
| Kafka Topics .....                                       | 383 |
| Kafka Offsets .....                                      | 384 |
| Protokół AMQP1 .....                                     | 385 |
| TsgcWSPClient_AMQP1 .....                                | 387 |
| Połączenie klienta AMQP1 .....                           | 391 |
| Rozłączenie klienta AMQP1 .....                          | 393 |
| Klient AMQP1 — limit czasu bezczynności połączenia ..... | 394 |
| Stan połączenia klienta AMQP1 .....                      | 395 |
| Uwierzytelnianie klienta AMQP1 .....                     | 397 |
| Klient AMQP1 Azure MessageBus .....                      | 398 |
| Sesje AMQP1 .....                                        | 401 |
| AMQP1 Łączy .....                                        | 404 |
| AMQP1 Sender Links .....                                 | 406 |
| Łączy odbiorcy AMQP1 .....                               | 409 |
| AMQP1 — wysyłanie wiadomości .....                       | 412 |
| AMQP1 Read Message .....                                 | 414 |
| Protokół STOMP .....                                     | 415 |
| TsgcWSPClient_STOMP .....                                | 416 |
| TsgcWSPClient_STOMP_RabbitMQ .....                       | 417 |
| TsgcWSPClient_STOMP_ActiveMQ .....                       | 418 |
| Protokół AppRTC .....                                    | 420 |
| TsgcWSPServer_AppRTC .....                               | 421 |
| Protokół WebRTC .....                                    | 423 |
| TsgcWSPServer_WebRTC .....                               | 424 |
| Protocol WebRTC Javascript .....                         | 425 |
| Protokół WAMP .....                                      | 427 |
| TsgcWSPServer_WAMP .....                                 | 429 |
| TsgcWSPClient_WAMP .....                                 | 430 |
| Protokół WAMP Javascript .....                           | 431 |
| Subskrybenci .....                                       | 435 |
| Wydawcy .....                                            | 436 |
| Proste RPC .....                                         | 437 |

|                                                               |     |
|---------------------------------------------------------------|-----|
| Wyniki postępu RPC .....                                      | 439 |
| Protocol WAMP2 .....                                          | 441 |
| TsgcWSPClient_WAMP2 .....                                     | 442 |
| Protokół domyślny .....                                       | 446 |
| TsgcWSPServer_sgc .....                                       | 448 |
| TsgcWSPClient_sgc .....                                       | 449 |
| TsgcIWWSPClient_sgc .....                                     | 450 |
| Domyślny protokół JavaScript .....                            | 451 |
| Zestaw danych protokołu .....                                 | 456 |
| TsgcWSPServer_Dataset .....                                   | 458 |
| TsgcWSPClient_Dataset .....                                   | 459 |
| TsgcIWWSPClient_Dataset .....                                 | 460 |
| Protocol Dataset Javascript .....                             | 461 |
| Replikacja tabeli zestawu danych protokołu .....              | 464 |
| Powiadomienia o aktualizacjach zestawu danych protokołu ..... | 466 |
| Protokół Files .....                                          | 468 |
| TsgcWSPServer_Files .....                                     | 469 |
| TsgcWSPClient_Files .....                                     | 470 |
| TsgcWSMessageFile .....                                       | 471 |
| Jak wysyłać pliki do serwera .....                            | 472 |
| Jak wysyłać pliki do klientów .....                           | 473 |
| Jak wysyłać duże pliki .....                                  | 474 |
| Obecność protokołu .....                                      | 475 |
| TsgcWSPServer_Presence .....                                  | 476 |
| TsgcWSPPresenceMessage .....                                  | 477 |
| TsgcWSPClient_Presence .....                                  | 479 |
| Protocol Presence Javascript .....                            | 480 |
| Protokół E2EE .....                                           | 484 |
| TsgcWSPServer_E2EE .....                                      | 487 |
| TsgcWSPClient_E2EE .....                                      | 488 |
| API .....                                                     | 489 |
| API Binance .....                                             | 491 |
| Binance Connect WebSocket API .....                           | 500 |
| Binance — subskrypcja kanału WebSocket .....                  | 501 |
| Binance — pobieranie danych rynkowych .....                   | 502 |
| Binance Private REST API .....                                | 503 |
| Binance Trade Spot .....                                      | 504 |
| Czas żądań prywatnych Binance .....                           | 506 |
| Binance — wypłata .....                                       | 507 |
| API Binance Futures .....                                     | 508 |
| API Binance Futures — Handel .....                            | 515 |
| API SocketIO .....                                            | 517 |
| API Coinbase .....                                            | 521 |
| Coinbase Connect WebSocket API .....                          | 527 |
| Coinbase Subscribe WebSocket Channel .....                    | 528 |
| Coinbase Get Market Data .....                                | 529 |
| Coinbase Private REST API .....                               | 530 |
| Coinbase Private Requests Time .....                          | 531 |
| Coinbase Place Orders .....                                   | 532 |
| Coinbase SandBox Account .....                                | 533 |
| API SignalRCore .....                                         | 534 |
| API SignalR .....                                             | 542 |
| API Kraken .....                                              | 546 |
| API Kraken WebSockets Public .....                            | 548 |

|                                                    |     |
|----------------------------------------------------|-----|
| API Kraken WebSockets Private .....                | 556 |
| API Kraken REST Public .....                       | 560 |
| Prywatne API REST Kraken.....                      | 562 |
| API Kraken Futures .....                           | 566 |
| API Kraken Futures WebSockets Public.....          | 568 |
| API Kraken Futures WebSockets Prywatne.....        | 581 |
| API Kraken Futures REST Public .....               | 590 |
| API Kraken Futures REST Private .....              | 592 |
| API Pusher .....                                   | 599 |
| API Bitmex .....                                   | 610 |
| Bitmex Connect WebSocket API .....                 | 615 |
| Bitmex Subskrybuj kanał WebSocket.....             | 616 |
| Jak złożyć zlecenie w Bitmex .....                 | 617 |
| API Bitfinex .....                                 | 620 |
| API Kucoin .....                                   | 626 |
| Kucoin — połączenie z WebSocket API.....           | 633 |
| Kucoin — subskrypcja kanału WebSocket.....         | 634 |
| Kucoin — pobieranie danych rynkowych .....         | 635 |
| Kucoin Private REST API .....                      | 636 |
| Kucoin Trade Spot .....                            | 637 |
| Czas prywatnych żądań Kucoin.....                  | 639 |
| API Kucoin Futures .....                           | 640 |
| Kucoin Futures Połącz WebSocket API.....           | 646 |
| Kucoin Futures — subskrypcja kanału WebSocket..... | 647 |
| Kucoin Futures — pobieranie danych rynkowych ..... | 648 |
| Kucoin Futures — prywatne REST API.....            | 649 |
| Handel kontraktami Kucoin Futures .....            | 650 |
| Czas żądań prywatnych Kucoin Futures.....          | 653 |
| API 3Commas.....                                   | 654 |
| API OKX .....                                      | 659 |
| API XTB .....                                      | 665 |
| API Bybit .....                                    | 669 |
| API Forex.....                                     | 674 |
| Forex Connect REST API.....                        | 676 |
| Forex Trade Orders .....                           | 677 |
| Zapytania o konto Forex .....                      | 679 |
| Dane rynku Forex .....                             | 680 |
| Forex Connect WebSocket API.....                   | 681 |
| Forex Subskrybuj strumienie .....                  | 682 |
| API Blockchain.....                                | 684 |
| API Cex.....                                       | 686 |
| API Cex Plus.....                                  | 696 |
| API Discord .....                                  | 701 |
| API OpenAI .....                                   | 706 |
| API MEXC.....                                      | 708 |
| API MEXC Futures.....                              | 712 |
| API Bitget .....                                   | 716 |
| API GateIO .....                                   | 719 |
| API Deribit.....                                   | 722 |
| API Crypto.com .....                               | 725 |
| API HTX.....                                       | 728 |
| API Whatsapp .....                                 | 731 |
| WhatsApp Tworzenie aplikacji.....                  | 737 |
| Identyfikator numeru telefonu WhatsApp .....       | 738 |

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| Token WhatsApp.....                                             | 739 |
| Webhook WhatsApp .....                                          | 740 |
| Bezpieczeństwo WhatsApp.....                                    | 741 |
| Wysyłanie wiadomości WhatsApp .....                             | 742 |
| WhatsApp — wysyłanie wiadomości interaktywnych .....            | 745 |
| WhatsApp Wysyłanie Wiadomości Szablonowych .....                | 748 |
| WhatsApp — odbieranie wiadomości i powiadomień o statusie ..... | 750 |
| WhatsApp — wysyłanie plików .....                               | 752 |
| WhatsApp — pobieranie multimediów .....                         | 755 |
| API Telegram .....                                              | 756 |
| Wyślij wiadomość Telegram z przyciskami wbudowanymi .....       | 769 |
| Wyślij wiadomość Telegram z przyciskami.....                    | 770 |
| Wyślij pogrubioną wiadomość Telegram.....                       | 771 |
| Telegram Chat nie znaleziono jako Bot .....                     | 772 |
| Telegram Sponsored Messages .....                               | 773 |
| Wyślij wiadomość faktury Telegram .....                         | 774 |
| Telegram — pobieranie członków SuperGroup.....                  | 775 |
| Dodaj proxy Telegram .....                                      | 777 |
| Rejestracja użytkownika Telegram .....                          | 778 |
| RCON .....                                                      | 779 |
| CryptoHopper .....                                              | 781 |
| RTCMultiConnection.....                                         | 787 |
| WebPush .....                                                   | 789 |
| TsgcWSAPIServer_WebPush .....                                   | 791 |
| TsgcWebPush_Client .....                                        | 792 |
| Rozszerzenia .....                                              | 793 |
| PerMessage-Deflate .....                                        | 794 |
| Deflate-Frame .....                                             | 795 |
| MCP .....                                                       | 796 |
| TsgcWSAPIServer_MCP .....                                       | 797 |
| Sesje serwera MCP .....                                         | 800 |
| Narzędzia serwera MCP .....                                     | 801 |
| Monity serwera MCP .....                                        | 805 |
| Zasoby serwera MCP .....                                        | 809 |
| Katalogi główne serwera MCP .....                               | 813 |
| Próbkowanie serwera MCP .....                                   | 815 |
| MCP Server Elicitation .....                                    | 817 |
| TsgcWSAPIClient_MCP .....                                       | 819 |
| Narzędzia klienta MCP .....                                     | 821 |
| Monity klienta MCP .....                                        | 823 |
| Zasoby klienta MCP .....                                        | 825 |
| Korzenie klienta MCP.....                                       | 827 |
| Próbkowanie klienta MCP .....                                   | 829 |
| Elicitation klienta MCP .....                                   | 831 |
| OpenAI.....                                                     | 833 |
| TsgcHTTP_API_OpenAI.....                                        | 843 |
| Uzupełnienie OpenAI .....                                       | 844 |
| OpenAI Chat .....                                               | 845 |
| OpenAI Edit .....                                               | 846 |
| OpenAI Audio.....                                               | 847 |
| Moderacja OpenAI.....                                           | 848 |
| OpenAI RealTime .....                                           | 849 |
| OpenAI Responses.....                                           | 850 |
| OpenAI Speech .....                                             | 851 |



|                                                 |     |
|-------------------------------------------------|-----|
| Dostrajanie OpenAI .....                        | 852 |
| OpenAI Batch .....                              | 853 |
| OpenAI Uploads .....                            | 854 |
| Aplikacje OpenAI .....                          | 855 |
| TsgcAIOpenAIAssistant .....                     | 857 |
| TsgcAIOpenAIAssistant Wyszukiwanie plików ..... | 859 |
| TsgcAIOpenAIAssistant Streaming .....           | 862 |
| TsgcAIOpenAIAssistant Function Calling .....    | 864 |
| OpenAI Audio .....                              | 866 |
| TsgcAudioRecorderMCI .....                      | 867 |
| TsgcAudioPlayerMCI .....                        | 868 |
| TsgcTextToSpeechSystem .....                    | 869 |
| TsgcTextToSpeechGoogle .....                    | 870 |
| TsgcTextToSpeechAmazon .....                    | 871 |
| TsgcAIChat .....                                | 872 |
| TsgcAIOpenAIChatBot .....                       | 873 |
| TsgcAIOpenAITranslator .....                    | 874 |
| TsgcAIOpenAIEmbeddings .....                    | 875 |
| TsgcAIDatabaseVectorFile .....                  | 876 |
| TsgcAIDatabaseVectorPinecone .....              | 877 |
| Embeddingi Tworzenie wektorów .....             | 878 |
| Chatbot Embeddings .....                        | 879 |
| Pinecone .....                                  | 880 |
| Anthropic .....                                 | 883 |
| TsgcHTTP_API_Anthropic .....                    | 888 |
| Wiadomości Anthropic .....                      | 889 |
| Anthropic Vision .....                          | 891 |
| Anthropic Tool Use .....                        | 893 |
| Modele Anthropic .....                          | 895 |
| Partie Anthropic .....                          | 896 |
| Anthropic Extended Thinking .....               | 897 |
| Dokumenty Anthropic .....                       | 899 |
| Anthropic Prompt Caching .....                  | 901 |
| Anthropic Citations .....                       | 903 |
| Wyszukiwanie w sieci Anthropic .....            | 904 |
| Anthropic Structured Outputs .....              | 906 |
| Pliki Anthropic .....                           | 908 |
| Łącznik Anthropic MCP .....                     | 910 |
| Gemini .....                                    | 912 |
| TsgcHTTP_API_Gemini .....                       | 915 |
| DeepSeek .....                                  | 916 |
| TsgcHTTP_API_DeepSeek .....                     | 918 |
| Wiadomości DeepSeek .....                       | 919 |
| DeepSeek Vision .....                           | 920 |
| Modele DeepSeek .....                           | 921 |
| Gemini .....                                    | 922 |
| Wiadomości Gemini .....                         | 925 |
| Gemini Vision .....                             | 927 |
| Modele Gemini .....                             | 928 |
| Gemini — strukturyzowane dane wyjściowe .....   | 929 |
| Zliczanie tokenów Gemini .....                  | 931 |
| Osadzenia Gemini .....                          | 932 |
| Gemini Tool Use .....                           | 933 |
| Ollama .....                                    | 935 |

|                                                           |      |
|-----------------------------------------------------------|------|
| TsgcHTTP_API_Ollama.....                                  | 937  |
| Wiadomości Ollama.....                                    | 938  |
| Modele Ollama.....                                        | 939  |
| Ollama Embeddings.....                                    | 940  |
| Grok .....                                                | 941  |
| TsgcHTTP_API_Grok .....                                   | 943  |
| Wiadomości Grok .....                                     | 944  |
| Grok Vision.....                                          | 945  |
| Modele Grok .....                                         | 946  |
| Mistral AI .....                                          | 947  |
| TsgcHTTP_API_Mistral .....                                | 949  |
| Wiadomości Mistral .....                                  | 950  |
| Mistral Vision.....                                       | 951  |
| Mistral Models .....                                      | 952  |
| Osadzenia Mistral .....                                   | 953  |
| IoT.....                                                  | 954  |
| Klient IoT Amazon MQTT .....                              | 955  |
| IoT Azure MQTT Client .....                               | 967  |
| HTTP.....                                                 | 975  |
| HTTP2.....                                                | 977  |
| TsgcHTTP2Client .....                                     | 979  |
| Metoda żądania HTTP/2 .....                               | 981  |
| HTTP/2 Server Push .....                                  | 982  |
| Pobieranie pliku przez HTTP/2 .....                       | 983  |
| HTTP/2 Partial Responses .....                            | 984  |
| Nagłówki HTTP/2.....                                      | 985  |
| Zamknięcie połączenia przez klienta .....                 | 986  |
| Utrzymywanie aktywnego połączenia klienta .....           | 987  |
| Przyczyna rozłączenia HTTP/2.....                         | 988  |
| Oczekujące żądania klienta.....                           | 989  |
| Uwierzytelnianie klienta .....                            | 990  |
| HTTP/2 i OAuth2 .....                                     | 992  |
| TsgcHTTP2ConnectionClient.....                            | 994  |
| TsgcHTTP2RequestProperty .....                            | 995  |
| TsgcHTTP2ResponseProperty .....                           | 996  |
| Apple Push Notifications .....                            | 997  |
| Generuj zdalne powiadomienie APNs .....                   | 998  |
| Wysyłanie żądań powiadomień do APNs .....                 | 999  |
| Połączenie z APNs oparte na tokenach.....                 | 1001 |
| Połączenie z APNs oparte na certyfikatach .....           | 1003 |
| HTTP1.....                                                | 1005 |
| OAuth2.....                                               | 1011 |
| TsgcHTTP_OAuth2_Client .....                              | 1013 |
| Klient OAuth2 dla aplikacji internetowych .....           | 1018 |
| Klient OAuth2 dla aplikacji desktopowych.....             | 1019 |
| TsgcHTTP_OAuth2_Client_Google.....                        | 1020 |
| TsgcHTTP_OAuth2_Client_Microsoft.....                     | 1021 |
| Authorization Code Grant (RFC 6749).....                  | 1022 |
| Kod autoryzacji z PKCE (RFC 7636).....                    | 1024 |
| Client Credentials Grant (RFC 6749) .....                 | 1026 |
| Resource Owner Password Credentials Grant (RFC 6749)..... | 1028 |
| Device Authorization Grant (RFC 8628).....                | 1030 |
| DPoP — Demonstrating Proof of Possession (RFC 9449).....  | 1032 |
| TsgcHTTP_OAuth2_Server.....                               | 1036 |

|                                                                |      |
|----------------------------------------------------------------|------|
| Przykład serwera OAuth2 .....                                  | 1040 |
| Dostosowanie HTML strony logowania OAuth2 .....                | 1042 |
| Punkty końcowe serwera OAuth2 .....                            | 1043 |
| OAuth2 Register Apps .....                                     | 1044 |
| Odzyskiwanie tokenów dostępu OAuth2.....                       | 1045 |
| Uwierzytelnianie serwera OAuth2.....                           | 1046 |
| OAuth2 None Authenticate URLs .....                            | 1048 |
| Authorization Code Grant.....                                  | 1049 |
| Przyznanie poświadczeń klienta (Client Credentials Grant)..... | 1051 |
| Resource Owner Password Credentials Grant .....                | 1053 |
| Odświeżanie tokena (Refresh Token Grant) .....                 | 1055 |
| Device Authorization Grant (RFC 8628).....                     | 1057 |
| Unieważnienie tokenu (RFC 7009) .....                          | 1059 |
| Token Introspection (RFC 7662) .....                           | 1061 |
| Walidacja DPoP (RFC 9449) .....                                | 1063 |
| TsgcHTTP_OAuth2_Server_Provider.....                           | 1065 |
| Dostawca OAuth2 Azure AD.....                                  | 1067 |
| Prywatne punkty końcowe dostawcy OAuth2.....                   | 1069 |
| Uwierzytelnianie dostawcy OAuth2 .....                         | 1070 |
| Żądania dostawcy OAuth2.....                                   | 1073 |
| JWT.....                                                       | 1074 |
| TsgcHTTP_JWT_Client .....                                      | 1076 |
| TsgcHTTP_JWT_Server.....                                       | 1080 |
| WebAuthn .....                                                 | 1081 |
| TsgcWSAPIServer_WebAuthn.....                                  | 1082 |
| Rejestracja WebAuthn.....                                      | 1087 |
| Żądanie rejestracji WebAuthn.....                              | 1091 |
| Odpowiedź na rejestrację WebAuthn.....                         | 1093 |
| Wynik rejestracji WebAuthn.....                                | 1095 |
| Uwierzytelnianie WebAuthn.....                                 | 1097 |
| Żądanie uwierzytelnienia WebAuthn.....                         | 1100 |
| Odpowiedź uwierzytelniania WebAuthn .....                      | 1101 |
| Wynik uwierzytelniania WebAuthn .....                          | 1103 |
| WebAuthn MDS.....                                              | 1104 |
| Autoryzacja WebAuthn.....                                      | 1106 |
| WebAuthn Authorization HTTP .....                              | 1107 |
| WebAuthn Authorization WebSocket .....                         | 1108 |
| Klient Javascript WebAuthn .....                               | 1109 |
| Amazon SQS.....                                                | 1115 |
| Klucze Google OAuth2.....                                      | 1121 |
| Google Service Accounts .....                                  | 1123 |
| Google Cloud Pub/Sub .....                                     | 1125 |
| Google Calendar .....                                          | 1134 |
| Synchronizacja kalendarzy Google Calendar .....                | 1141 |
| Synchronizacja zdarzeń kalendarza Google .....                 | 1142 |
| Google Calendar RefreshToken.....                              | 1143 |
| Konto usługi Google Calendar.....                              | 1144 |
| Google Cloud FCM.....                                          | 1145 |
| TsgcWebView2 .....                                             | 1149 |
| Nawigacja WebView2.....                                        | 1152 |
| WebView2 JavaScript .....                                      | 1155 |
| Pliki cookie WebView2.....                                     | 1158 |
| Pobieranie plików WebView2.....                                | 1160 |
| Ustawienia WebView2 .....                                      | 1162 |

|                                                          |      |
|----------------------------------------------------------|------|
| WebView2 Advanced Features .....                         | 1164 |
| TsgcUDPCClient .....                                     | 1168 |
| TsgcUDPSTUN .....                                        | 1169 |
| STUN .....                                               | 1170 |
| TsgcSTUNClient .....                                     | 1171 |
| Retransmisje UDP klienta STUN .....                      | 1173 |
| Długoterminowe dane uwierzytelniające klienta STUN ..... | 1174 |
| Atrybuty klienta STUN .....                              | 1175 |
| TsgcSTUNServer .....                                     | 1176 |
| Długoterminowe poświadczenia serwera STUN .....          | 1177 |
| Alternatywny serwer STUN .....                           | 1178 |
| TURN .....                                               | 1179 |
| TsgcTURNClient .....                                     | 1181 |
| Adres IP alokacji klienta TURN .....                     | 1183 |
| TURN Client Create Permissions .....                     | 1184 |
| Klient TURN wysyła wskazanie .....                       | 1185 |
| Kanały klienta TURN .....                                | 1186 |
| TsgcTURNServer .....                                     | 1187 |
| Długoterminowe poświadczenia serwera TURN .....          | 1189 |
| Alokacje serwera TURN .....                              | 1190 |
| ICE .....                                                | 1192 |
| TsgcICEClient .....                                      | 1193 |
| ICE — zbieranie kandydatów .....                         | 1195 |
| Para kandydatów ICE .....                                | 1196 |
| TsgcRTCPeerConnection .....                              | 1197 |
| Serwer WebSocket RTCPeerConnection .....                 | 1200 |
| RTCPeerConnection Klient WebSocket .....                 | 1201 |
| RTCPeerConnection STUN TURN .....                        | 1202 |
| Sygnalizacja RTCPeerConnection .....                     | 1203 |
| RTCPeerConnection ICE .....                              | 1204 |
| RTCPeerConnection DTLS .....                             | 1206 |
| RTCPeerConnection Data .....                             | 1208 |
| Datasnap .....                                           | 1209 |
| TsgcWSHTTPWebBrokerBridgeServer .....                    | 1210 |
| TsgcWSHTTP2WebBrokerBridgeServer .....                   | 1212 |
| TsgcWSServer_HTTPAPI_WebBrokerBridge .....               | 1213 |
| OpenAPI .....                                            | 1214 |
| OpenAPI .....                                            | 1214 |
| OpenAPI Parser Pascal .....                              | 1216 |
| Dodatkowe właściwości OpenAPI .....                      | 1219 |
| OpenAPI Command Line .....                               | 1221 |
| Biblioteka OpenAPI .....                                 | 1223 |
| OpenAPI API .....                                        | 1226 |
| OpenAPI Client .....                                     | 1227 |
| Serwer OpenAPI .....                                     | 1233 |
| Serwer OpenAPI w podejściu Spec-First .....              | 1236 |
| Serwer OpenAPI Code-First .....                          | 1238 |
| OpenAPI Amazon AWS .....                                 | 1240 |
| Poświadczenia Amazon AWS dla sgcOpenAPI .....            | 1247 |
| OpenAPI Amazon AWS S3 .....                              | 1250 |
| OpenAPI Google Cloud .....                               | 1251 |
| OpenAPI Google Cloud OAuth2 .....                        | 1257 |
| OpenAPI konta usług Google Cloud .....                   | 1259 |
| OpenAPI Google Cloud PubSub .....                        | 1262 |

|                                                 |      |
|-------------------------------------------------|------|
| OpenAPI Google Cloud Calendar .....             | 1263 |
| OpenAPI Google Drive.....                       | 1264 |
| OpenAPI Microsoft.....                          | 1266 |
| OpenAPI Microsoft Tenant .....                  | 1273 |
| OpenAPI Microsoft — rejestracja aplikacji ..... | 1274 |
| Kod OAuth2 Microsoft OpenAPI.....               | 1276 |
| OpenAPI Microsoft OAuth2 Credentials.....       | 1278 |
| OpenAPI Microsoft Graph.....                    | 1279 |
| API .....                                       | 1280 |
| AbstractApi Geolocation .....                   | 1281 |
| Przykłady demonstracyjne .....                  | 1282 |
| Czat serwera.....                               | 1282 |
| Czat klienta .....                              | 1285 |
| Klient.....                                     | 1287 |
| Klient MQTT .....                               | 1289 |
| Client SocketIO.....                            | 1291 |
| Monitor serwera .....                           | 1292 |
| Migawki serwera.....                            | 1295 |
| Migawki klientów .....                          | 1296 |
| Prześlij plik .....                             | 1297 |
| Uwierzytelnianie serwera.....                   | 1299 |
| KendoUI_Grid .....                              | 1300 |
| ServerSentEvents .....                          | 1302 |
| Serwer WebRTC .....                             | 1303 |
| Server AppRTC.....                              | 1304 |
| Klient Telegram.....                            | 1306 |
| Rozwiązania zewnętrzne.....                     | 1308 |
| Coturn .....                                    | 1308 |
| Licencja.....                                   | 1311 |
| Licencja.....                                   | 1311 |
| Indeks.....                                     | 1314 |
| PDF-Back-Cover.....                             | 1327 |

# Wprowadzenie

**sgcWebSockets** to profesjonalna biblioteka komponentów do komunikacji w czasie rzeczywistym, przesyłania wiadomości i integracji z AI. Zapewnia gotowe do produkcji implementacje WebSockets, HTTP/2, MQTT, AMQP, WebRTC, Server-Sent Events oraz ponad 30 łączników API innych firm, dostępnych przez spójną, opartą na zdarzeniach architekturę komponentów. Niezależnie od tego, czy budowane są platformy handlowe, bramy IoT, aplikacje oparte na AI, czy systemy przesyłania wiadomości dla przedsiębiorstw, **sgcWebSockets** dostarcza niezbędne fundamenty sieciowe.

Biblioteka obsługuje **Delphi 7 do Delphi 13, C++Builder, Lazarus/FreePascal** oraz **.NET** (Framework 2.0+, .NET Core 1.0+, .NET 5–9, .NET Standard). Aplikacje mogą być kierowane na platformy Windows, macOS, Linux, iOS i Android z jednej bazy kodu przy użyciu VCL i FireMonkey.

## Funkcje Delphi / C++Builder / FreePascal

### Rdzeń WebSocket

- W pełni funkcjonalny **wielowątkowy serwer WebSocket** implementujący **RFC 6455**.
- Obsługa wiadomości **tekstowych i binarnych**.
- **Kompresja wiadomości** przy użyciu rozszerzenia **PerMessage\_Deflate (RFC 7692)**.
- Wbudowany **WatchDog i HeartBeat** dla automatycznego ponownego połączenia i monitorowania połączenia.
- **Równoważenie obciążenia** Serwer do dystrybucji połączeń między wieloma serwerami zaplecza.
- **Serwer proxy** umożliwiający przeglądarkom internetowym łączenie się z dowolnym serwerem TCP.
- Połączenia klienta przez serwery **HTTP Proxy i SOCKS Proxy**.
- Zdarzenia: **OnConnect, OnDisconnect, OnMessage, OnError, OnHandshake**.

### Serwery wysokiej wydajności

- **Microsoft HTTP Server API (HTTP.SYS)** z **IOCP** dla wydajnych serwerów Windows.
- Serwery Indy z obsługą **IOCP** (Windows), **EPOLL** (Linux) lub domyślny model jeden-wątek-na-połączenie.
- **WebSocket i HTTP/2** na **tym samym porcie**.
- **Serwer WebBroker** obsługujący **DataSnap, HTTP/2 i WebSocket** na jednym porcie.
- Klient WebSocket oparty na **WinHTTP API**.

### Bezpieczeństwo i uwierzytelnianie

- **SSL/TLS** dla komponentów serwera i klienta. Obsługiwane **OpenSSL 1.1.1 i 3.0**. Klient obsługuje **SChannel** w systemie Windows.
- Komponenty klienta i serwera **OAuth 2.0 i JWT** (JSON Web Token).

- **Protokół serwera WebAuthn** do uwierzytelniania opartego na kluczach dostępu.
- **E2EE** (End-To-End Encryption) sub-protocol.

## AI i uczenie maszynowe

- **OpenAI i Claude** integracja z komponentami **MCP Server** i **MCP Client**.
- Komponenty aplikacji **ChatBot** i **Translator** z obsługą poleceń głosowych.
- **Osadzenia (Embeddings)** i obsługa **bazy wektorowej** (plikowej i Pinecone).
- **Text-To-Speech** przez domyślny systemowy, Google Cloud i Amazon AWS.

## Protokół HTTP/2

- Pełna implementacja **klienta i serwera HTTP/2**.
- **Server-Sent Events** (powiadomienia push) przez HTTP.

## Protokoły przesyłania wiadomości

- **MQTT 3.1.1 i 5.0** przez WebSocket i zwykły TCP.
- **AMQP 0.9.1 i 1.0** z obsługą brokera RabbitMQ.
- **STOMP** przez WebSocket i TCP (brokerzy ActiveMQ, RabbitMQ).
- **WAMP 1.0 i 2.0** — protokoły klienta i serwera.
- **JSON-RPC 2.0** subprotokół.

## P2P i komunikacja w czasie rzeczywistym

- **WebRTC** protokół serwera.
- Komponenty klienta i serwera **STUN** oraz **TURN**.
- **ICE** klient do przechodzenia przez NAT.
- **UDP** klient i serwer.

## Integracje API

- **30+ wbudowanych konektorów API**: Binance, Coinbase, Kraken, Bitfinex, Discord, WhatsApp, Telegram i inne.
- **SignalR** i **SignalR Core**, **Socket.IO** oraz klienty **Pusher**.
- Usługi w chmurze: **Amazon SQS**, **Google Cloud Pub/Sub**, **Google Calendar**.

## Wieloplatformowy

- **VCL** i **FireMonkey** (Windows, macOS, iOS, Android, Linux).
- **C++Builder** wsparcie.
- **Lazarus / FreePascal** wsparcie.

- Wbudowane **biblioteki JavaScript** dla klientów przeglądarek.

Biblioteka sgcWebSockets zawiera następujące komponenty:

### 1. sgcWebSockets

- **TsgcWebSocketClient**: Klient WebSocket oparty na bibliotece Indy.
- **TsgcWebSocketServer**: Serwer WebSocket oparty na bibliotece Indy
- **TsgcWebSocketHTTPServer**: Serwer WebSocket + HTTP oparty na bibliotece Indy.
- **TsgcWebSocketServer\_HTTPAPI**: Wysoko wydajny serwer WebSocket + HTTP oparty na Microsoft HTTP API (HTTP.SYS).
- **TsgcWebSocketClient\_WinHTTP**: Klient WebSocket oparty na bibliotece WinHTTP.
- **TsgcWebSocketFirewall**: Zapora serwera WebSocket z filtrowaniem IP, ochroną przed atakami brute force, ograniczaniem przepustowości i wykrywaniem zagrożeń.
- **TsgcWebSocketLoadBalancerServer**: Serwer równoważenia obciążenia dla protokołów WebSocket i HTTP.
- **TsgcWebSocketProxyServer**: Serwer proxy WebSocket umożliwiający przeglądarkom internetowym połączenie z serwerami TCP.

### 2. sgcWebSockets APIs

- **TsgcWSAPI\_Binance**: Klient Binance Spot, obsługuje WebSocket + API REST.
- **TsgcWSAPI\_Binance\_Futures**: Klient Binance Futures, obsługuje WebSocket + API REST.
- **TsgcWSAPI\_SocketIO**: Klient Socket.IO.
- **TsgcWSAPI\_Coinbase**: Klient Coinbase Pro, obsługuje WebSocket + API REST.
- **TsgcWSAPI\_Bitmex**: Klient Bitmex, obsługuje WebSocket + API REST.
- **TsgcWSAPI\_SignalR**: Klient WebSocket SignalR.
- **TsgcWSAPI\_SignalRCore**: Klient WebSocket SignalRCore.
- **TsgcWSAPI\_Pusher**: Klient WebSocket Pusher.
- **TsgcWSAPI\_Kraken**: API klienta Kraken, obsługuje WebSocket i API REST.
- **TsgcWSAPI\_Kraken\_Futures**: API klienta Kraken Futures, obsługuje WebSocket i API REST.
- **TsgcWSAPI\_Bitstamp**: Klient WebSocket Bitstamp.
- **TsgcWSAPI\_Cex**: Klient WebSocket Cex.
- **TsgcWSAPI\_Huobi**: Klient WebSocket Huobi.
- **TsgcWSAPI\_ThreeCommas**: API klienta ThreeCommas.
- **TsgcWSAPI\_Bitfinex**: API WebSocket Bitfinex.
- **TsgcWSAPI\_Discord**: Klient WebSocket Discord.
- **TsgcWSAPI\_BlockChain**: Klient WebSocket BlockChain.
- **TsgcWSAPI\_Bybit**: Klient WebSocket Bybit.
- **TsgcWSAPI\_OKX**: Klient WebSocket OKX.
- **TsgcWSAPI\_Kucoin**: Klient WebSocket KuCoin Spot.
- **TsgcWSAPI\_Kucoin\_Futures**: Klient WebSocket KuCoin Futures.
- **TsgcWSAPI\_Deribit**: Klient WebSocket Deribit.
- **TsgcWSAPI\_CryptoCom**: Klient WebSocket Crypto.com.
- **TsgcWSAPI\_HTX**: Klient WebSocket HTX.
- **TsgcWSAPI\_MEXC**: Klient WebSocket MEXC.
- **TsgcWSAPI\_Bitget**: Klient WebSocket Bitget.
- **TsgcWSAPI\_GateIO**: Klient WebSocket Gate.io.
- **TsgcWSAPI\_XTB**: Klient WebSocket XTB.
- **TsgcWSAPI\_CexPlus**: Klient WebSocket CexPlus.
- **TsgcWSAPI\_OpenAI**: Klient WebSocket API czasu rzeczywistego OpenAI.

### 3. sgcWebSocket Libs

- **TsgcTDLib\_Telegram**: Klient Telegram API.
- **TsgcWhatsApp\_Client**: Klient WhatsApp Business Cloud.
- **TsgcHTTP\_Cryptohopper**: Klient Cryptohopper API.
- **TsgcLib\_RCON**: Klient RCON.

### 4. Protokoły sgcWebSocket



- **TsgcWSPClient\_MQTT**: Klient MQTT (3.1.1 i 5.0). Obsługuje połączenia WebSocket i zwykłe TCP.
- **TsgcWSPClient\_AMQP1**: Klient AMQP 1.0.0. Obsługuje brokery RabbitMQ.
- **TsgcWSPClient\_AMQP**: Klient AMQP 0.9.1. Obsługuje brokery RabbitMQ.
- **TsgcWSPClient\_STOMP**: Klient STOMP, obsługuje połączenia WebSocket i zwykłe TCP.
  - **TsgcWSPClient\_STOMP\_ActiveMQ**: Klient STOMP dla brokera ActiveMQ.
  - **TsgcWSPClient\_STOMP\_RabbitMQ**: Klient STOMP dla brokera RabbitMQ.
- **TsgcWSPClient\_WAMP**: Protokół klienta WAMP 1.0.
- **TsgcWSPServer\_WAMP**: Protokół serwera WAMP 1.0.
- **TsgcWSPClient\_WAMP2**: Protokół klienta WAMP 2.0.
- **TsgcWSPServer\_AppRTC**: Serwer WebRTC oparty na projekcie Google AppRTC.
- **TsgcWSPServer\_WebRTC**: Protokół serwera WebRTC.
- **TsgcWSPClient\_sgc**: Protokół klienta WebSocket SGC oparty na JSON RPC.
- **TsgcWSPServer\_sgc**: Protokół serwera WebSocket SGC oparty na JSON RPC.
- **TsgcWSPClient\_Files**: Protokół klienta WebSocket do transferu plików.
- **TsgcWSPServer\_Files**: Protokół serwera WebSocket do transferu plików.
- **TsgcWSPClient\_Dataset**: Protokół klienta WebSocket do synchronizacji zestawów danych.
- **TsgcWSPServer\_Dataset**: Protokół serwera WebSocket do synchronizacji zestawów danych.
- **TsgcWSPClient\_Presence**: Protokół klienta WebSocket do obsługi obecności.
- **TsgcWSPServer\_Presence**: Protokół serwera WebSocket do obsługi obecności.
- **TsgcWSPClient\_E2EE**: Protokół klienta szyfrowania End-To-End.
- **TsgcWSPServer\_E2EE**: Protokół serwera szyfrowania End-To-End.

## 5. sgcWebSockets HTTP

- **TsgcHTTP1Client**: Klient HTTP 1.0 oparty na Indy TIdHTTP.
- **TsgcHTTP2Client**: Klient HTTP 2.0.
- **TsgcHTTP\_JWT\_Client**: Klient JWT (JSON WEB TOKEN).
- **TsgcHTTP\_JWT\_Server**: Serwer JWT (JSON WEB TOKEN).
- **TsgcHTTP\_OAuth2\_Client**: Klient OAuth 2.0.
- **TsgcHTTP\_OAuth2\_Server**: Serwer OAuth 2.0.
- **TsgcHTTPAWS\_SQS\_Client**: Klient Amazon AWS SQS.
- **TsgcHTTPGoogleCloud\_PubSub\_Client**: Klient Google Cloud Pub/Sub.
- **TsgcHTTPGoogleCloud\_Calendar\_Client**: Klient Google Calendar.
- **TsgcHTTP\_OAuth2\_Client\_Google**: Klient OAuth 2.0 dla Google.
- **TsgcHTTP\_OAuth2\_Client\_Microsoft**: Klient OAuth 2.0 dla Microsoft.
- **TsgcHTTP\_OAuth2\_Server\_Provider**: Dostawca serwera OAuth 2.0.
- **TsgcWSAPIServer\_WebAuthn**: Serwer WebAuthn do uwierzytelniania opartego na kluczach dostępu.
- **TsgcHTTPGoogleCloud\_FCM\_Client**: Klient Google Cloud FCM (Firebase Cloud Messaging).

## 6. sgcWebSockets IoT

- **TsgcIoTAmazon\_MQTT\_Client**: Klient Amazon MQTT IoT Core.
- **TsgcIoTAzure\_MQTT\_Client**: Klient Azure IoT MQTT.

## 7. sgcWebSockets P2P

- **TsgcUDPClient**: Klient UDP.
- **TsgcUDPServer**: Serwer UDP.
- **TsgcSTUNClient**: Klient STUN.
- **TsgcSTUNServer**: Serwer STUN.
- **TsgcTURNClient**: Klient STUN / TURN.
- **TsgcTURNServer**: Serwer STUN / TURN.
- **TsgcICEClient**: Klient ICE.
- **TsgcRTCPeerConnection**: Klient WebRTC Peer Connection.

## 8. sgcWebSockets DataSnap

- **TsgcWSHTTPWebBrokerBridgeServer:** Zastępstwo serwera DataSnap z obsługą HTTP + WebSockets.
- **TsgcWSHTTP2WebBrokerBridgeServer:** Zastępstwo serwera DataSnap z obsługą HTTP + HTTP/2 + WebSockets.
- **TsgcWSServer\_HTTPAPI\_WebBrokerBridge:** Zastępstwo serwera DataSnap oparte na serwerze Microsoft HTTP.SYS.

## 9. sgcWebSockets AI

- **TsgcAIOpenAIChatBot:** Tworzenie ChatBota z komendami głosowymi.
- **TsgcAIOpenAITranslator:** Tłumaczenie w czasie rzeczywistym.
- **TsgcAudioRecorderMCI:** Nagrywanie dźwięku przy użyciu MCI.
- **TsgcAudioPlayerMCI:** Odtwarzanie dźwięku przy użyciu MCI.
- **TsgcTextToSpeechSystem:** Zamiana tekstu na mowę przy użyciu domyślnego mechanizmu systemu operacyjnego.
- **TsgcTextToSpeechGoogle:** Zamiana tekstu na mowę przy użyciu Google Cloud.
- **TsgcTextToSpeechAmazon:** Zamiana tekstu na mowę przy użyciu Amazon AWS.
- **TsgcAIOpenAIEmbeddings:** umożliwia wykorzystanie własnych danych do tworzenia aplikacji AI.
- **TsgcAIDatabaseVectorFile:** przechowuje wektory w zwykłym pliku tekstowym.
- **TsgcAIDatabaseVectorPinecone:** obsługuje bazę danych wektorów Pinecone.
- **TsgcAIOpenAIAssistant:** Klient API Asystentów OpenAI.
- **TsgcWSAPIServer\_MCP:** Serwer MCP (Model Context Protocol).
- **TsgcWSAPIClient\_MCP:** Klient MCP (Model Context Protocol).

# Obsługa wersji

## Obsługiwane środowisko IDE Delphi

- Delphi 7 (\* obsługiwany tylko po uaktualnieniu do Indy 10; IntraWeb nie jest obsługiwany)
- Delphi 2007
- Delphi 2009
- Delphi 2010
- Delphi XE
- Delphi XE2
- Delphi XE3
- Delphi XE4
- Delphi XE5
- Delphi XE6
- Delphi XE7
- Delphi XE8
- Delphi 10 Seattle
- Delphi 10.1 Berlin
- Delphi 10.2 Tokyo
- Delphi 10.3 Rio
- Delphi 10.4 Sydney
- Delphi 11 Alexandria
- Delphi 12 Athens
- Delphi 13 Florence

## Obsługiwane IDE CBuilder

- CBuilder 2007
- CBuilder 2010
- CBuilder XE
- CBuilder XE2

- CBuilder XE3
- CBuilder XE4
- CBuilder XE5
- CBuilder XE6
- CBuilder XE7
- CBuilder XE8
- CBuilder 10 Seattle
- CBuilder 10.1 Berlin
- CBuilder 10.2 Tokyo
- CBuilder 10.3 Rio
- CBuilder 10.4 Sydney
- CBuilder 11 Alexandria
- CBuilder 12 Athens
- CBuilder 13 Florence

## **Obsługiwane IDE dla FreePascal**

- Lazarus

## **Wersja próbna**

Skompilowane pliki \*.dcu dostarczone z wersją bezpłatną używają domyślnych wersji Indy i IntraWeb. Jeśli którykolwiek z tych pakietów został zaktualizowany, prawdopodobnie nie będzie działał i konieczny będzie zakup wersji z pełnym kodem źródłowym.

## **Pakiet Indy**

Niektóre komponenty używają Indy jako biblioteki TCP/IP (np. TsgcWebsocketClient lub TsgcWebsocketServer), co oznacza, że Indy jest wymagany do zainstalowania pakietu sgcWebSockets. Domyślnie sgcWebSockets korzysta z biblioteki Indy wbudowanej w RAD Studio, jednak dostarczamy niestandardową wersję Indy oferującą więcej funkcji: obsługę OpenSSL API 1.1, OpenSSL 3.0, protokołu ALPN i innych.

# Instalacja

## Delphi / CBuilder / Lazarus

### 1. Wypakuj dołączone pliki do katalogu {\$DIR}

### 2. Z Delphi\CBuilder:

Dodaj katalog, do którego wypakowane są pliki {\$DIR}, do ścieżki biblioteki Delphi\CBuilder w menu Narzędzia, Opcje środowiska, Katalogi

### Wszystkie wersje Delphi\CBuilder

Dodaj katalog {\$DIR}\source do ścieżki biblioteki

### Dla określonej wersji Delphi

Delphi 7 : Dodaj katalog {\$DIR}\libD7 do ścieżki bibliotek

Delphi 2007 : Dodaj katalog {\$DIR}\libD2007 do ścieżki biblioteki

Delphi 2009 : Dodaj katalog {\$DIR}\libD2009 do ścieżki biblioteki

Delphi 2010 : Dodaj katalog {\$DIR}\libD2010 do ścieżki biblioteki

Delphi XE : Dodaj katalog {\$DIR}\libDXE do ścieżki biblioteki

Delphi XE2 : Dodaj katalog {\$DIR}\libDXE2\\$(Platform) do ścieżki biblioteki

Delphi XE3 : Dodaj katalog {\$DIR}\libDXE3\\$(Platform) do ścieżki bibliotek

Delphi XE4 : Dodaj katalog {\$DIR}\libDXE4\\$(Platform) do ścieżki biblioteki

Delphi XE5 : Dodaj katalog {\$DIR}\libDXE5\\$(Platform) do ścieżki bibliotek

Delphi XE6 : Dodaj katalog {\$DIR}\libDXE6\\$(Platform) do ścieżki bibliotek

Delphi XE7 : Dodaj katalog {\$DIR}\libDXE7\\$(Platform) do ścieżki biblioteki

Delphi XE8 : Dodaj katalog {\$DIR}\libDXE8\\$(Platform) do ścieżki biblioteki

Delphi 10 : Dodaj katalog {\$DIR}\libD10\\$(Platform) do ścieżki biblioteki

Delphi 10.1 : Dodaj katalog {\$DIR}\libD10\_1\\$(Platform) do ścieżki biblioteki

Delphi 10.2 : Dodaj katalog {\$DIR}\libD10\_2\\$(Platform) do ścieżki biblioteki

Delphi 10.3 : Dodaj katalog {\$DIR}\libD10\_3\\$(Platform) do ścieżki biblioteki

Delphi 10.4 : Dodaj katalog {\$DIR}\libD10\_4\\$(Platform) do ścieżki biblioteki

Delphi 11 : Dodaj katalog {\$DIR}\libD11\\$(Platform) do ścieżki biblioteki

Delphi 12 : Dodaj katalog {\$DIR}\libD12\\$(Platform) do ścieżki biblioteki

Delphi 13 : Dodaj katalog {\$DIR}\libD13\\$(Platform) do ścieżki biblioteki

### **Dla konkretnej wersji CBuilder**

C++ Builder 2010 : Dodaj katalog {\$DIR}\libD2010 do ścieżki biblioteki

C++ Builder XE : Dodaj katalog {\$DIR}\libDXE do ścieżki biblioteki

C++ Builder XE2 : Dodaj katalog {\$DIR}\libDXE2\\$(Platform) do ścieżki biblioteki

C++ Builder XE3 : Dodaj katalog {\$DIR}\libDXE3\\$(Platform) do ścieżki biblioteki

C++ Builder XE4 : Dodać katalog {\$DIR}\libDXE4\\$(Platform) do ścieżki biblioteki

C++ Builder XE5 : Dodaj katalog {\$DIR}\libDXE5\\$(Platform) do ścieżki bibliotek

C++ Builder XE6 : Dodaj katalog {\$DIR}\libDXE6\\$(Platform) do ścieżki biblioteki

C++ Builder XE7 : Dodaj katalog {\$DIR}\libDXE7\\$(Platform) do ścieżki biblioteki

C++ Builder XE8 : Dodaj katalog {\$DIR}\libDXE8\\$(Platform) do ścieżki biblioteki

C++ Builder 10 : Dodaj katalog {\$DIR}\libD10\\$(Platform) do ścieżki biblioteki

C++ Builder 10.1 : Dodaj katalog {\$DIR}\libD10\_1\\$(Platform) do ścieżki bibliotek

C++ Builder 10.2 : Dodaj katalog {\$DIR}\libD10\_2\\$(Platform) do ścieżki biblioteki

C++ Builder 10.3 : Dodaj katalog {\$DIR}\libD10\_3\\$(Platform) do ścieżki biblioteki

C++ Builder 10.4 : Dodaj katalog {\$DIR}\libD10\_4\\$(Platform) do ścieżki biblioteki

C++ Builder 11 : Dodaj katalog {\$DIR}\libD11\\$(Platform) do ścieżki bibliotek

C++ Builder 12 : Dodaj katalog {\$DIR}\libD12\\$(Platform) do ścieżki bibliotek

C++ Builder 13 : Dodaj katalog {\$DIR}\libD13\\$(Platform) do ścieżki biblioteki

Dla wszystkich wersji CBuilder dodaj dcp\\$(Platform) do ścieżki bibliotek (zawiera pliki .bpi)

### **3. Z Delphi**

#### **Wybierz**

Plik, Otwórz i przejdź do właściwego pliku Packages\sgcWebSockets.groupproj (najpierw skompiluj sgcWebSocketsX.dpk, a następnie zainstaluj dclsgcWebSocketsX.dpk)

#### **pliki pakietów dla Delphi**

sgcWebSocketsD7.groupproj : Delphi 7

sgcWebSocketsD2007.groupproj : Delphi 2007

sgcWebSocketsD2009.groupproj : Delphi 2009

sgcWebSocketsD2010.groupproj : Delphi 2010

sgcWebSocketsDXE.groupproj : Delphi XE

sgcWebSocketsDXE2.groupproj : Delphi XE2

sgcWebSocketsDXE3.groupproj : Delphi XE3

sgcWebSocketsDXE4.groupproj : Delphi XE4

sgcWebSocketsDXE5.groupproj : Delphi XE5

sgcWebSocketsDXE6.groupproj : Delphi XE6

sgcWebSocketsDXE7.groupproj : Delphi XE7

sgcWebSocketsDXE8.groupproj : Delphi XE8

sgcWebSocketsD10.groupproj : Delphi 10  
sgcWebSocketsD10\_1.groupproj : Delphi 10.1  
sgcWebSocketsD10\_2.groupproj : Delphi 10.2  
sgcWebSocketsD10\_3.groupproj : Delphi 10.3  
sgcWebSocketsD10\_4.groupproj : Delphi 10.4  
sgcWebSocketsD11.groupproj : Delphi 11  
sgcWebSocketsD12.groupproj : Delphi 12  
sgcWebSocketsD13.groupproj : Delphi 13

#### **4. Z C++Builder**

##### **Wybierz**

Plik, Otwórz i przejdź do właściwego pliku Packages\sgcWebSockets.groupproj (najpierw skompiluj sgcWebSocketsX.dpk, a następnie zainstaluj dclsgcWebSocketsX.dpk)

##### **pliki pakietów dla CBuilder**

sgcWebSocketsC2010.groupproj : C++ Builder 2010  
sgcWebSocketsCXE.groupproj : C++ Builder XE  
sgcWebSocketsCXE2.groupproj : C++ Builder XE2  
sgcWebSocketsCXE3.groupproj : C++ Builder XE3  
sgcWebSocketsCXE4.groupproj : C++ Builder XE4  
sgcWebSocketsCXE5.groupproj : C++ Builder XE5  
sgcWebSocketsCXE6.groupproj : C++ Builder XE6  
sgcWebSocketsCXE7.groupproj : C++ Builder XE7  
sgcWebSocketsCXE8.groupproj : C++ Builder XE8  
sgcWebSocketsC10.groupproj : C++ Builder 10  
sgcWebSocketsC10\_1.groupproj : C++ Builder 10.1



sgcWebSocketsC10\_2.groupproj : C++ Builder 10.2

sgcWebSocketsC10\_3.groupproj : C++ Builder 10.3

sgcWebSocketsC10\_4.groupproj : C++ Builder 10.4

sgcWebSocketsC11.groupproj : C++ Builder 11

sgcWebSocketsC12.groupproj : C++ Builder 12

sgcWebSocketsC13.groupproj : C++ Builder 13

## **5. Z poziomu Lazarus**

Wybierz: File, Open i przejdź do Packages\sgcWebSocketsLazarus.lpk (najpierw skompiluj, a następnie zainstaluj)

Skompilowane pliki znajdują się w katalogu Lazarus. Wewnątrz niego znajduje się katalog Indy z najnowszą wersją źródłową Indy.

Testowano z Lazarus 2.0.6 oraz Indy 10.5.9.4930

## **6. Dema**

Wszystkie dema są dostępne w podkatalogu Demos. Wystarczy otworzyć projekt i uruchomić go. Dema IntraWeb mogą wymagać modyfikacji niektórych jednostek ze względu na różne wersje IntraWeb.



# Install Setup

\*Wymaga systemu Windows Vista jako minimum (systemy Windows 2000, XP i Server 2003 nie są obsługiwane).

Jeśli do instalacji biblioteki sgcWebSockets używany jest Instalator Windows, proces instalacji jest prowadzony krok po kroku i jest bardzo prosty. W przypadku wystąpienia błędu podczas instalacji należy zapoznać się ze stroną [Błędy instalacji](#) lub spróbować [zainstalować pakiet ręcznie](#).

## Konfiguracja wersji próbnej

- Uruchom instalator próbny.
- Instalacja wersji Trial wymaga uprawnień administratora.
- Instalator wyświetli listę wersji Delphi / CBuilder / Rad Studio, a domyślnie pobrana wersja będzie włączona. Jeśli ta wersja NIE zostanie wykryta przez instalator, instalator wyodrębni pliki, ale nie spróbuje kompilować. Należy zapoznać się z instrukcją [ręcznej instalacji pakietu](#).



- Na następnej stronie wyświetlone są platformy dostępne do instalacji; włączone są tylko te wykryte przez instalator.



- Następna strona zawiera umowę licencyjną, którą należy zaakceptować w celu zainstalowania wersji próbnej.
- Po zaakceptowaniu umowy licencyjnej wyświetlane są komponenty, które zostaną zainstalowane. Domyślnie instalowane są: pakiet, skompilowane pliki DCU, demonstracje oraz pliki pomocy. Można dostosować, czy pliki pomocy i demonstracje mają być instalowane.



- Na koniec zostaną wyodrębnione pliki, skompilowany i zainstalowany pakiet oraz zarejestrowane wymagane ścieżki w IDE.

## Konfiguracja klientów

Użytkownicy posiadający licencję mogą zainstalować bibliotekę sgcWebSockets za pomocą instalatora. Poniżej znajdują się instrukcje krok po kroku dotyczące instalacji pakietu.

- Uruchom instalator.
- Instalator działa z najniższymi uprawnieniami (jeśli jest uruchamiany jako administrator, nie można go zainstalować na dyskach sieciowych). Jeśli miejsce docelowe wymaga uprawnień administratora, należy uruchomić instalator jako administrator.
- Najpierw należy ustawić nazwę użytkownika i hasło prywatnego konta eSeGeCe. Wystarczy to zrobić tylko raz; przy kolejnym użyciu instalatora odczyta on wcześniej zapisane wartości.



- Dostępne są 3 opcje:
  - **Użyj istniejącej licencji:** jeśli ta wersja była już instalowana, opcja zostanie domyślnie wybrana. Zostanie użyta ostatnia konfiguracja dla tej wersji.
  - **Nowa / aktualizacja licencji:** jeśli ta wersja nie była wcześniej instalowana, opcja zostanie domyślnie wybrana. Nastąpi połączenie z serwerem licencji w celu pobrania informacji o licencji. Jeśli licencja została niedawno zaktualizowana, można wybrać tę opcję, aby ją odświeżyć.
  - **Instalacja offline:** jeśli komputer nie ma dostępu do Internetu, należy wybrać tę opcję, aby aktywować licencję.

## Generowanie klucza

Ta opcja generuje klucz, który zostanie użyty do aktywacji licencji.



Skopiuj klucz i uzyskaj dostęp do prywatnego konta online: [www.esegece.com/my-account/subscriptions](http://www.esegece.com/my-account/subscriptions).



Wybierz subskrypcję do aktywacji i wklej klucz.



## Aktywacja licencji

Jeśli żądanie jest prawidłowe, zostanie zwrócona licencja, którą należy skopiować do instalatora.



- Jeśli licencja została pomyślnie aktywowana, należy wybrać, czy instalacja ma być przeprowadzona w środowisku Delphi, CBuilder czy Rad Studio IDE. Dostępna jest opcja wyodrębnienia wymaganych plików Lazarus (Lazarus wymaga ręcznej instalacji pakietu).



- Przy każdym użyciu instalatora dostępne są pewne opcje do dostosowania; należy nacisnąć przycisk **Options**, aby uzyskać dostęp do tych właściwości.
  - **Build Packages:** jeśli zaznaczono, instalator spróbuje skompilować pakiety.
  - **Register Paths IDE:** jeśli zaznaczono, instalator zarejestruje wymagane ścieżki bibliotek w środowisku IDE.
  - **Register BPLs IDE:** jeśli zaznaczono i instalator pomyślnie skompilował pakiety, zarejestruje pakiet czasu projektowania w środowisku IDE.
- Następujące opcje są dostępne tylko dla licencji z kodem źródłowym:
  - **Build IntraWeb:** jeśli zaznaczone, instalator zainstaluje wymagane pliki IntraWeb (domyślnie wyłączone). Obecnie instaluje wersję IntraWeb XVI.
  - **Build CBuilder Dproj:** jeśli zaznaczone, instalator zbuduje pakiet CBuilder przy użyciu pakietu Delphi sgcWebSockets i wygeneruje wszystkie wymagane pliki CBuilder.
  - **Use Custom Indy:** (tylko Enterprise), jeśli zaznaczone, sgcWebSockets będzie używać niestandardowej wersji Indy (z obsługą OpenSSL 1.1 i 3.0, TLS 1.3, ALPN...)
  - **sgcIndy Already Installed:** jeśli pakiet sgcIndy jest zainstalowany i ma być używany do kompilacji pakietu sgcWebSockets, należy zaznaczyć tę opcję.
  - **sgcIndy Compatibility Mode:** jeśli pakiet sgcIndy został zainstalowany w trybie zgodności (ponieważ inne pakiety używają Indy, jak DevExpress), należy zaznaczyć tę opcję.
  - **Always use of the following OpenSSL API Versions:** zaznacz tę opcję, jeśli chcesz wymusić użycie OpenSSL 1.1.1 lub OpenSSL 3.0.0 APIs
  - **Debug Mode:** zapisuje komunikaty debugowania w pliku dziennika. Nie należy używać tego trybu w środowisku produkcyjnym.
  - **Build Rad Studio IDE Win64:** buduje pakiet czasu projektowania dla Rad Studio IDE 12+ 64-bitowego.

- **Include Resources:** włączone domyślnie. Osadza pakiet klienta JavaScript (`sgcResources.RES`) w pakietach środowiska wykonawczego. Osadzony zasób jest potrzebny tylko wtedy, gdy serwer dostarcza klientowi przeglądarkowemu `sgcWebSockets` przez HTTP do zdalnych przeglądarek. Odznaczenie tej opcji powoduje usunięcie definicji `SGC_RESOURCES` w `sgcVer.inc` przed kompilacją pakietów, zmniejszając rozmiar każdego pliku wykonywalnego linkowanego ze źródłem. Bezpieczne do wyłączenia dla aplikacji klienckich Delphi / C++Builder i serwerów, które nie serwują dołączonego klienta JS. Patrz [Configure Install](#), aby poznać powiązaną dyrektywę kompilatora.



- Teraz można wybrać wersje IDE, które mają zostać zainstalowane. Dostępne będą wyłącznie te wersje IDE, które instalator wykryje jako zainstalowane.



- Następnym krokiem jest wybór platform.



- Wybierz folder, w którym zostanie zainstalowany pakiet. Przy ponownej instalacji pakietu instalator domyślnie wybierze ten sam folder, który był wybrany podczas poprzedniej instalacji.



- Wybierz komponenty do zainstalowania. Zarejestrowani klienci mają dostęp do **eksperta IDE**, który umożliwia połączenie z kontem eSeGeCe bezpośrednio z IDE, sprawdzanie dostępnych aktualizacji, bezpośredni dostęp do działu pomocy technicznej i wiele więcej.



- Na koniec zostaną wyodrębnione pliki, skompilowany i zainstalowany pakiet oraz zarejestrowane wymagane ścieżki w IDE.

## Błędy instalacji

- MsBuild zgłasza błąd, jeśli **ścieżka biblioteki jest zbyt długa**. Aby rozwiązać ten problem, należy usunąć nieużywane ścieżki ze ścieżki biblioteki. MsBuild ma ograniczenie wynoszące 32 000 znaków.
- błąd **F2110: 'tchar.h' file not found**. Ten błąd może pojawić się podczas kompilacji dla CBuilder; aby go naprawić, wystarczy otworzyć CBuilder, zamknąć go i spróbować ponownie zainstalować.

## Parametry wiersza poleceń instalatora

Instalator obsługuje następujące polecenia.

### **/SILENT**

Kreator i okno tła nie są wyświetlane, lecz postęp instalacji jest widoczny

### **/VERYSILENT**

Gdy instalacja jest bardzo cicha, to okno postępu instalacji nie jest wyświetlane.

### **/EXTRACT**

Pakiet nie jest instalowany, lecz tylko wypakowywany. Ścieżkę docelową można dostosować za pomocą parametru `/EXTRACT=ścieżka-do-folderu`

Należy użyć tego parametru wraz z `/SILENT`, jeśli wymagane jest tylko rozpakowanie plików bez interakcji z użytkownikiem.

### **/IDE**

Ten parametr pozwala określić, które środowisko IDE ma zostać zainstalowane. Należy ustawić jedną z następujących wartości:

- delphi
- cbuilder
- radstudio

Można dodatkowo dodać Lazarus.

Przykład: instalacja Delphi i Lazarus.

/ide=delphi-lazarus.

## **/VERSIONS**

Za pomocą tego parametru można określić, które wersje RAD Studio mają zostać zainstalowane. Dozwolone jest podanie wielu opcji:

- D7
- D2007
- D2009
- D2010
- DXE
- DXE2
- DXE3
- DXE4
- DXE5
- DXE6
- DXE7
- DXE8
- D10
- D10\_1
- D10\_2
- D10\_3
- D10\_4
- D11
- D12
- D13

Użyj wartości „All”, aby zainstalować wszystkie możliwe wersje.

Przykład: instalacja Delphi 10 i Delphi 13.

/versions=D10-D13

## **/PLATFORMS**

Przy użyciu tego parametru można określić, które platformy RAD Studio mają zostać zainstalowane. Dozwolone jest wybranie wielu opcji:



- Win32
- Win64
- Win64x
- Android
- Android64
- iOSDevice32
- iOSDevice64
- iOSSimulator
- iOSSimARM64
- OSX32
- OSX64
- OSXARM64
- Linux64

Należy użyć wartości "All", aby zainstalować wszystkie możliwe platformy.

Przykład: instalacja Win32 i Win64.

/platforms=Win32-Win64

**/USERNAME**

Ustawia nazwę użytkownika subskrypcji.

**/PASSWORD**

Ustawia hasło subskrypcji.

## **Ekspert IDE**

Jeśli ekspert IDE jest zainstalowany, dostępne są następujące opcje menu:



- **MyAccount:** bezpośredni dostęp do menu Pobierania (skąd można pobrać najnowszą wersję beta) oraz do sekcji Subskrypcje, w której można zarządzać licencją lub odnowić wygasłą licencję.
- **Wsparcie:** bezpośredni dostęp do centrum pomocy lub forum z automatycznym logowaniem. Dostępna jest również dokumentacja oraz formularz kontaktowy.
- **Opcje:** w tym menu można skonfigurować nazwę użytkownika i hasło do konta. Należy wybrać domyślną przeglądarkę i sprawdzić, czy są dostępne aktualizacje.

# Ręczna instalacja pakietu

Należy postępować zgodnie z kolejnymi krokami, aby zainstalować pakiet sgcWebSockets; zrzuty ekranu przedstawiają wersję Delphi 10.3.

## 1. Otwórz projekt grupy sgcWebSocketsD10\_3.



2. Teraz należy **najpierw skompilować pakiety runtime** (nazwa zaczyna się od sgcWebSockets). Dla każdej platformy docelowej istnieje jeden pakiet, co zależy od wersji Delphi, dlatego **należy wybierać platformę docelową jedna po drugiej i budować** każdy pakiet.

## 3. Wybierz win64 jako platformę docelową i skompiluj pakiet.



## 4. Wybierz Android jako platformę docelową i skompiluj pakiet.



## 5. Wybierz iOS Device 32 jako platformę docelową i skompiluj pakiet.



## 6. Wybierz iOS Device 64 jako platformę docelową i skompiluj pakiet.



## 7. Wybierz symulator urządzenia iOS jako platformę docelową i skompiluj pakiet.

☐

8. Wybierz **MacOS 32** jako platformę docelową i skompiluj pakiet.

☐

9. Wybierz **Win32** jako platformę docelową i skompiluj pakiet.

☐

10. Po skompilowaniu wszystkich pakietów runtime wybierz **pakiet design-time** (nazwa zaczyna się od dcl) i najpierw wykonaj operację **build**, a następnie **install** (pakiety design-time mają tylko Win32 jako platformę docelową).

☐☐

11. Jeśli instalacja zakończy się **pomyślnie**, zostanie wyświetlony komunikat z informacją o zainstalowaniu wszystkich komponentów.

☐

12. Następnie wystarczy dodać katalog, w którym znajdują się skompilowane pliki, do **ścieżki biblioteki Rad Studio**. Należy to zrobić dla każdej platformy docelowej (win32, win64, osx64...).

☐

\* Jeśli używane są serwery **Datasnap**, są one **NIE zawarte** w pakiecie sgcWebSockets, ponieważ nie mogą być instalowane; są to wyłącznie komponenty czasu wykonania. W takim przypadku do ścieżki biblioteki należy dodać również folder Source.



# Błędy instalacji

Podczas instalacji komponentów mogą pojawić się pewne błędy.

## Nie znaleziono pakietu Intraweb

sgcWebSockets jest kompilowany przy użyciu **domyślnej wersji Intraweb** dostarczonej z Delphi. Jeśli Intraweb nie jest zainstalowany, można **zmodyfikować plik sgcVer.inc** (znajdujący się w folderze Source).

Znajdź swoją wersję Delphi i **skomentuj wszystkie dyrektywy kompilatora dla Intraweb** (zaczynające się od IW). **Przykład:** dla Delphi 10.4 skomentuj wszystkie dyrektywy kompilatora dla Intraweb.

```
{ $IFDEF VER340 } { Delphi 10.4 }
```

```
{ $DEFINE D2006 }
```

```
{ $DEFINE D2007 }
```

```
{ $DEFINE D2009 }
```

```
{ $DEFINE D2010 }
```

```
{ $DEFINE DXE }
```

```
{ $DEFINE DXE2 }
```

```
{ $DEFINE DXE3 }
```

```
{ $DEFINE DXE4 }
```

```
{ $DEFINE DXE5 }
```

```
{ $DEFINE DXE6 }
```

```
{ $DEFINE DXE7 }
```

```
{ $DEFINE DXE8 }
```

```
{ $DEFINE D10 }
```

```
{ $DEFINE D10_1 }
```

```
{ $DEFINE D10_2 }
```

```
{ $DEFINE D10_3 }
```

```
{ $DEFINE D10_4 }
```

```
{ $DEFINE INDY10_1 }
```

```
{ $DEFINE INDY10_2}  
  
{ $DEFINE INDY10_5_5}  
  
{ $DEFINE INDY10_5_7}  
  
{ $DEFINE INDY10_5_8}  
  
{ $DEFINE INDY10_6}  
  
{ $DEFINE INDY10_6_0_5122}  
  
{ $DEFINE INDY10_6_0_5169}  
  
{ $DEFINE INDY10_6_2_5263}  
  
{ $DEFINE INDY10_6_2_5366}  
  
{ $DEFINE INDY10_6_2_D10_4}
```

```
  
  
{ $IFDEF BCB}  
  
{ $IFDEF MACOS}  
  
{ $IFDEF ANDROID}  
  
{ . $DEFINE IWIX}  
  
{ . $DEFINE IWXI}  
  
{ . $DEFINE IWXIV}  
  
{ . $DEFINE IWXV}  
  
{ $ENDIF}  
  
{ $ENDIF}  
  
{ $IFDEF NEXTGEN}  
  
{ $DEFINE SGC_JSON_INTF}  
  
{ $ENDIF}  
  
{ $ENDIF}  
  
{ $ENDIF}
```

Jeśli IntraWeb jest zainstalowany, ale jest to inna wersja niż domyślna dostarczana z Delphi, pakiet IntraWeb może mieć inną nazwę. W takim przypadku należy otworzyć pakiet uruchomieniowy sgcWebSockets i **zmienić nazwę IntraWeb** w źródle projektu.



## Nie znaleziono pakietu Indy

**sgcWebSockets** wymaga **biblioteki Indy** do instalacji komponentów w IDE. Instalacja wersji próbnej jest skompilowana z biblioteką Indy dostarczoną z Delphi / CBuilder, więc jeśli pojawi się komunikat podobny do poniższego:

[DCC Fatal Error] dclsgcWebSocketsDX.dpk(31): E2202 Required package 'IndyCore' not found

Prawdopodobnie zainstalowana jest **nowsza wersja Indy**, dlatego w celu zainstalowania wersji próbnej należy usunąć tę wersję i zainstalować wbudowaną wersję Indy przy użyciu instalatora Delphi / CBuilder.

Jeśli dostępny jest pełny kod źródłowy, wystarczy sprawdzić:

1. Wymagane pakiety Indy: IndyCore, IndySystem i IndyProtocols. W przypadku nowszej wersji Indy pakiety mogą mieć inną nazwę (z numerem wersji). Należy przejść do menu "**Component / Install Packages**" i sprawdzić nazwy pakietów Indy, a następnie odpowiednio je zmienić w projekcie.

2. sgcWebSockets obsługuje kilka wersji Indy; dostępne są dyrektywy kompilatora umożliwiające kompilację dla każdej wersji Indy. Należy otworzyć plik **sgcVer.inc** znajdujący się w folderze źródłowym i dokonać odpowiednich zmian dla swojej wersji Indy (wartość `gsIdVersion` z pliku `IdVers.inc` biblioteki Indy). Przykładowe dyrektywy kompilatora:

```
{ $DEFINE INDY10_1 }
```

```
{ $DEFINE INDY10_2 }
```

```
{ $DEFINE INDY10_5_5 }
```

```
{ $DEFINE INDY10_5_7 }
```

```
{ $DEFINE INDY10_5_8 }
```

```
{ $DEFINE INDY10_6 }
```

```
{ $DEFINE INDY10_6_0_5122 }
```

```
{ $DEFINE INDY10_6_0_5169 }
```



```
{ $DEFINE INDY10_6_2_5263 }
```

```
{ $DEFINE INDY10_6_2_5366 }
```

```
{ $DEFINE INDY10_6_2_D10_4 }
```

## **c00000005 ACCESS\_VIOLATION w CBuilder**

Jeśli podczas kompilacji projektu w CBuilder pojawi się ten błąd, należy ustawić następujące opcje w projekcie:

Project > Options > C++ Linker

odznaczyć "Link with Dynamic RTL"

Projekt > Opcje > Pakiety > Pakiety środowiska wykonawczego

odznaczyć „Link with runtime packages"

## **Nie można znaleźć importu pakietu: sgcWebSocketsCXXX.bpi w CBuilder Win64**

Podczas kompilowania pakietu runtime dla win64 należy skompilować zarówno wersję Release, jak i Debug.

## **Niejednoznaczne odwołanie System.ZLib.hpp i IdZLib.hpp CBuilder**

sgcWebSockets Standard i Professional używają Indy w niektórych komponentach, a Indy nie korzysta z jednostki ZLib, lecz z własnej kopii: IdZLib, IdZLibHeaders... Projekt jest linkowany z jednostkami ZLib i Indy ZLib, więc podczas kompilacji kompilator nie wie, które odwołanie jest poprawne, ponieważ nazwy są takie same. Istnieją 2 rozwiązania:

1. Wyszukaj, gdzie dołączono link do System.ZLib.hpp i usuń go lub przenieś za IdZLibHeaders.hpp.
2. Użyj poniższych definicji warunkowych NO\_USING\_NAMESPACE\_SYSTEM\_ZLIB lub DELPHIHEADER\_NO\_IMPLICIT\_NAMESPACE\_USE w opcjach projektu, aby uniknąć stosowania przestrzeni nazw System.Zlib.hpp.

## Niejednoznaczne odwołanie System.ZLib.hpp i sgcdZLib.hpp CBuilder

sgcWebSockets Enterprise używa niestandardowej wersji Indy dla niektórych komponentów, a Indy nie korzysta z jednostki ZLib — używa własnej kopii ZLib: sgcdZLib, sgcdZLibHeaders. Projekt odwołuje się zarówno do ZLib, jak i do jednostek ZLib Indy, więc podczas kompilacji kompilator nie wie, która z nich jest właściwa, ponieważ nazwy są identyczne. Istnieją 2 rozwiązania:

1. Znajdź miejsce, w którym dołączone jest łącze do System.ZLib.hpp, i usuń je lub przenieś za sgcdZLibHeaders.hpp.
2. Użyj poniższych definicji warunkowych NO\_USING\_NAMESPACE\_SYSTEM\_ZLIB lub DELPHIHEADER\_NO\_IMPLICIT\_NAMESPACE\_USE w opcjach projektu, aby uniknąć stosowania przestrzeni nazw System.Zlib.hpp.

## Undefined reference to vTable for Sgcwebsocket... w CBuilder i Android

Aby naprawić błąd, należy zastosować następujące obejście: dodaj plik libsgcwebsocketsC\*.a, który znajduje się w domyślnym folderze dcp/android, do projektu za pomocą menu „Project / Add to Project”.

Przykład: dla CBuilder 11 należy dodać do projektu plik "libsgcWebSocketsC11.a", który domyślnie znajduje się w folderze "C:\Users\Public\Documents\Embarcadero\Studio\22.0\DCP\Android\Release”.

## Zmiana sumy kontrolnej w Lazarus

Błąd ten może wystąpić podczas próby instalacji składników w środowisku Lazarus, jeśli profil budowania środowiska IDE nie jest ustawiony na „Optimized IDE”. Wersja próbna jest kompilowana z profilem „Optimized IDE”.

## Nie można odnaleźć X używanego przez Y, niezgodny ppu

Wypróbuj następujące obejście: „Run / Clean up and rebuild” z opcji menu.

# Konfiguracja instalacji

W folderze źródłowym znajduje się plik sgcVer.inc zawierający wszystkie dyrektywy kompilatora dla wszystkich środowisk IDE Delphi, CBuilder i Lazarus.

W tym miejscu można dostosować konfigurację dla IntraWeb, Indy... **zazwyczaj nie ma potrzeby wprowadzania żadnych zmian**, chyba że chce się włączyć lub wyłączyć niektóre funkcje.

**Należy ostrożnie modyfikować definicje kompilatora i skontaktować się z nami w razie potrzeby uzyskania pomocy.**

Dla każdej wersji Delphi istnieje sekcja, w której można skonfigurować wszystkie definicje kompilatora; poniżej przykład dla Delphi 10.4

```
{ $IFDEF VER340 } { Delphi 10.4 }
```

```
{ $DEFINE D2006 }
```

```
{ $DEFINE D2007 }
```

```
{ $DEFINE D2009 }
```

```
{ $DEFINE D2010 }
```

```
{ $DEFINE DXE }
```

```
{ $DEFINE DXE2 }
```

```
{ $DEFINE DXE3 }
```

```
{ $DEFINE DXE4 }
```

```
{ $DEFINE DXE5 }
```

```
{ $DEFINE DXE6 }
```

```
{ $DEFINE DXE7 }
```

```
{ $DEFINE DXE8 }
```

```
{ $DEFINE D10 }
```

```
{ $DEFINE D10_1 }
```

```
{ $DEFINE D10_2 }
```

```
{ $DEFINE D10_3 }
```

```
{ $DEFINE D10_4}

{ $DEFINE INDY10_1}

{ $DEFINE INDY10_2}

{ $DEFINE INDY10_5_5}

{ $DEFINE INDY10_5_7}

{ $DEFINE INDY10_5_8}

{ $DEFINE INDY10_5_9}

{ $DEFINE INDY10_6}

{ $DEFINE INDY10_6_0_5122}

{ $DEFINE INDY10_6_0_5169}

{ $DEFINE INDY10_6_2_5263}

{ $DEFINE INDY10_6_2_5366}

{ $DEFINE INDY10_6_2_D10_4}
```

```
{ $IFDEF BCB}

{ $IFDEF MACOS}

{ $IFDEF ANDROID}

{. $DEFINE IWIX}

{. $DEFINE IWXI}

{. $DEFINE IWXIV}

{. $DEFINE IWXV}

{ $ENDIF}

{ $ENDIF}

{ $IFDEF NEXTGEN}

{ $DEFINE SGC_JSON_INTF}

{ $ENDIF}

{ $ENDIF}

{ $ENDIF}
```

## Indy

Istnieje kilka dyrektyw kompilatora dla biblioteki Indy. Zależy to od zainstalowanej wersji Indy; domyślnie skonfigurowane jest dla pakietu Indy dołączonego do Delphi. Wersja Indy to parametr `gsIdVersion` w pliku `IdVers.inc` biblioteki Indy.

## Intraweb

Jeśli Intraweb nie jest zainstalowany, należy zakomentować dyrektywy kompilatora dla Intraweb (te zaczynające się od `IW...`).

## SGC\_RESOURCES

Domyślnie plik `sgcVer.inc` deklaruje `{ $DEFINE SGC_RESOURCES }`, co powoduje, że pakiety runtime osadzają plik zasobów `sgcResources.RES`. Dodaje to dodatkowe bajty do każdego pliku wykonywalnego linkowanego statycznie ze źródłem `sgcWebSockets`.

Zasób zawiera spakowany kod JavaScript klienta, który serwer `sgcWebSockets` może dostarczyć przez HTTP do zdalnej przeglądarki (aby przeglądarka mogła obsługiwać subprotokół `sgc`). Jest potrzebny wyłącznie gdy:

- Używany jest serwer HTTP/WebSocket `sgcWebSockets`, który dostarcza pakiet klienta JS do przeglądarek (typowe dla serwerów opartych na `TsgcWSProtocol_JS_*`).

**Nie** jest wymagane w przypadku:

- Klienty WebSocket w Delphi / C++Builder (dane są przetwarzane w kodzie natywnym, nie w przeglądarce).
- Serwery, które nie serwują dołączonego klienta JS (np. serwery wyłącznie WebSocket, serwery native-to-native).

Aby wykluczyć osadzony zasób, należy zakomentować odpowiedni wiersz w pliku `sgcVer.inc`:

```
{ $DEFINE SGC_RESOURCES } { RESOURCES }
```

Alternatywnie, podczas uruchamiania [instalatora wersji źródłowej](#), należy odznaczyć opcję **Include Resources** na stronie Opcje; instalator przepisze plik `sgcVer.inc` przed skompilowaniem pakietów.

# Instalacja pakietu sgcIndy

## Konfiguracja instalacji

\*Wymaga systemu Windows Vista jako minimum (systemy Windows 2000, XP i Server 2003 nie są obsługiwane).

Użytkownicy, którzy zakupili licencję, mogą zainstalować pakiet sgcIndy przy użyciu instalatora. Poniżej znajdują się instrukcje krok po kroku dotyczące instalacji pakietu.

- Uruchom instalator.
- Najpierw należy ustawić nazwę użytkownika i hasło prywatnego konta eSeGeCe. Dane te należy wprowadzić tylko raz, przy następnym uruchomieniu instalatora zostanie odczytana ostatnio zapisana wartość.



- Jeśli użytkownik zalogował się pomyślnie, należy wybrać, czy instalacja ma dotyczyć Delphi, CBuilder czy Rad Studio IDE.



- Istnieje kilka opcji, które można dostosować przy każdym użyciu instalatora. Naciśnij przycisk **Options**, aby uzyskać dostęp do tych właściwości.
  - **Build Packages:** jeśli zaznaczone, instalator spróbuje zbudować pakiety.
  - **Register Paths IDE:** jeśli zaznaczone, instalator zarejestruje wymagane ścieżki bibliotek w IDE.
  - **Register BPLs IDE:** jeśli zaznaczone i instalator pomyślnie zbudował pakiety, zarejestruje pakiet czasu projektowania w IDE.
  - **Remove Default Indy Version:** jeśli zaznaczone, instalator najpierw odinstaluje standardową wersję Indy dostarczoną z Rad Studio.
  - **Restore Default Indy Version when Uninstalling:** jeśli zaznaczone, instalator przywróci odinstalowaną standardową wersję Indy podczas deinstalacji pakietu.
  - **Compatibility Mode:** jeśli zaznaczone, pliki dcp są kompilowane bez wersji i kopiowane do folderu Embarcadero/lib. Zaznacz tę opcję, jeśli inne pakiety korzystają z pakietów Indy, np. DevExpress. Dodatkowo kompiluje jednostki Embarcadero IP Abstraction z zainstalowaną wersją sgcIndy.
  - **Always use of the following OpenSSL API Versions:** zaznacz tę opcję, jeśli chcesz wymusić użycie API OpenSSL 1.1.1 lub OpenSSL 3.0.0



- Teraz można wybrać wersje IDE, które mają zostać zainstalowane. Dostępne będą tylko te wersje IDE, które instalator wykryje jako zainstalowane.



- Następnym krokiem jest wybór platform.



- Wybierz folder, w którym zostanie zainstalowany pakiet. Przy ponownej instalacji pakietu instalator domyślnie wybierze ten sam folder, który był wybrany podczas poprzedniej instalacji.



- Wybierz komponenty do zainstalowania.



- Na koniec zostaną wyodrębnione pliki, skompilowany i zainstalowany pakiet oraz zarejestrowane wymagane ścieżki w IDE.

## Błędy instalacji

- MsBuild zgłasza błąd, jeśli **ścieżka biblioteki jest zbyt długa**. Aby rozwiązać ten problem, należy usunąć nieużywane ścieżki ze ścieżki biblioteki. MsBuild ma ograniczenie wynoszące 32 000 znaków.

## Instalacja ręczna

Jeśli Indy jest **już zainstalowane**, należy je najpierw **odinstalować**.

1. **Usuń prekompilowane pliki BPL** — dclIndyCoreX.bpl i dclIndyProtocolsX.bpl — ze środowiska IDE za pomocą okna dialogowego „Komponenty > Zainstaluj pakiety”.
2. Następnie należy **usunąć wszystkie istniejące pliki binarne** (IndySystemX., IndyCoreX., IndyProtocolsX., dclIndyCoreX. i dclIndyProtocolsX.\*) oraz wszelkie pliki źródłowe Indy 10, jeśli są obecne.
3. Należy **sprawdzić pliki** w folderach \bin, \lib i \source IDE, podfolderach \Indy oraz folderach systemowych systemu operacyjnego.

Aby **zbudować pakiet sgcIndy**, można:

1. **(Tylko CBuilder)** Należy użyć skryptu wiersza poleceń **FULLC#.BAT** odpowiadającego używanej wersji CBuilder.
2. **Otwórz poszczególne pliki DPK w IDE i skompiluj je w następującej kolejności:**
  1. IndySystemX.dpk (w Lib\System)
  2. IndyCoreX.dpk (w Lib\Core)
  3. IndyProtocolsX.dpk (w Lib\Protocols)
  4. dclIndyCoreX.dpk (w Lib\Core)
  5. dclIndyProtocolsX.dpk (w Lib\Protocols)

Po zbudowaniu pakietów Indy przejdź do menu **Components / Install Packages** i zainstaluj pakiety projektowe Indy.

1. dclIndyCore\*.bpl
2. dclIndyProtocols\*.bpl

Na koniec należy ustawić ścieżki do pakietów sgcIndy w IDE.



# Konfiguruj ZLib

**Wersja ZLib: 1.2.12**

sgcWebSockets używa kompresji ZLib, gdy włączone jest rozszerzenie WebSocket [Compression PerMessage Deflate](#). Domyślnie biblioteka ZLib jest statycznie dołączona do aplikacji, więc nie ma potrzeby wdrażania jej osobno.

Jeśli chcesz użyć określonej biblioteki, dodaj następującą dyrektywę warunkową do swojego projektu:

```
SGC_DYNAMICLOAD_ZLIB
```

Alternatywnie można edytować plik sgcIndy.inc (znajdujący się w folderze źródłowym) i dodać następującą linię

```
{ $DEFINE SGC_DYNAMICLOAD_ZLIB }
```

Na koniec należy ustawić lokalizację biblioteki ZLib, korzystając z poniższej metody i podając pełną ścieżkę (bez nazwy biblioteki) do jej lokalizacji:

```
sgcIdZLibHeaders.IdZLibSetLibPath('c:\software\zlib');
```

*\*Ta konfiguracja jest dostępna wyłącznie dla sgcWebSockets Enterprise Edition z kodem źródłowym.*

# Szybki start

## Komponenty WebSockets

Tworzenie nowego serwera WebSocket lub klienta WebSocket jest bardzo proste. Wystarczy utworzyć nową instancję klasy, skonfigurować Host/Port i ustawić właściwość `Active = true`, aby uruchomić proces.

[Szybki start WebSockets](#)

## Komponenty HTTP

Protokół HTTP/2 umożliwia tworzenie znacznie szybszych serwerów i klientów HTTP niż przy użyciu protokołu HTTP/1. Serwer HTTP/2 jest wbudowany w serwer WebSocket, natomiast klient HTTP/2 to dedykowany komponent implementujący protokół HTTP/2.

[Szybki start HTTP](#)

## Przepływ wątków

Komponenty `sgcWebSockets` są wielowątkowe, co oznacza, że **połączenia działają w wątkach pomocniczych**. Domyślnie, główne zdarzenia są wywoływane w wątku głównym. Jest to przydatne, gdy liczba zdarzeń do wywołania jest mała, ale dla **lepszey wydajności** można skonfigurować komponenty tak, aby **zdarzenia były wywoływane w kontekście wątku połączenia**. Przeczytaj następujący artykuł wyjaśniający, jak skonfigurować przepływ wątków:

[Jak skonfigurować NotifyEvents](#)

## Jak budować aplikacje

Tworzenie aplikacji przy użyciu biblioteki `sgcWebSockets` jest bardzo proste. Wystarczy postępować zgodnie z poniższymi wskazówkami, które pomogą **pomyślnie zbudować aplikację**.

[Build](#)

## Serwer o wysokiej wydajności

sgcWebSockets posiada **2 implementacje serwera**: jedną opartą na **serwerze Indy** i drugą opartą na **serwerze Microsoft HTTP.SYS**. Drugi jest zalecany dla serwerów o wysokiej wydajności wymagających obsługi tysięcy równoczesnych połączeń. Zapoznaj się z poniższym artykułem wyjaśniającym, jak poprawić wydajność serwera.

[Serwer o wysokiej wydajności](#)

## Menedżer pamięci

Wybór odpowiedniego menedżera pamięci może poprawić wydajność aplikacji. Zapoznaj się z poniższym artykułem, który przedstawia porównanie kilku menedżerów pamięci.

[Menedżer pamięci](#)

## OpenSSL

Gdy aplikacja wymaga bezpiecznych połączeń, zazwyczaj wymagane są **biblioteki openssl** do **szyfrowania komunikacji**. Należy postępować zgodnie z poniższymi krokami, aby pomyślnie skonfigurować aplikację z bibliotekami openssl.

[Konfiguracja OpenSSL](#)

## Indy

Biblioteka Indy stanowi podstawę niektórych składników sgcWebSockets; edycja Enterprise sgcWebSockets zawiera niestandardową wersję Indy umożliwiającą korzystanie z openssl 1.1.1 i openssl 3.0.0, ALPN itd.

[Indy](#)

## Linux (Lazarus)

Jeśli podczas kompilowania projektu Lazarus dla systemu Linux pojawi się następujący komunikat:

```
Inicjalizacja semafora nie powiodła się (możliwe zbyt wiele równo-  
ległych wątków)
```

Wystarczy dodać jednostkę **cthreads** do pliku projektu.

# QuickStart | WebSockets

Zacznijmy od podstawowego przykładu, w którym należy utworzyć serwer WebSocket oraz dwa typy klientów WebSocket: klienta aplikacyjnego i klienta przeglądarki internetowej.

## Serwer WebSocket

1. Utwórz nową aplikację Windows Forms
2. Upuść komponent TsgcWebSocketServer na formularz.
3. Na zakładce Zdarzenia kliknij dwukrotnie zdarzenie OnMessage i wpisz następujący kod:

```
void OnMessage(TsgcWSConnection *Connection, const string Text)
{
    ShowMessage("Message Received From Client: " + Text);
}
```

4. Upuść przycisk na formularz, kliknij go dwukrotnie i wpisz następujący kod:

```
TsgcWebSocketServer1->Active = true;
```

## Klient WebSocket

1. Utwórz nową aplikację Windows Forms
2. Upuść komponent TsgcWebSocketClient na formularz i skonfiguruj właściwości Host i Port, aby połączyć się z serwerem.
3. Upuść TButton na formularzu, kliknij dwukrotnie i wpisz ten kod:

```
TsgcWebSocketClient1->Active = true;
```

4. Upuść przycisk na formularz, kliknij go dwukrotnie i wpisz następujący kod:

```
TsgcWebSocketClient1->WriteData("Hello Server From VCL Client");
```

## Klient przeglądarki internetowej

1. Utwórz nowy plik HTML
2. Otwórz plik w edytorze tekstu i skopiuj następujący kod:

```
<html>
<head>
<script type="text/javascript" src="http://host:port/sgcWebSockets.js"></script>
</head>
<body>
<a href="javascript:var socket = new sgcWebSocket('ws://host:port');">Open</a>
<a href="javascript:socket.send('Hello Server From Web Browser');">Send</a>
</body>
</html>
```

Należy zastąpić host i port w tym pliku własnym hostem i portem!

3. Zapisz plik i to wszystko — skonfigurowano podstawowego klienta WebSocket dla przeglądarki internetowej.

## Sposób użycia

1. Uruchom aplikację serwera i naciśnij przycisk, aby uruchomić serwer WebSocket nasłuchujący nowych połączeń.
2. Uruchom aplikację klienta i naciśnij button1, aby połączyć się z serwerem, a następnie naciśnij button2, aby wysłać wiadomość. Po stronie serwera zostanie wyświetlona wiadomość z tekstem wysłanym przez klienta.
3. Otwórz plik HTML w przeglądarce internetowej (Chrome, Firefox, Safari lub Internet Explorer 10+), naciśnij Open, aby otworzyć połączenie, a następnie naciśnij Send, aby wysłać wiadomość do serwera. Po stronie serwera zostanie wyświetlona wiadomość z tekstem wysłanym przez klienta w przeglądarce.

## Kompilator Linux

Prosty przykład serwera (nasłuchującego na porcie 5000).

```

program sgcWebSockets_linux;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils, sgcWebSocket, sgcWebSocket_Types;

var
  oServer: TsgcWebSocketServer;

begin
  try
    oServer := TsgcWebSocketServer.Create(nil);
    oServer.Port := 5000;
    oServer.NotifyEvents := neNoSync;
    oServer.Active := True;

    while oServer.Active do
      Sleep(10);
    except
      on E: Exception do
        Writeln(E.ClassName, ': ', E.Message);
      end;
    end.
end.

```

## Linux (Lazarus)

Jeśli podczas kompilowania projektu Lazarus dla systemu Linux pojawi się następujący komunikat:

```

Inicjalizacja semafora nie powiodła się (możliwe zbyt wiele równo-
ległych wątków)

```

Wystarczy dodać jednostkę **cthreads** do pliku projektu.

# QuickStart | HTTP

Zacznijmy od podstawowego przykładu, w którym należy utworzyć serwer HTTP/2 oraz klienta HTTP/2.

## Serwer HTTP/2

1. Utwórz nową aplikację Windows Forms
2. Umieść komponent TsgcWebSocketHTTPServer na formularzu.
3. Na karcie Events kliknij dwukrotnie zdarzenie OnCommandGet i wpisz następujący kod:

```
void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo,
    TidHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/")
    {
        AResponseInfo->ContentText = "<html><head><title>Test Page</title></head><body></body></html>";
        AResponseInfo->ContentType = "text/html";
        AResponseInfo->ResponseNo = 200;
    }
}
```

4. Domyślnie serwer obsługuje wyłącznie połączenia HTTP/1, dlatego należy włączyć opcje HTTP/2 we właściwości **HTTP2Options.Enabled = true**, a następnie skonfigurować opcje SSL. Połączenia szyfrowane wymagają [bibliotek OpenSSL](#).

```
TsgcWebSocketHTTPServer1->Port = 443;
TsgcWebSocketHTTPServer1->SSL = true;
TsgcWebSocketHTTPServer1->SSLOptions->CertFile = "server cert file";
TsgcWebSocketHTTPServer1->SSLOptions->KeyFile = "server private key file";
TsgcWebSocketHTTPServer1->SSLOptions->RootCertFile = "server root cert file";
TsgcWebSocketHTTPServer1->SSLOptions->OpenSSL_Options->APIVersion = oslAPI_1_1;
TsgcWebSocketHTTPServer1->SSLOptions->Port = 443;
TsgcWebSocketHTTPServer1->SSLOptions->Version = tls1_3;
```

5. Upuść przycisk na formularz, kliknij go dwukrotnie i wpisz następujący kod:

```
TsgcWebSocketHTTPServer1->Active = true;
```

## Klient HTTP/1

1. Utwórz nową aplikację Windows Forms
2. Upuść komponent TButton na formularz, kliknij dwukrotnie i wpisz następujący kod:



```
TsgcHTTP1Client *oHTTP1 = new TsgcHTTP1Client();
try
{
    ShowMessage(oHTTP1->Get("https://127.0.0.1"));
}
__finally
{
    oHTTP1->Free();
}
```

## Klient HTTP/2

1. Utwórz nową aplikację Windows Forms
2. Upuść komponent TButton na formularz, kliknij dwukrotnie i wpisz następujący kod:

```
TsgcHTTP2Client *oHTTP2 = new TsgcHTTP2Client();
try
{
    ShowMessage(oHTTP2->Get("https://127.0.0.1"));
}
__finally
{
    oHTTP2->Free();
}
```

## QuickStart | Przepływ wątków

Składniki `sgcWebSockets` są wielowątkowe. Przykładowo **`TsgcWebSocketHTTPServer`** (oparty na bibliotece `Indy`) tworzy jeden wątek dla każdego połączenia, natomiast **`TsgcWebSocketServer_HTTPAPI`** (oparty na `Microsoft HTTP.SYS`) korzysta z puli wątków obsługującej połączenia (maksymalnie 64 wątki). Z kolei **`TsgcWebSocketClient`** działa we własnym wątku, asynchronicznie przetwarzając odpowiedzi z serwera `WebSocket`.

Domyślnie istnieje właściwość o nazwie **`NotifyEvents`** z wartością `neAsynchronous`. Oznacza to, że gdy klient `WebSocket` odbiera wiadomość, jest ona kolejkowana i wysyłana do wątku głównego przez system operacyjny w późniejszym czasie. Działa to dobrze w przypadku klientów, którzy nie odbierają dużej liczby wiadomości i jest wygodne w użyciu, ponieważ nie wymaga synchronizacji z wątkiem głównym w przypadku na przykład aktualizacji kontrolki formularza.

Gdy jednak serwer lub klient musi przetwarzać wiele wiadomości w krótkim czasie, lepiej zmienić przepływ wątkowy na taki, w którym zdarzenia są wywoływane w kontekście wątku połączenia. W tym celu należy ustawić właściwość `NotifyEvents` na **`neNoSync`**. W takim przypadku, gdy klient odbierze wiadomość z serwera, zostanie ona przetworzona w kontekście wątku pomocniczego — jeśli wymagana jest aktualizacja kontrolki formularza, należy najpierw zsynchronizować się z wątkiem głównym, a następnie zaktualizować kontrolkę (ponieważ kontrolki formularza nie są bezpieczne wątkowo). To samo dotyczy dostępu do obiektów współdzielonych — należy zaimplementować własne metody synchronizacji.

### Przepływ wątków w trybie łatwym (`NotifyEvents = neAsynchronous`) przy niskiej wydajności

Jest to domyślny przepływ wątków, stosowany zazwyczaj w przykładowych aplikacjach demonstracyjnych. Należy wybrać ten tryb, gdy nie przewiduje się obsługi wielu wiadomości na sekundę oraz gdy wymagana jest aktualizacja kontrolki formularza lub dostęp do obiektów współdzielonych.

`NotifyEvents = neAsynchronous`

### Przepływ wątkowania — najlepsza wydajność (`NotifyEvents = neNoSync`)

Należy ustawić ten przepływ wątkowy dla komponentów serwera oraz dla klientów wymagających wysokiej wydajności, gdy spodziewane jest przetwarzanie wielu wiadomości. W tej konfiguracji zdarzenia są

wywoływane w kontekście wątku połączenia, więc aby zaktualizować kontrolkę formularza, należy najpierw zsynchronizować się z wątkiem głównym.

NotifyEvents = neNoSync

## **Jak synchronizować wątek główny**

Można synchronizować z wątkiem głównym wywołując TThread.Synchronize lub TThread.Queue, obie metody mogą być używane, a wybór jednej lub drugiej zależy od sposobu implementacji synchronizacji.

### TThread.Synchronize

Ta metoda jest blokująca, co oznacza, że podczas wywołania Synchronize kod blokuje się do momentu synchronizacji z wątkiem głównym.

### TThread.Queue

Ta metoda jest nieblokująca, więc po wywołaniu kolejki wiadomość jest umieszczana w kolejce i zostanie wysłana później.

## **Przykładowy kod**

Aktualizuj komponent Memo wiadomościami odbieranymi od klienta WebSocket.

```
void OnClientMessage(TsgcWSConnection *Connection, string aText)
{
    TThread::Queue(NULL,
        [&]() {
            memo1->lines->add(aText);
        });
}
```

## QuickStart | Kompilacja

Budowanie aplikacji z użyciem biblioteki sgcWebSockets jest bardzo proste. Należy jedynie pamiętać, czy używane komponenty wymagają bibliotek OpenSSL. Jeśli aplikacja wymaga bezpiecznych połączeń, biblioteki OpenSSL muszą zostać wdrożone (z wyjątkiem sytuacji, gdy na komponentach klienta jest używany [SChannel dla systemu Windows](#)).

W przypadku **aplikacji Windows** wystarczy wdrożyć biblioteki OpenSSL w tym samym folderze, w którym znajduje się aplikacja.

W przypadku innych osobowości należy zapoznać się z następującymi artykułami:

- [Kompilacja aplikacji OSX](#)
- [Budowanie aplikacji Android](#)
- [Tworzenie aplikacji iOS](#)

## CBuilder DEBUG

Jeśli używasz C++Builder i chcesz debugować bibliotekę sgcWebSockets, wykonaj następujące kroki:

- Przejdź do opcji projektu.
- Włącz opcję Używaj debugowych plików .dcu w sekcji Kompilator Delphi > Kompilowanie.
- Wyłącz łączenie z dynamiczną biblioteką RTL w opcjach linkera C++.
- Wyłącz opcję Link with Runtime Packages w sekcji Packages > Runtime Packages.

Następnie będzie możliwe debugowanie krok po kroku kodu źródłowego VCL/RTL.

# Kompilacja | Aplikacja OSX

Aby zbudować aplikację OSX z biblioteką sgcWebSockets, należy postępować zgodnie z instrukcjami ze strony Embarcadero dotyczącymi tworzenia aplikacji OSX.

## Instalacja PASServer w systemie MacOS

[http://docwiki.embarcadero.com/RADStudio/Sydney/en/PAServer,\\_the\\_Platform\\_Assistant\\_Server\\_Application](http://docwiki.embarcadero.com/RADStudio/Sydney/en/PAServer,_the_Platform_Assistant_Server_Application)

## Uzyskaj certyfikat Developer ID

Zaloguj się przy użyciu ważnego konta Apple Developer na stronie <https://developer.apple.com> i utwórz nowy certyfikat „Developer ID Application” z menu Certificates.

[http://docwiki.embarcadero.com/RADStudio/Rio/en/MacOS\\_Notarization](http://docwiki.embarcadero.com/RADStudio/Rio/en/MacOS_Notarization)

## Utwórz nowy Apple ID

Następnie przejdź do <https://appleid.apple.com/account>, aby utworzyć nowe Apple ID.

## Konfiguracja aprowizacji

Na koniec otwórz menu **Project / Options / Provisioning** i wypełnij wymagane dane, aby notaryzować aplikację OSX.

☐

Jeśli projekt wymaga określonych bibliotek, nie zapomnij uwzględnić ich w menu **Project / Deployment**.  
Ustaw Remote Path na „Contents\MacOS”

☐

Biblioteki te zostaną automatycznie podpisane w momencie notaryzacji aplikacji; można sprawdzić, czy biblioteka została podpisana, używając następującego polecenia:

```
codesign -dv --verbose=4 libcrypto.1.1.dylib
```

Przeczytaj więcej o tym, jak [skonfigurować OpenSSL w systemie OSX](#).

# Buduj | Aplikacja Android

Aby zbudować aplikację dla systemu Android z użyciem biblioteki sgcWebSockets, należy postępować zgodnie z krokami opisanymi na stronie Embarcadero dotyczącej tworzenia aplikacji Android.

## Tworzenie aplikacji Android

[http://docwiki.embarcadero.com/RADStudio/Sydney/en/Creating\\_an\\_Android\\_App](http://docwiki.embarcadero.com/RADStudio/Sydney/en/Creating_an_Android_App)

## Wdrożenie projektu

Jeśli projekt wymaga pewnych bibliotek, nie należy zapominać o ich dołączeniu w menu **Project / Deployment**. Ustaw Remote Path na ".\assets\internal"



Więcej informacji można znaleźć w sekcji [Konfiguracja OpenSSL na platformie Android](#).

# Budowanie | Aplikacja iOS

Aby zbudować aplikację iOS z biblioteką sgcWebSockets, należy wykonać kroki opisane na stronie Embarcadero dotyczącej budowania aplikacji iOS.

## Instalacja PASServer w systemie MacOS

[http://docwiki.embarcadero.com/RADStudio/Sydney/en/PAServer,\\_the\\_Platform\\_Assistant\\_Server\\_Application](http://docwiki.embarcadero.com/RADStudio/Sydney/en/PAServer,_the_Platform_Assistant_Server_Application)

## Uzyskaj certyfikat deweloperski iOS

Zaloguj się prawidłowym kontem Apple Developer na stronie <https://developer.apple.com> i utwórz nowy certyfikat „iOS Development Certificate” z menu Certificates.

Utwórz nowy identyfikator dla aplikacji iOS i nowy profil inicjowania obsługi.

[http://docwiki.embarcadero.com/RADStudio/Sydney/en/IOS\\_Mobile\\_Application\\_Development](http://docwiki.embarcadero.com/RADStudio/Sydney/en/IOS_Mobile_Application_Development)

## Konfiguruj identyfikator pakietu

Otwórz menu **Project / Options / Application / Version Info** i ustaw identyfikator Bundle Identifier



## Wdrożenie

Jeśli projekt wymaga bibliotek statycznych, należy skopiować te biblioteki do folderu Embarcadero lib/iosDevice64:

- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\debug
- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\release



Przeczytaj więcej o tym, jak [skonfigurować OpenSSL na iOS](#)

## Apro wizacja

Na koniec sprawdź w menu **Project / Options / Deployment**, czy certyfikat został pomyślnie załadowany.



# Serwery o wysokiej wydajności

## Serwery oparte na bibliotece Indy

[TsgcWebsocketServer](#) i [TsgcWebsocketHTTPServer](#) bazują na bibliotece Indy, dlatego każde połączenie jest obsługiwane przez oddzielny wątek — przy 1000 równoległych połączeń istnieje co najmniej 1000 wątków. Gdy wydajność jest priorytetem, należy zastosować odpowiednie optymalizacje. **Od wersji sgcWebSockets 4.3.3 serwery Indy obsługują również IOCP**; więcej informacji można znaleźć [tutaj](#).

Skorzystaj z poniższych wskazówek, aby zwiększyć wydajność serwera.

1. Ustaw we właściwości komponentu serwera **NotifyEvents := neNoSync**. Oznacza to, że zdarzenia są wywoływane w kontekście wątku połączenia, bez mechanizmu synchronizacji. Jeśli konieczny jest dostęp do kontrolek VCL lub obiektów współdzielonych, należy stosować własne mechanizmy synchronizacji.

2. W komponencie serwera ustaw właściwość **Optimizations.Connections.Enabled := True**. Jeśli planowane jest utrzymywanie ponad 1000 jednoczesnych połączeń na serwerze i metoda `Server.WriteData` jest wywoływana często, należy włączyć tę właściwość. Zasadniczo zapisuje ona połączenia na liście pamięci podręcznej, w której wyszukiwanie jest szybsze niż dostęp do listy połączeń Indy.

2.1 `CacheSize`: liczba połączeń przechowywanych w szybkiej pamięci podręcznej. Wartość domyślna = 100.

2.2 `GroupLevel`: wewnętrznie tworzy kilka list podzielonych według pierwszego znaku, więc przy dużej liczbie połączeń wyszukiwania są szybsze. Wartość domyślna = 1.

3. Ustaw właściwość składnika serwera **Optimizations.Channels.Enabled := True**. Po włączeniu tej właściwości kanały są zapisywane na liście, co umożliwia szybsze przeszukiwanie niż poprzednia metoda.

4. Ustaw we właściwości komponentu serwera **Optimizations.ConnectionsFree.Enabled := True**. Jeśli ta właściwość jest włączona, przy każdym rozłączeniu zamiast usuwać obiekt `TsgcWSConnection`, jest on przechowywany na liście i co X sekund wszystkie obiekty z tej listy są niszczone. Włączenie tej właściwości zwiększa zużycie pamięci, dlatego w przypadku dużej liczby rozłączeń w krótkim czasie należy ustawić tę właściwość na `false`.

4.1 `Interval`: liczba sekund, po której wszystkie rozłączone połączenia przechowywane na liście są niszczone. Domyślnie 60.

5. Domyślnie sgcWebSockets używa **sekcji krytycznych** do ochrony dostępu do współdzielonych obiektów. Zamiast sekcji krytycznych można jednak używać TMonitor lub SpinLocks. Wystarczy skompilować projekt z jedną z następujących dyrektyw kompilatora

3.1 {\$DEFINE SGC\_SPINLOCKS}

3.2 {\$DEFINE SGC\_TMONITOR}

6. Należy użyć najnowszej wersji **FastMM4**, dostępnej do pobrania pod adresem: <https://github.com/plieriche/FastMM4>

FastMM4 to bardzo dobry menedżer pamięci, jednak czasem nie skaluje się dobrze w aplikacjach wielowątkowych. W aplikacji należy użyć następującej dyrektywy kompilatora:

{\$DEFINE UseReleaseStack}

Następnie dodaj FastMM4 jako pierwszą jednostkę w sekcji uses projektu i skompiluj ponownie. W przypadku serwera o wysokiej współbieżności zauważysz wzrost wydajności.

Ta zmiana działa w następujący sposób: jeśli blok nie może zostać zwolniony natychmiast podczas wywołania FreeMem, zostanie dodany do listy bloków, które zostaną zwolnione później, w wątku czyszczenia działającym w tle lub podczas następnego wywołania FreeMem.

7. Zamiast FastMM4 lepiej jest użyć najnowszego **FastMM5**, który można pobrać ze strony: <https://github.com/plieriche/FastMM5>

Jest to nowa wersja autorstwa tego samego twórcy co FastMM4, obsługuje Delphi XE3 i nowsze oraz może być używana na platformach Windows 32-bit i Windows 64-bit.

FastMM5 jest objęty podwójną licencją, co oznacza dwie dostępne licencje: GPL i komercyjną. Jeśli użytkownik chce korzystać z FastMM5 w projektach komercyjnych, musi nabyć odpowiednią licencję.

Poniżej przedstawiono siatkę porównującą wydajność FastMM4 i FastMM5 przy 100 000 żądaniach i odpowiedziach WebSocket przy użyciu 1, 10, 100, 500 i 1000 równoczesnych klientów. Wydajność FastMM5 w aplikacjach wielowątkowych jest znacznie lepsza niż FastMM4.

| Klienci | Windows | FMM4 | FMM5 | Różnica |
|---------|---------|------|------|---------|
| 1       | Win32   | 4135 | 4214 | 1,91%   |
|         | Win64   | 4052 | 4520 | 11,55%  |
| 10      | Win32   | 4214 | 1729 | -58,97% |
|         | Win64   | 4104 | 1875 | -54,31% |
| 100     | Win32   | 3958 | 1604 | -59,47% |
|         | Win64   | 3958 | 1614 | -59,22% |

|            |      |      |         |
|------------|------|------|---------|
| 500 Win32  | 4098 | 1723 | -57,96% |
| Win64      | 5333 | 1791 | -66,42% |
| 1000 Win32 | 5927 | 2208 | -62,75% |
| Win64      | 8166 | 2229 | -72,70% |

## Indy Server Windows

sgcWebSockets Enterprise Edition obsługuje **IOCP** w systemie Windows, co oznacza, że zamiast tworzenia jednego wątku dla każdego połączenia, pula wątków obsługuje wszystkie połączenia. Aby włączyć IOCP, wystarczy ustawić IOHandler na IOCP.

```
IOHandlerOptions.IOHandlerType = iohIOCP
```

Właściwość IOHandlerOptions.IOCP umożliwia dostosowanie właściwości IOCP.

- **IOCPThreads:** są to wątki używane do obsługi połączeń; domyślnie wartość wynosi zero, co oznacza, że liczba wątków zostanie obliczona automatycznie na podstawie liczby procesorów (w przypadku Delphi 7 do Delphi 2007 wartość ta jest ustawiona na 32, ponieważ funkcja zliczania procesorów nie jest obsługiwana).
- **WorkOpThreads:** ustaw wartość większą od zera, jeśli żądania dla każdego połączenia mają być zawsze obsługiwane przez ten sam wątek. Domyślnie żądania IOCP są obsługiwane przez losowe wątki. Jeśli połączenia mają być zawsze obsługiwane przez ten sam wątek, należy ustawić wartość większą od zera. Przykład: ustawienie WorkOpThreads = 32 spowoduje utworzenie 32 wątków, a każde nowe żądanie dla wcześniej przetworzonego połączenia będzie kolejgowane w tym samym wątku.

IOCP jest zalecane, gdy wymagana jest obsługa tysięcy równoczesnych połączeń.

## Indy Server Linux

sgcWebSockets Enterprise Edition obsługuje **EPOLL** w systemie Linux, co oznacza, że zamiast tworzyć jeden wątek dla każdego połączenia, pula wątków obsługuje wszystkie połączenia. Aby włączyć EPOLL, należy ustawić IOHandler na EPOLL.

```
IOHandlerOptions.IOHandlerType = iohEPOLL
```

Właściwość `IOHandlerOptions.EPoll` umożliwia dostosowanie właściwości `EPOLL`.

- **EPOLLThreads:** wątki używane do obsługi połączeń; domyślna wartość wynosi zero, co oznacza, że liczba wątków będzie obliczana automatycznie na podstawie liczby procesorów.
- **WorkOpThreads:** należy ustawić wartość większą od zera, jeśli żądania dla każdego połączenia mają być obsługiwane zawsze przez ten sam wątek. Domyślnie żądania `EPOLL` są obsługiwane przez losowe wątki; jeśli połączenia mają być zawsze obsługiwane przez ten sam wątek, należy ustawić wartość większą od zera. Przykład: ustawienie `WorkOpThreads = 32` spowoduje utworzenie 32 wątków — każde nowe żądanie dla połączenia przetworzonego wcześniej zostanie kolejkiwane w tym samym wątku.

`EPOLL` jest zalecane, gdy zachodzi potrzeba obsługi tysięcy jednoczesnych połączeń.

## Serwer oparty na HTTP.SYS

Komponent `TsgcWebSocketServer_HTTPAPI` oparty jest na Microsoft HTTP API i jest zaprojektowany do pracy z IOCP, dlatego jest zalecany, gdy serwer musi obsługiwać tysiące połączeń. Ma jednak ograniczenie polegające na tym, że może działać wyłącznie w systemie Windows.

Serwer może obsługiwać protokoły **WebSocket** i **HTTP/2** na tym samym porcie i może współpracować z innymi implementacjami, ponieważ można go skonfigurować tak, aby obsługiwał tylko wybrane punkty końcowe.

**Przykład:** można skonfigurować ten serwer do obsługi połączeń `WebSocket` za pomocą biblioteki `sgcWebSockets` i pozwolić innym implementacjom, stronom trzecim lub dowolnym innym rozwiązaniom korzystać z innych punktów końcowych.

- Punkt końcowy: `https://server/ws` będzie obsługiwać połączenia korzystające z protokołu `WebSocket` przy użyciu `sgcWebSockets`
- Punkt końcowy: `https://server/other` będzie obsługiwać połączenia przy użyciu innej biblioteki.

Użyj najnowszego **FastMM5**; możesz pobrać go ze strony: <https://github.com/pleriche/FastMM5>

Jest to nowa wersja autorstwa tego samego twórcy co `FastMM4`, obsługuje Delphi XE3 i nowsze oraz może być używana na platformach Windows 32-bit i Windows 64-bit.

`FastMM5` jest objęty podwójną licencją, co oznacza dwie dostępne licencje: GPL i komercyjną. Jeśli użytkownik chce korzystać z `FastMM5` w projektach komercyjnych, musi nabyć odpowiednią licencję.

Poniżej przedstawiono siatkę porównującą wydajność `FastMM4` i `FastMM5` przy 100 000 żądaniach i odpowiedziach `WebSocket` przy użyciu 1, 10, 100, 500 i 1000 równoczesnych klientów. Wydajność `FastMM5` w aplikacjach wielowątkowych jest znacznie lepsza niż `FastMM4`.

| Klienci | Windows | FMM4 | FMM5 | Różnica |
|---------|---------|------|------|---------|
| 1       | Win32   | 5364 | 5182 | -3,39%  |
|         | Win64   | 5057 | 5026 | -0,61%  |
| 10      | Win32   | 4922 | 1744 | -64,57% |
|         | Win64   | 4958 | 1770 | -64,30% |
| 100     | Win32   | 3359 | 1682 | -49,93% |
|         | Win64   | 3979 | 1536 | -61,40% |
| 500     | Win32   | 2364 | 1890 | -20,05% |
|         | Win64   | 2901 | 1666 | -42,57% |
| 1000    | Win32   | 3296 | 1968 | -40,29% |
|         | Win64   | 4469 | 1989 | -55,49% |

# Menedżer pamięci

Niedawno ukazała się nowa wersja FastMM, opracowana przez Pierre le Riche — nowa wersja nosi nazwę **FastMM5** i została przepisana w celu poprawy wydajności w aplikacjach wielowątkowych; można ją skonfigurować pod kątem większej szybkości lub mniejszego zużycia pamięci i wiele więcej.

Obsługa od kompilatora Delphi **XE3** i możliwość użycia na platformach **Windows32** oraz **Windows64**.

FastMM5 jest **dostępny na podwójnej licencji**, czyli w dwóch wariantach: **GPL** i **Komercyjny**. Aby używać w projektach komercyjnych, należy zakupić licencję. Więcej szczegółów tutaj

<https://github.com/pleriche/FastMM5>

FastMM4 ma nową gałąź o nazwie **FastMM4-AVX**, opracowaną przez Maxima Masiutina, która dodaje bardzo interesujące funkcje, takie jak: bardziej wydajna synchronizacja, instrukcje AVX dla szybszego kopiowania pamięci, poprawa szybkości i inne. FastMM4-AVX jest objęty podwójną licencją: MPL i GPL. Więcej szczegółów tutaj:

<https://github.com/maximmasiutin/FastMM4-AVX>

## Konfiguracja

W celu przetestowania wydajności naszych komponentów utworzono nową konsolową aplikację Windows, **sgcBenchmark.exe**, służącą do pomiaru wydajności każdego menedżera pamięci przy użyciu komponentów sgcWebSockets.

Test jest bardzo prosty: klient (lub więcej niż jeden klient) łączy się z serwerem, wysyła wiadomość do serwera, a serwer odpowiada tym samym komunikatem do klienta. Operacja ta jest powtarzana 100 000 razy. Testy są powtarzane przy zmieniającej się liczbie równoczesnych klientów: najpierw 1, następnie 10, 100... Mierzony czas to czas upływający między pierwszą wiadomością wysłaną przez klienta a ostatnią wiadomością odebraną z serwera (czas potrzebny na połączenie z serwerem nie jest mierzony).

Benchmark porówna wydajność z użyciem domyślnego menedżera pamięci dostarczanego z Delphi 10.4.1, FastMM5 i FastMM4-AVX

## Benchmark serwera WebSocket Indy

W pierwszym teście porównawczym używany serwer to [serwer WebSocket Indy](#), oparty na serwerze TCP Indy, co oznacza, że każde połączenie tworzy 1 wątek.

Wartości są mierzone w milisekundach, więc na przykład pierwszy test przeprowadzony z 1 klientem na platformach Windows32, przy użyciu domyślnego menedżera pamięci, zajmuje 4135 milisekund, przy użyciu FastMM5 zajmuje 4214 milisekund, a przy użyciu FastMM4-AVX zajmuje 4823 milisekund. Obliczany procent odnosi się do wartości referencyjnej, którą jest w tym przypadku domyślny menedżer pamięci dostarczany z Delphi; im niższy procent, tym lepsza wydajność.

Test porównawczy został przeprowadzony 3 razy, a podane wartości są sumą wyników podzieloną przez 3.

Na potrzeby testu porównawczego użyto następującego serwera:

- Windows 2016 Server Datacenter
- 16 procesorów wirtualnych
- 32 GB RAM
- 2,2 GHz

Użyta wersja Delphi to Delphi 10.4.1 oraz najnowsze wersje FastMM5 i FastMM4-AVX z serwerów github.

Poniżej znajduje się wynik testu wydajności.

| Klien-<br>ci | Platfor-<br>ma | Domyślnie<br>(ms) | FMM5<br>(ms) | FMM5<br>(%) | FMM4-AVX<br>(ms) | FMM4-AVX<br>(%) |
|--------------|----------------|-------------------|--------------|-------------|------------------|-----------------|
| 1            | Win32          | 4135              | 4214         | 1,91%       | 4823             | 16,64%          |
| 1            | Win64          | 4052              | 4520         | 11.55%      | 4328             | 6,81%           |
| 10           | Win32          | 4214              | 1729         | -58.97%     | 1828             | -56,62%         |
| 10           | Win64          | 4104              | 1875         | -54,31%     | 1651             | -59.77%         |
| 100          | Win32          | 3958              | 1604         | -59,47%     | 1583             | -60.01%         |
| 100          | Win64          | 3958              | 1614         | -59.22%     | 1635             | -58,69%         |
| 500          | Win32          | 4098              | 1723         | -57.96%     | 1854             | -54,76%         |
| 500          | Win64          | 5333              | 1791         | -66,42%     | 1833             | -65,63%         |
| 1000         | Win32          | 5927              | 2208         | -62.75%     | 2328             | -60,72%         |
| 1000         | Win64          | 8166              | 2229         | -72.70%     | 2234             | -72,64%         |

## Benchmark HTTP.SYS Server

W drugim benchmarku użyty serwer to [serwer WebSocket HTTP.SYS](#), oparty na platformie HTTP API Microsoft i obsługujący połączenia w puli wątków.

Wartości są mierzone w milisekundach, więc na przykład pierwszy test przeprowadzony z 1 klientem na platformach Windows32 przy użyciu domyślnego menedżera pamięci zajmuje 5364 milisekundy, przy użyciu FastMM5 zajmuje 5182 milisekundy, a przy użyciu FastMM4-AVX zajmuje 5838 milisekund. Obliczany procent odnosi się do wartości referencyjnej, w tym przypadku do domyślnego menedżera pamięci dostarczanego z Delphi — im niższy procent, tym lepsza wydajność.

Test porównawczy został przeprowadzony 3 razy, a podane wartości są sumą wyników podzieloną przez 3.

Na potrzeby testu porównawczego użyto następującego serwera:

- Windows 2016 Server Datacenter
- 16 procesorów wirtualnych
- 32 GB RAM
- 2,2 GHz

Użyta wersja Delphi to Delphi 10.4.1 oraz najnowsze wersje FastMM5 i FastMM4-AVX z serwerów github.

Poniżej znajduje się wynik testu wydajności.



| Klien-<br>ci | Platfor-<br>ma | Domyślnie<br>(ms) | FMM5<br>(ms) | FMM5<br>(%) | FMM4-AVX<br>(ms) | FMM4-AVX<br>(%) |
|--------------|----------------|-------------------|--------------|-------------|------------------|-----------------|
| 1            | Win32          | 5364              | 5182         | -3,39%      | 5838             | 8,84%           |
| 1            | Win64          | 5507              | 5206         | -0,61%      | 5135             | 1.54%           |
| 10           | Win32          | 4922              | 1744         | -64,57%     | 2088             | -57.58%         |
| 10           | Win64          | 4958              | 1770         | -64.30%     | 1953             | -60.61%         |
| 100          | Win32          | 3359              | 1682         | -49,93%     | 2244             | -33.19%         |
| 100          | Win64          | 3979              | 1536         | -61,40%     | 1859             | -53.28%         |
| 500          | Win32          | 2364              | 1890         | -20,05%     | 2344             | -0.85%          |
| 500          | Win64          | 2901              | 1666         | -42.57%     | 1859             | -35.92%         |
| 1000         | Win32          | 3296              | 1968         | -40.29%     | 2531             | -23.21%         |
| 1000         | Win64          | 4469              | 1989         | -55,49%     | 2047             | -54,20%         |

## Uwagi dotyczące testów wydajności

Poniżej przedstawiono komentarze dotyczące wyników uzyskanych po testach porównawczych 3 różnych menedżerów pamięci:

- W aplikacjach jednowątkowych nie ma istotnych różnic w wydajności między FastMM4, FastMM5 a FastMM4-AVX.
- **FastMM5 i FastMM4-AVX** działają znacznie **lepiej** w aplikacjach **wielowątkowych**.
- **Różnice** między FastMM5 a FastMM4-AVX są **niewielkie**, przynajmniej w tych testach porównawczych.
- **Windows 32** uzyskuje **lepsze** wyniki w testach niż **Windows 64**. Użycie FastMM5 lub FastMM4-AVX w aplikacjach Windows 64 poprawia wydajność w większym stopniu niż w Windows 32.

Ostateczny wybór menedżera pamięci zależy od projektu. Uważa się, że nie istnieje jeden menedżer pamięci, który sprawdza się najlepiej we wszystkich warunkach, dlatego przed dokonaniem wyboru należy przeprowadzić dokładne testy, aby sprawdzić, który z nich zapewnia najlepszą wydajność dla danych potrzeb.

# OpenSSL

OpenSSL to biblioteka oprogramowania dla aplikacji zabezpieczających komunikację w sieciach komputerowych przed podsłuchem lub wymagających identyfikacji strony zdalnej. Jest szeroko stosowana przez serwery internetowe, w tym przez zdecydowaną większość witryn HTTPS.

Ta biblioteka jest wymagana przez komponenty oparte na bibliotece Indy, gdy wymagane jest bezpieczne połączenie. Jeśli aplikacja wymaga OpenSSL, przed wdrożeniem aplikacji należy upewnić się, że niezbędne pliki są obecne w systemie plików:

Aktualnie sgcWebSockets obsługuje wersje OpenSSL: **1.0.2**, **1.1** oraz **3.0 do 3.3**.

| Platforma                              | API 1.0                        | API 1.1                               | API 3.*                           | Linkowanie statyczne/dynamiczne |
|----------------------------------------|--------------------------------|---------------------------------------|-----------------------------------|---------------------------------|
| Windows (32-bit i 64-bit)              | libeay32.dll and sslseay32.dll | libcrypto-1_1.dll and libssl-1_1.dll  | libcrypto-3.dll i libssl-3.dll    | Dynamiczny                      |
| OSX                                    | libcrypto.dylib, libssl.dylib  | libcrypto.1.1.dylib, libssl.1.1.dylib | libcrypto.3.dylib, libssl.3.dylib | Dynamiczny                      |
| Urządzenie iOS (32-bitowe i 64-bitowe) | libcrypto.a i libssl.a         | libcrypto.a i libssl.a                | libcrypto.a i libssl.a            | Statyczna                       |
| Symulator iOS                          | libcrypto.dylib, libssl.dylib  | libcrypto.1.1.dylib, libssl.1.1.dylib | libcrypto.3.dylib, libssl.3.dylib | Dynamiczny                      |
| Urządzenie Android                     | libcrypto.so, libssl.so        | libcrypto.so, libssl.so               | libcrypto.so, libssl.so           | Dynamiczny                      |

Poniżej opisano sposób **konfiguracji bibliotek OpenSSL** dla każdej platformy:

- [Windows](#)
- [OSX](#)
- [Android](#)
- [iOS](#)

## Konfiguracje openssl

Komponenty sgcWebSockets oparte na Indy umożliwiają konfigurację wybranych właściwości OpenSSL. Dostępne są następujące właściwości:

- **Komponenty serwera:** `SSLOptions.OpenSSL_Options`.
- **Komponenty klienta:** `TLSOptions.OpenSSL_Options`.

## Wersja API

**Standardowa biblioteka Indy** umożliwia ładowanie wyłącznie bibliotek **OpenSSL 1.0.2**, które zostały wycofane, a najnowsze wersje OpenSSL korzystają z API 1.1.1.

**sgcWebSockets Enterprise** umożliwia ładowanie bibliotek **openssl 1.1.1**. W tej właściwości można skonfigurować, która wersja API openssl zostanie załadowana. Na jeden proces może być załadowana tylko jedna wersja API (nie można mieszać bibliotek openssl 1.0.2 i 1.1.1 w tej samej aplikacji).

## LibPath

Ta właściwość umożliwia ustawienie lokalizacji bibliotek openssl. Jest to przydatne w projektach dla systemu Android lub OSX, gdzie lokalizacja bibliotek openssl musi być określona.

Akceptuje następujące wartości:

- **oslpNone:** ta wartość nie ustawia żadnej ścieżki biblioteki (jest wartością domyślną).
- **oslpDefaultFolder:** wartość ta określa domyślny folder bibliotek OpenSSL. Ścieżka ta jest różna dla każdej platformy (Windows, macOS itd.).

## Ładowanie dodatkowych funkcji OpenSSL

Użyj wywołania zwrotnego, aby załadować dodatkowe funkcje OpenSSL niezdefiniowane domyślnie. Więcej informacji można znaleźć pod adresem [OpenSSL Load Additional Functions](#).

## Szyfry

Aby zapewnić obsługę TLS 1.2 i 1.3 na serwerze przy użyciu najwyższego poziomu bezpieczeństwa i wydajności, należy zastosować następującą konfigurację:

```
SSLOptions.Version := tls1_3;
```

```
SSLOptions.OpenSSL_Options.VersionMin := tls1_2;
```

```
SSLOptions.OpenSSL_Options.APIVersion := oslAPI_3_0;
```

Następnie ustaw poniższą listę szyfrów.

```
AEAD-AES128-GCM-SHA256:AEAD-AES256-GCM-SHA384:AEAD-CHACHA20-POLY1305-  
SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-  
RSA-AES128-GCM-SHA256:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-GCM-  
SHA384:ECDHE-RSA-AES256-GCM-SHA384
```

## Certyfikaty z podpisem własnym

Do celów testowych można używać certyfikatów z podpisem własnym. Wystarczy wykonać następujące polecenie, aby utworzyć certyfikat z podpisem własnym:

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
```

Zostaną utworzone 2 pliki: cert.pem (certyfikat) i key.pem (klucz prywatny). Oba pliki można połączyć w jeden. Należy utworzyć nowy plik i skopiować do niego zawartość obu plików. Uzyskana struktura będzie wyglądać następująco:

```
-----BEGIN PRIVATE KEY-----
```

```
. . . .
```

```
-----END PRIVATE KEY-----
```

```
-----BEGIN CERTIFICATE-----
```

```
. . . .
```

```
-----END CERTIFICATE-----
```

## Typowe błędy

**SSL\_GET\_RECORD: wrong version number**

Błąd ten oznacza, że serwer i klient używają różnych wersji protokołu SSL/TLS. Aby naprawić ten problem, należy ustawić właściwą wersję w komponencie serwera lub klienta.

```
Server.SSLOptions.Version
```

```
Client.TLSOptions.Version
```

### **SSL3\_GET\_RECORD: decryption failed or bad record mac**

Zwykle ten błąd pojawia się, gdy:

1. Należy sprawdzić, czy jest używana najnowsza wersja OpenSSL. Jeśli jest zbyt stara, należy zaktualizować ją do najnowszej obsługiwanej wersji.
2. Jeśli ten błąd pojawia się losowo, zazwyczaj oznacza to, że więcej niż jeden wątek uzyskuje dostęp do połączenia OpenSSL. Można spróbować ustawić `NotifyEvents = neNoSync`, co oznacza, że zdarzenia: `OnConnect`, `OnDisconnect`, `OnMessage...` będą wywoływane w kontekście wątku połączenia — unika to pewnych problemów z synchronizacją i zapewnia lepszą wydajność. Wadą jest to, że na przykład podczas aktualizowania kontrolki wizualnej w formularzu po odebraniu wiadomości, należy samodzielnie implementować metody synchronizacji, ponieważ kontrolki wizualne nie są bezpieczne wątkowo.

# OpenSSL | Windows

Dostępna jest jedna wersja 32-bitowa i jedna 64-bitowa. Biblioteki te należy skopiować do tego samego folderu, w którym znajduje się aplikacja, lub do ścieżki systemowej.

Jeśli system operacyjny jest 32-bitową wersją systemu Windows, należy skopiować pliki do folderu System32.

Jeśli system operacyjny to 64-bitowy Windows, należy skopiować wersję 64-bitową do folderu System32, a wersję 32-bitową do folderu SysWow64.

## API 1.0

Wymaga następujących bibliotek:

- libeay32.dll
- ssleay32.dll

Jeśli system operacyjny jest 32-bitową wersją systemu Windows, należy skopiować pliki do folderu System32.

Jeśli system operacyjny to 64-bitowy Windows, należy skopiować wersję 64-bitową do folderu System32, a wersję 32-bitową do folderu SysWow64.

Najnowsze biblioteki można pobrać ze swojego konta (biblioteki nie mają zewnętrznych zależności i są podpisane cyfrowo).

## API 1.1

Wymaga następujących bibliotek:

## Windows 32

- libcrypto-1\_1.dll
- libssl-1\_1.dll

## Windows 64

- libcrypto-1\_1-x64.dll
- libssl-1\_1-x64.dll

Jeśli system operacyjny jest 32-bitową wersją systemu Windows, należy skopiować pliki do folderu System32.

Jeśli system operacyjny to 64-bitowy Windows, należy skopiować wersję 64-bitową do folderu System32, a wersję 32-bitową do folderu SysWow64.

Najnowsze biblioteki można pobrać ze swojego konta (biblioteki nie mają zewnętrznych zależności i są podpisane cyfrowo).

### **API 3.\***

Wymaga następujących bibliotek:

#### **Windows 32**

- libcrypto-3.dll
- libssl-3.dll

#### **Windows 64**

- libcrypto-3-x64.dll
- libssl-3-x64.dll

Jeśli system operacyjny jest 32-bitową wersją systemu Windows, należy skopiować pliki do folderu System32.

Jeśli system operacyjny to 64-bitowy Windows, należy skopiować wersję 64-bitową do folderu System32, a wersję 32-bitową do folderu SysWow64.

Najnowsze biblioteki można pobrać ze swojego konta (biblioteki nie mają zewnętrznych zależności i są podpisane cyfrowo).

W przypadku użycia certyfikatu p12 wymagane jest wdrożenie biblioteki legacy.dll. Więcej informacji na stronie [Certyfikaty OpenSSL p12](#).





# OpenSSL | OSX

Nowsze wersje OSX nie zawierają bibliotek OpenSSL lub są zbyt przestarzałe, dlatego należy je wdrożyć wraz z aplikacją. Wdrożenia tych bibliotek należy dokonać zgodnie z następującymi krokami:

- Otwórz projekt/wdrożenie w swoim projekcie.
- Dodaj wymagane biblioteki.
- Ustaw RemotePath = 'Contents\Macos\'.
  - Skonfiguruj ścieżkę LibPath OpenSSL na folder domyślny:
    - Client.TLSOptions.OpenSSL\_Options.LibPath = oslpDefaultFolder.
    - Server.SSLOptions.OpenSSL\_Options.LibPath = oslpDefaultFolder.

## API 1.0

Wymaga następujących bibliotek:

- libcrypto.dylib
- libssl.dylib

Najnowsze biblioteki można pobrać ze swojego konta.

## API 1.1

Wymaga następujących bibliotek:

- libcrypto.1.1.dylib
- libssl.1.1.dylib

Dostępna jest jedna wersja dla 32 bitów i jedna dla 64 bitów. Biblioteki te należy skopiować do tego samego folderu, w którym znajduje się aplikacja.

Najnowsze biblioteki można pobrać ze swojego konta.

## API 3.0

Wymaga następujących bibliotek:

- libcrypto.3.dylib
- libssl.3.dylib

Dostępna jest wyłącznie wersja 64-bitowa. Biblioteki te należy skopiować do tego samego folderu, w którym znajduje się aplikacja.

Najnowsze biblioteki można pobrać ze swojego konta.

Jeśli biblioteki OpenSSL zostaną dołączone do aplikacji OSX, po notaryzacji aplikacji biblioteki zostaną podpisane; można to sprawdzić za pomocą następującego polecenia:

```
codesign -dv --verbose=4 libcrypto.1.1.dylib
```

Poniższy film pokazuje, jak zbudować aplikację MacOSX64 z bibliotekami OpenSSL.

<https://www.esegece.com/websockets/videos/delphi/quickstart/275-build-macosx64-application/file>

## Błędy

**Klienci nie powinni ładować wersjonowanego dylib libcrypto, ponieważ nie ma stabilnego ABI.**

W systemie MacOS Monterey i nowszych może wystąpić ten błąd podczas próby załadowania bibliotek OpenSSL — błąd pojawia się przy pierwszej próbie załadowania bibliotek OpenSSL bez numeru wersji (np. libcrypto.dylib).

Aby naprawić ten błąd, należy ustawić we właściwości **OpenSSL\_Options.UnixSymLinks** wartość **osls-SymLinksDontLoad**. Zapobiega to ładowaniu bibliotek openssl bez wersji.

# OpenSSL | Android

Nowsze wersje systemu Android nie zawierają bibliotek OpenSSL lub zawierają zbyt stare wersje, dlatego należy je dołączyć do aplikacji. Wdróż te biblioteki, wykonując następujące kroki:

- Otwórz projekt/wdrożenie w swoim projekcie.
- Dodaj wymagane biblioteki.
- Ustaw RemotePath = '.\assets\internal'.
- Skonfiguruj ścieżkę LibPath OpenSSL na folder domyślny:
  - Client.TLSOptions.OpenSSL\_Options.LibPath = oslpDefaultFolder.
  - Server.SSLOptions.OpenSSL\_Options.LibPath = oslpDefaultFolder.

## API 1.0

Wymaga następujących bibliotek:

- libcrypto.so
- libssl.so

Najnowsze biblioteki można pobrać ze swojego konta.

Na **Android 64-bitowy**, użycie TLS 1.2 może spowodować następujący błąd:

*INT\_RSA\_VERIFY:bad signature*

Jest to błąd OpenSSL naprawiony w API 1.1.

Można spróbować użyć TLS 1.0 lub TLS 1.1 (jeśli serwer nadal obsługuje te metody szyfrowania, aby uniknąć tego błędu).

## API 1.1

Wymaga następujących bibliotek:

- [libcrypto.so](#)
- [libssl.so](#)

Najnowsze biblioteki można pobrać ze swojego konta.

## **API 3.0**

Wymaga następujących bibliotek:

- [libcrypto.so](#)
- [libssl.so](#)

Najnowsze biblioteki można pobrać ze swojego konta.

# OpenSSL | iOS

Aby zainstalować OpenSSL na 64-bitowym urządzeniu iOS, należy skopiować pliki biblioteki SSL libcrypto.a i libssl.a do systemu. Należy pobrać plik .zip iOS OpenSSL, wypakować go i znaleźć pliki .a w katalogu \lib. Pliki biblioteki SSL libcrypto.a i libssl.a należy skopiować do następujących katalogów:

- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\debug
- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\release

Dodaj jednostkę **sgcIdSSLOpenSSLHeaders\_static** (lub IdSSLOpenSSLHeaders\_static, jeśli Twoja edycja sgcWebSockets nie jest Enterprise) do klauzuli uses.

Jeśli zachodzi potrzeba **wdrożenia** dowolnego pliku, można ustawić RemotePath = StartUp\Documents, a do załadowania pliku należy użyć poniższego kodu (wymaga dodania System.IOUtils do klauzuli uses):

```
TPath.GetDocumentsPath + PathDelim + <nazwa pliku>
```

**Biblioteki openssl nie mogą być wdrażane za pomocą menu Project/Deployment w systemie iOS.**

## API 1.1

Zmodyfikuj **IdCompilerDefines.inc** i włącz SGC\_OPENSSL\_API\_1\_1 w sekcji IOS:

```
{ $IFDEF IOS }

{ $DEFINE HAS_getifaddrs }

{ $DEFINE USE_OPENSSL }

{ $IFDEF CPUARM }

// RLebeau: For iOS devices, OpenSSL cannot be used as an external
library,

// musi być statycznie zlinkowany z aplikacją. W przypadku symula-
tora iOS,

// nie jest prawdą. Użytkownicy chcący korzystać z OpenSSL w apli-
kacjach na urządzenia iOS będą potrzebować
```

```
// aby dodać statyczną bibliotekę OpenSSL do projektu i następnie
dołączyć

// IdSSLOpenSSLHeaders_static unit in their uses clause. It hooks
up the

// statycznie linkowane funkcje dla jednostki IdSSLOpenSSLHe-
aders...

{$DEFINE STATICLOAD_OPENSSL}

// sgc--> enable for openssl API 1.1

{$DEFINE SGC_OPENSSL_API_1_1}

{$ENDIF}

{$ENDIF}
```

Biblioteki można pobrać ze swojego konta.

### API 3.0

Zmodyfikuj **IdCompilerDefines.inc** i włącz SGC\_OPENSSL\_API\_1\_1 oraz SGC\_OPENSSL\_API\_3\_0 w sekcji IOS:

```
{ $IFDEF IOS}

{$DEFINE HAS_getifaddrs}

{$DEFINE USE_OPENSSL}

{$IFDEF CPUARM}

// RLebeau: For iOS devices, OpenSSL cannot be used as an external
library,

// musi być statycznie zlinkowany z aplikacją. W przypadku symula-
tora iOS,

// nie jest prawdą. Użytkownicy chcący korzystać z OpenSSL w apli-
kacjach na urządzenia iOS będą potrzebować

// aby dodać statyczną bibliotekę OpenSSL do projektu i następnie
dołączyć

// IdSSLOpenSSLHeaders_static unit in their uses clause. It hooks
up the
```

```
// statycznie linkowane funkcje dla jednostki IdSSLOpenSSLHe-  
aders...  
  
{ $DEFINE STATICLOAD_OPENSSL}  
  
// sgc--> enable for openssl API 1.1  
  
{ $DEFINE SGC_OPENSSL_API_1_1}  
  
// sgc--> enable for openssl API 3.0  
  
{ $DEFINE SGC_OPENSSL_API_3_0}  
  
{ $ENDIF}  
  
{ $ENDIF}
```

Biblioteki można pobrać ze swojego konta.

# OpenSSL | Własne certyfikaty CA

[Wpis na Github](#)

Aby utworzyć certyfikat podpisany przez własne CA, który może być zaufany przez przeglądarki internetowe (jak Chrome) po dodaniu certyfikatu CA do lokalnego magazynu.

## 1. Przygotuj pliki konfiguracyjne do tworzenia certyfikatów bez monitów

### CA.cnf

```
[ req ]
prompt = no
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
C = US
ST = Localzone
L = localhost
O = Certificate Authority Local Center
OU = Develop
CN = develop.localhost.localdomain
emailAddress = root@localhost.localdomain
```

### localhost.cnf

```
[req]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
countryName = US
stateOrProvinceName = Localzone
localityName = Localhost
organizationName = Certificate signed by my CA
commonName = localhost.localdomain
[req_ext]
subjectAltName = @alt_names
[v3_req]
subjectAltName = @alt_names
[alt_names]
IP.1 = 127.0.0.1
IP.2 = 127.0.0.2
IP.3 = 127.0.0.3
IP.4 = 192.168.0.1
IP.5 = 192.168.0.2
IP.6 = 192.168.0.3
DNS.1 = localhost
```



```
DNS.2 = localhost.localdomain
DNS.3 = dev.local
```

## 2. Wygeneruj prywatny klucz CA i certyfikat (ważny przez 5 lat).

```
openssl req -nodes -new -x509 -keyout CA_key.pem -out CA_cert.pem -days 1825 -config CA.cnf
```

## 3. Generowanie klucza tajnego serwera sieciowego i CSR

```
openssl req -sha256 -nodes -newkey rsa:2048 -keyout localhost_key.pem -out localhost.csr -config localhost.cnf
```

## 4. Utwórz certyfikat i podpisz go własnym urzędem certyfikacji (ważny 1 rok)

```
openssl x509 -req -days 398 -in localhost.csr -CA CA_cert.pem -CAkey CA_key.pem -CAcreateserial -out localhost_cert.pem -ex
```

## 5. Pliki wyjściowe będą następujące:

- `CA.cnf` → Plik konfiguracyjny OpenSSL CA. Może zostać usunięty po zakończeniu procesu tworzenia certyfikatu.
- `CA_cert.pem` → certyfikat [Urzędu Certyfikacji]. Ten certyfikat należy dodać do lokalnego magazynu urzędów certyfikacji w przeglądarce, aby ufała wszystkim certyfikatом utworzonym przy użyciu tego CA.
- `CA_cert.srl` → Losowy numer seryjny. Można usunąć po zakończeniu procesu tworzenia certyfikatu.
- `CA_key.pem` → Musi być użyty przy tworzeniu nowego certyfikatu [localhost]. Może zostać usunięty po zakończeniu procesu tworzenia certyfikatu (jeśli nie planuje się jego ponownego użycia wraz z `CA_cert.pem`).
- `localhost.cnf` → Plik konfiguracyjny certyfikatu OpenSSL SSL. Może zostać usunięty po zakończeniu procesu tworzenia certyfikatu.
- `localhost.csr` → Żądanie podpisania certyfikatu. Może zostać usunięte po zakończeniu procesu tworzenia certyfikatu.
- `localhost_cert.pem` → certyfikat SSL. **Musi być skonfigurowany we właściwości `SSLOptions.CertFile` serwera.**
- `localhost_key.pem` → Klucz tajny. **Musi zostać zainstalowany we właściwości `SSLOptions.KeyFile` serwera.**

# OpenSSL | Certyfikaty P12

OpenSSL 3.0 przeniósł kilka przestarzałych lub niezabezpieczonych algorytmów do wewnętrznego modułu biblioteki zwanego dostawcą legacy. Nie jest on domyślnie ładowany, dlatego aplikacje (lub ich środowiska uruchomieniowe) korzystające z OpenSSL do operacji kryptograficznych nie mogą używać takich algorytmów podczas wczytywania certyfikatów, tworzenia skrótów wiadomości itd.

Algorytmy dostawcy legacy obejmują: MD2, MD4, MDC2, RMD160, CAST5, BF (Blowfish), IDEA, SEED, RC2, RC4, RC5 oraz DES (ale nie 3DES).

Ze względów bezpieczeństwa zdecydowanie zaleca się rezygnację z używania tych starszych algorytmów.

Jeśli aplikacja korzysta z certyfikatów klienta przechowywanych w pliku zaszyfrowanym starszym szyfrem, takim jak RC2-40-CBC, istnieje możliwość „uwspółcześnienia” pliku certyfikatu przez ponowne szyfrowanie przy użyciu programu openssl.

Na przykład, jeśli na komputerze lokalnym znajduje się plik certyfikatu client.p12 (lub client.pfx):

```
$ openssl pkcs12 -legacy -in client.p12 -nodes -out cert-decrypted.tmp  
(wprowadź hasła, jeśli zostanie wyświetlony monit)
```

```
$ openssl pkcs12 -in cert-decrypted.tmp -export -out client-new.p12  
(wprowadź hasła, jeżeli zostaniesz o to poproszony)
```

```
$ rm cert-decrypted.tmp
```

Wyeksportowany plik certyfikatu client-new.p12 zawiera teraz te same klucze, ale zaszyfrowane przy użyciu AES-256-CBC.

Sprawdź poniżej konfigurację pakietów sgcWebSockets i sgcIndy:

## sgcWebSockets

- Ustaw właściwość **OpenSSL\_Options.Legacy.Enabled** na True.
- Ustaw lokalizację biblioteki Legacy.

- **OpenSSL\_Options.Legacy.LibPath:** tutaj można skonfigurować, gdzie znajduje się biblioteka legacy
  - **oslpNone:** wartość domyślna; biblioteka legacy powinna znajdować się w tym samym folderze co plik binarny lub w znanych ścieżkach.
  - **oslpDefaultFolder:** automatycznie ustawia ścieżkę biblioteki legacy tam, gdzie powinny znajdować się biblioteki dla wszystkich typów środowisk IDE.
  - **oslpCustomFolder:** jeśli wybrano tę opcję, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.
- **OpenSSL\_Options.Legacy.LibPathCustom:** gdy LibPath = oslpCustomFolder, należy tutaj zdefiniować pełną ścieżkę do biblioteki legacy.

## sgcIndy

- Ustaw właściwość **SSLOptions.Legacy** na True.
- Przed uruchomieniem serwera lub klienta należy ustawić ścieżkę do biblioteki legacy.dll. Należy użyć funkcji **IdOpenSSLSetOSSLPPath** i przekazać ścieżkę jako argument.

# OpenSSL | Weryfikacja certyfikatu

Podczas korzystania z OpenSSL przy włączonej opcji Verify Certificate może pojawić się następujący błąd:

```
Error connecting with SSL.error:80000002:system library::No such file  
or directory.
```

Jeśli zdarzenie OnVerifyPeer jest obsługiwane, a parametr Error ma wartość 20, błąd oznacza:

```
X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY
```

Główną przyczyną tego błędu jest obecność jednego lub więcej certyfikatów prezentowanych przez zdalny serwer, które nie są zapisane w magazynie certyfikatów aplikacji. Aby rozwiązać ten problem, można użyć właściwości RootCertFile i ustawić ścieżkę do pliku CA. Jeśli plik CA nie jest dostępny, można go pobrać np. z serwisu Mozilla:

<https://curl.haxx.se/docs/caextract.html>

Po ustawieniu RootCertFile poprzedni błąd powinien zniknąć.

# OpenSSL | Ładowanie dodatkowych funkcji

Domyślnie Indy definiuje najczęstsze funkcje OpenSSL potrzebne do szyfrowania komunikacji, jednak czasami wymagane są dodatkowe funkcje do szyfrowania, podpisywania itp. Można użyć metody `IdOpenSSLSetLoadFuncsCallback`, aby przypisać callback do dynamicznego ładowania dodatkowych funkcji OpenSSL.

## IdOpenSSLSetLoadFuncsCallback

```
TIdLoadSSLFuncsCallback = procedure(TIdLibHandle hIdSSL, TIdLibHandle hIdCrypto, const TStringList* FailedLoadList);
```

Jest to typ procedury służący jako wywołanie zwrotne przyjmujące trzy parametry:

- **hIdSSL:** `TIdLibHandle` — uchwyt do załadowanej biblioteki SSL.
- **hIdCrypto:** `TIdLibHandle` - uchwyt do załadowanej biblioteki kryptograficznej.
- **FailedLoadList:** `TStringList` — lista funkcji, których ładowanie zakończyło się niepowodzeniem.

Celem tego wywołania zwrotnego jest umożliwienie użytkownikowi wykonywania niestandardowego przetwarzania podczas ładowania funkcji OpenSSL, takiego jak rejestrowanie nieudanych prób załadowania funkcji lub obsługa błędów.

## IdOpenSSLSetUnLoadFuncsCallback

```
TIdLoadSSLFuncsCallback = procedure(TIdLibHandle hIdSSL, TIdLibHandle hIdCrypto, const TStringList* FailedLoadList);
```

Służy jako wywołanie zwrotne do zwalniania funkcji SSL. Jest przydatny do wykonywania czynności porządkowych podczas zwalniania bibliotek OpenSSL.

## Jak załadować funkcję niestandardową

Poniżej znajduje się prosty przykład ładowania funkcji `EVP_PKEY_CTX_set_rsa_padding` przy użyciu wywołań zwrotnych.

```
typedef int (__cdecl *TEVP_PKEY_CTX_set_rsa_padding)(void* ctx, int pad);  
TEVP_PKEY_CTX_set_rsa_padding EVP_PKEY_CTX_set_rsa_padding = nullptr;
```

```

// Mock function for LoadLibFunction (you need to implement this according to your needs)
void* LoadLibFunction(TIdLibHandle hLib, const char* funcName) {
    return GetProcAddress((HMODULE)hLib, funcName);
}

void __fastcall DoOpenSSLLoadFuncsCallback(TIdLibHandle hIdSSL, TIdLibHandle hIdCrypto, const TStringList* FailedLoadList)
    EVP_PKEY_CTX_set_rsa_padding = (TEVP_PKEY_CTX_set_rsa_padding)LoadLibFunction(hIdCrypto, "EVP_PKEY_CTX_set_rsa_padding")
}
void __fastcall DoOpenSSLUnloadFuncsCallback() {
    EVP_PKEY_CTX_set_rsa_padding = nullptr;
}

// Mock function to simulate IdOpenSSLSetLoadFuncsCallback
void IdOpenSSLSetLoadFuncsCallback(void (__fastcall *callback)(TIdLibHandle, TIdLibHandle, const TStringList*)) {
    callback(nullptr, nullptr, nullptr); // Example call
}
// Mock function to simulate IdOpenSSLSetUnloadFuncsCallback
void IdOpenSSLSetUnloadFuncsCallback(void (__fastcall *callback)()) {
    callback(); // Example call
}

int main() {
    IdOpenSSLSetLoadFuncsCallback(DoOpenSSLLoadFuncsCallback);
    IdOpenSSLSetUnloadFuncsCallback(DoOpenSSLUnloadFuncsCallback);
    return 0;
}

```

# Indy

Indy to biblioteka komunikacyjna klient/serwer o otwartym kodzie źródłowym obsługująca gniazda TCP/UDP/RAW oraz ponad 100 protokołów wyższego poziomu, w tym SMTP, POP3, IMAP, NNTP, HTTP, FTP i wiele innych. Indy jest napisana w Delphi, ale jest również dostępna dla C++Builder i FreePascal. sgcWebSockets używa Indy jako podstawy dla niektórych komponentów, a różne edycje sgcWebSockets w różny sposób korzystają z biblioteki Indy.

sgcWebSockets obsługuje protokoły takie jak HTTP/2, które wymagają użycia ALPN, może korzystać z TLS 1.3 przy użyciu openssl 1.1.1 lub openssl 3.0.0 — wszystkie te funkcje nie są obsługiwane przez standardową bibliotekę Indy, dlatego edycja Enterprise sgcWebSockets zawiera niestandardową bibliotekę Indy obsługującą te funkcje. Aby uniknąć odinstalowywania standardowej biblioteki Indy z IDE, wymagane pliki Indy są przemianowywane przez dodanie prefiksu "sgc", na przykład jednostka "IdGlobal" otrzymuje nazwę "sgcIdGlobal". Dzięki temu obie wersje mogą współistnieć bez problemów.

Poniżej przedstawiono wersję Indy używaną przez poszczególne edycje sgcWebSockets:

|              |                |
|--------------|----------------|
| STANDARD     | Standard       |
| PROFESSIONAL | Standard       |
| ENTERPRISE   | Niestandardowy |

Klienci posiadający licencje „Registered” (stare licencje sprzed podziału pakietu sgcWebSockets) znajdują następujące wersje pakietu sgcWebSockets:

|                         |                |
|-------------------------|----------------|
| sgcWebSockets           | Standard       |
| sgcWebSockets min       | Standard       |
| sgcWebSockets min Indy* | Niestandardowy |

\*sgcWebSockets\_min\_indy jest tożsamy z edycją sgcWebSockets Enterprise.

Użycie niestandardowej wersji Indy jest określone w pliku "sgcVer.inc" znajdującym się w folderze źródłowym. Istnieje dyrektywa kompilatora o nazwie "SGC\_CUSTOM\_INDY", która włącza lub wyłącza korzystanie z tej wersji Indy. Jeśli posiadasz edycję Enterprise i chcesz wyłączyć korzystanie z niestandardowego Indy, usuń następującą dyrektywę kompilatora:

```
{ $DEFINE SGC_CUSTOM_INDY }
```

Oczywiście, jeśli włączono SGC\_CUSTOM\_INDY, ale w folderze źródłowym brakuje wymaganych jednostek niestandardowej wersji Indy, ta definicja kompilatora nie zadziała.

## Pakiet sgcIndy

Użycie niestandardowej wersji Indy nie jest ograniczone wyłącznie do komponentów sgcWebSockets. Niektórzy klienci chcą korzystać z nowych funkcji tej niestandardowej wersji Indy w standardowych komponentach Indy, takich jak SMTP, aby na przykład używać TLS 1.3 podczas wysyłania wiadomości e-mail, korzystania z serwerów FTP itp. Edycja eSeGeCe ALL-ACCESS oferuje dodatkowy pełny pakiet Indy ze wszystkimi tymi funkcjami. Ten pakiet, zwany „pakietem sgcIndy”, zawiera pełną bibliotekę Indy z obsługą openssl 1.1.1 i openssl 3.0.0. W związku z tym należy najpierw odinstalować aktualnie zainstalowaną bibliotekę Indy z IDE, a następnie zainstalować tę wersję. Proces instalacji pakietu sgcIndy jest dokładnie taki sam jak w przypadku każdej biblioteki Indy (nazwy jednostek nie są tutaj zmieniane).

Aby korzystać z bibliotek OpenSSL, należy ustawić zmienną globalną OPENSSL\_API\_VERSION na żądaną wersję API OpenSSL przed załadowaniem bibliotek OpenSSL. Ta zmienna globalna znajduje się w module IdSSLOpenSSLHeaders.

**Przykład:** użycie bibliotek openssl 1.1.1

```
OPENSSL_API_VERSION := opSSL_1_1;
```

## Jak używać pojedynczego pakietu sgcIndy

Przy jednoczesnym użyciu eSeGeCe ALL-ACCESS i pakietu sgcIndy w tej samej aplikacji, źródła mogą być zduplikowane, ponieważ wersja Enterprise sgcWebSockets używa niestandardowej wersji Indy z przemianowanymi jednostkami. Oznacza to, że na przykład jednostki takie jak sgcIdGlobal.pas i IdGlobal.pas będą kompilowane w tej samej aplikacji (pierwsza jest używana przy korzystaniu z dowolnego komponentu pakietu sgcWebSockets Enterprise, a druga przy korzystaniu z dowolnego komponentu pakietu sgcIndy, jak ftp, smtp...).

Aby uniknąć tego zachowania, pakiet sgcWebSockets można skonfigurować tak, aby używał zainstalowanej wersji sgcIndy, zachowując dostęp do wszystkich komponentów. W tym celu należy postępować zgodnie z poniższymi instrukcjami:

1. Otwórz plik sgcVer.inc, znajdujący się w folderze Source pakietu sgcWebSockets.



2. Wyłącz dyrektywę kompilatora: `SGC_CUSTOM_INDY`. Ta opcja informuje kompilator, że pliki zaczynające się od `sgcId*.pas` istnieją i muszą być używane podczas kompilowania pakietu `sgcWebSockets Enterprise`.

3. Włącz następujący kompilator: `SGC_INDY_LIB`. Ta opcja informuje kompilator, że pakiet `sgcIndy` jest zainstalowany i musi być używany podczas kompilacji pakietu.

```
{ $DEFINE SGC_INDY_LIB }
```

Przy powyższej konfiguracji pakiet `Enterprise sgcWebSockets` będzie korzystał z zainstalowanego pakietu `sgcIndy`, a wszystkie funkcje korzystające z tego pakietu (takie jak `HTTP/2`, `IOCP`, `OpenSSL 3.0` itp.) zostaną włączone.

# Zdarzenia WebSocket

Połączenia WebSocket mają następujące zdarzenia:

## OnConnect

Zdarzenie wywoływane po nawiązaniu nowego połączenia.

## OnDisconnect

Zdarzenie wywoływane po zamknięciu połączenia.

## OnError

Zdarzenie wywoływane, gdy połączenie napotka błąd.

## OnMessage

Zdarzenie wywoływane po odebraniu nowej wiadomości tekstowej.

## OnBinary

Zdarzenie wywoływane po odebraniu nowej wiadomości binarnej.

Domyślnie `sgcWebSockets` używa mechanizmu **asynchronicznego** do wywoływania tych zdarzeń. Gdy którekolwiek z tych zdarzeń jest wywoływane wewnętrznie, wiadomość jest kolejkowana i wysyłana przez system operacyjny, gdy jest to dozwolone. To zachowanie można zmienić za pomocą właściwości **NotifyEvents**; domyślnie wybrana jest wartość **neAsynchronous**. Jeśli zostanie zaznaczone **neNoSync**, zdarzenia będą wywoływane bez synchronizacji z głównym wątkiem (jeśli konieczna jest aktualizacja kontrolki VCL lub dostęp do zasobów współdzielonych, należy zaimplementować własną metodę synchronizacji).

**neNoSync** jest zalecane, gdy:

1. Konieczna jest obsługa dużej liczby wiadomości w bardzo krótkim czasie.
2. Projekt jest zbudowany jako aplikacja wiersza poleceń (brak zdarzeń bez ustawienia `neNoSync`).

3. Projekt jest biblioteką.

Jeśli nie, można użyć domyślnej wartości właściwości **neAsynchronous**.

# Parametry połączenia WebSocket

Obsługiwane przez

[TsgcWebSocketClient](#)

JavaScript

Czasami przydatne jest przekazanie parametrów od klienta do serwera podczas nawiązywania nowego połączenia WebSocket. Jeśli konieczne jest przekazanie parametrów do serwera, można użyć następującej właściwości:

## Opcje / Parametry

Domyślnie jest ustawione na '/', jeśli zachodzi potrzeba przekazania parametru, np. id=1, można ustawić tę właściwość na '/?id=1'

Po stronie serwera można obsługiwać parametry klienta przy użyciu następującego parametru:

```
void WSServerConnect(TsgcWSConnection *Connection);
{
    if (Connection->URL == "/?id=1")
    {
        HandleThisParameter;
    }
}
```

Używając języka JavaScript, można przekazywać parametry za pomocą adresu URL połączenia, przykład:

```
<script src="http://localhost/sgcWebSockets.js" type="text/javascript"></script>
<script type="text/javascript">var socket = new sgcWebSocket('ws://localhost/?id=1');</script>
```

## Używanie wewnątrz biblioteki DLL

Jeśli konieczna jest praca z bibliotekami DLL i `sgcWebSockets` (lub aplikacjami konsolowymi), właściwość **NotifyEvents** musi być ustawiona na **neNoSync**.

# Usługa Windows

Podczas uruchamiania biblioteki sgcWebSockets w usłudze systemu Windows właściwość **NotifyEvents** musi być ustawiona na **neNoSync**.

```
void __fastcall TService1::ServiceStart(TService *Sender, bool &Started)
{
    FServer = new TsgcWebsocketHTTPServer(nullptr);
    FServer->Port = 80;
    FServer->Active = true;
}
//-----
// OnServiceExecute
void __fastcall TService1::ServiceExecute(TService *Sender)
{
    while (!Terminated)
    {
        ServiceThread->ProcessRequests(true);
    }
}
//-----
// OnServiceStop
void __fastcall TService1::ServiceStop(TService *Sender, bool &Stopped)
{
    if (FServer != nullptr)
    {
        delete FServer;
        FServer = nullptr;
    }
}
```

# Test przeglądarki internetowej

TsgcWebSocketServer implementuje wbudowaną stronę internetową, na której można testować połączenie z serwerem WebSocket za pomocą wybranej przeglądarki internetowej.

Aby uzyskać dostęp do tej strony testowej, należy wpisać następujący adres URL:

```
http://host:port/sgcWebSockets.html
```

Przykład: jeśli serwer WebSocket skonfigurowano na adresie IP 127.0.0.1 i porcie 80, należy wpisać:

```
http://127.0.0.1:80/sgcWebSockets.html
```

Na tej stronie można przetestować następujące metody WebSocket:

Otwórz

Close

Status

Wyślij

Aby wyłączyć strony testowe HTML przeglądarki, wystarczy ustawić  
`TsgcWebSocketServer.Options.HTMLFiles = false;`

## Niestandardowe subprotokołu

Klient może zażądać, aby serwer używał określonego subprotokołu, podając jego nazwę podczas uzgadniania. Jeśli subprotokół zostanie określony, serwer musi uwzględnić jedną z wybranych wartości subprotokołu w swojej odpowiedzi, aby połączenie zostało nawiązane.

Aby utworzyć własny subprotokół, należy dziedziczyć po `TsgcWSProtocol_Client_Base` i `TsgcWSProtocol_Server_Base` w celu stworzenia niestandardowych subprotokołów.



# Uwierzytelnianie

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

Java script (\*obsługiwane jest wyłącznie uwierzytelnianie URL)

Specyfikacja WebSocket nie definiuje żadnej metody uwierzytelniania, a implementacje przeglądarek internetowych nie pozwalają na wysyłanie niestandardowych nagłówków przy nawiązywaniu nowych połączeń WebSocket.

Aby włączyć tę funkcję, należy uzyskać dostęp do następującej właściwości:

## Uwierzytelnianie / Włączone

sgcWebSockets implementuje 3 różne typy uwierzytelniania WebSocket:

**Sesja:** klient musi wykonać żądanie HTTP GET, przekazując nazwę użytkownika i hasło; jeśli uwierzytelnienie się powiedzie, serwer odpowiada identyfikatorem sesji. Przy użyciu tego identyfikatora sesji klient otwiera połączenie WebSocket, przekazując go jako parametr. Można użyć zwykłego żądania HTTP, aby uzyskać identyfikator sesji, przekazując nazwę użytkownika i hasło jako parametry.

`http://host:port/sgc/req/auth/session/:user/:password`

**przykład:** (user=admin, password=1234) --> `http://localhost/sgc/req/auth/session/admin/1234`

Zwraca token używany do połączenia z serwerem za pomocą połączeń WebSocket:

```
ws://localhost/sgc/auth/session/:token
```

**URL:** klient otwiera połączenie WebSocket, przekazując nazwę użytkownika i hasło jako parametry.

```
ws://host:port/sgc/auth/url/username/password
```

**przykład:** (user=admin, password=1234) --> http://localhost/sgc/auth/url/admin/1234

**Basic:** implementuje Basic Access Authentication, dotyczy wyłącznie WebSockets VCL (serwer i klient) oraz żądań HTTP (przeglądarki internetowe nie implementują tego typu uwierzytelniania). Gdy klient próbuje się połączyć, wysyła nagłówek zgodnie ze specyfikacją AUTH BASIC.

Można zdefiniować listę uwierzytelnionych użytkowników przy użyciu właściwości **Authentication/AuthUsers**. Każdy element należy zdefiniować według schematu: użytkownik=hasło. Przykład:

```
admin=admin
```

```
user=1234
```

```
....
```

Dostępne jest zdarzenie **OnAuthentication**, które umożliwia obsługę uwierzytelniania w przypadku, gdy użytkownik nie znajduje się na liście AuthUsers, klient nie wysyła żądania autoryzacji itp. Można sprawdzić parametry User i Password i, jeśli są poprawne, ustawić zmienną Authenticated na True. Przykład:

```
void WSServerAuthentication(TsgcWSCConnection *Connection, string aUser, string aPassword, bool &Authenticated)
{
    if ((aUser == "John") && (aPassword == "1234"))
    {
```

```
    Authenticated = true;  
  }  
}
```

# Bezpieczne połączenia

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

Przeglądarki internetowe

Obsługa SSL jest oparta na implementacji Indy, dlatego w celu korzystania z tej funkcji należy wdrożyć biblioteki openssl.

TsgcWebSocketClient obsługuje Microsoft SChannel, dlatego nie ma potrzeby wdrażania bibliotek OpenSSL dla systemów Windows 32 i 64-bitowych, jeśli w kliencie WebSocket wybrana jest opcja SChannel.

## Strona serwera

Aby włączyć tę funkcję, należy włączyć następującą właściwość:

**SSL / Włącz**

Istnieją inne właściwości, które należy zdefiniować:

**SSLOptions/ CertFile/ KeyFile/ RootCertFile:** do szyfrowania komunikacji WebSocket wymagany jest certyfikat w formacie .PEM.

**SSLOptions/ Password:** parametr opcjonalny, wymagany jedynie wówczas, gdy certyfikat jest chroniony hasłem.

**SSLOptions/ Port:** port używany w połączeniach SSL.

## Strona klienta

Aby włączyć tę funkcję, należy włączyć następującą właściwość:

### **TLS / Włącz**

## OpenSSL

Domyślnie komponenty klienta i serwera oparte na Indy korzystają z bibliotek OpenSSL podczas łączenia się z bezpiecznymi serwerami WebSocket.

Indy obsługuje wyłącznie API OpenSSL 1.0.2, dlatego API 1.1 nie jest obsługiwane. Jeśli sgcWebSockets zostanie skompilowane z naszą niestandardową biblioteką Indy, można korzystać z API 1.1 i wybrać wersję TLS 1.3. Wystarczy wybrać w właściwościach OpenSSL\_Options, której wersji API OpenSSL należy użyć:

- **oslAPI\_1\_0:** jest to domyślne API Indy; można używać standardowego pakietu Indy z bibliotekami openssl 1.0.2.
- **oslAPI\_1\_1:** wybierz tylko wtedy, gdy kompilujesz sgcWebSockets z naszą niestandardową biblioteką Indy (wydanie Enterprise). Zostaną użyte biblioteki OpenSSL 1.1.1.
- **oslAPI\_3\_0:** należy wybrać tylko w przypadku kompilowania sgcWebSockets z naszą niestandardową biblioteką Indy (Enterprise Edition). Korzysta z bibliotek openssl 3.0.0.
  - **ECDHE:** umożliwia włączenie ECDHE dla TLS 1.2 (bezpieczniejsze połączenia).

## Zdarzenia

Istnieją 2 zdarzenia, których można użyć do dostosowania ustawień SSL:

## OnSSLGetHandler

To zdarzenie jest wywoływane przed utworzeniem procedury obsługi SSL. Można tu utworzyć własną procedurę obsługi SSL (musi dziedziczyć po `TIdServerIOHandlerSSLBase` lub `TIdIOHandlerSSLBase`) i ustawić wymagane właściwości.

```
properties needed
void OnServerSSLGetHandler(TObject *Sender, TwsSSLHandler aType, ref TIdServerIOHandlerSSLBase
*aSSLHandler)
{
    TCustomSSLHandler aSSLHandler = new TCustomSSLHandler();
    ...
}
```

## OnSSLAfterCreateHandler

Jeśli żaden niestandardowy obiekt SSL nie został utworzony, jest on tworzony domyślnie przy użyciu procedury obsługi OpenSSL. Można uzyskać dostęp do właściwości procedury obsługi SSL i w razie potrzeby je zmodyfikować.

```
void OnSSLAfterCreateHandler(TObject *Sender, TwsSSLHandler aType, TIdServerIOHandlerSSLBase
*aSSLHandler)
{
    dynamic_cast <tidserveriohandlesslopenssl>*(aSSLHandler)->SSLOptions->Method = sslvTLSv1_2;
}
```

## Microsoft SChannel

Począwszy od wersji `sgcWebSockets 4.2.6`, zamiast biblioteki `openssl` można używać interfejsu `SChannel` (tylko dla systemu Windows 7 i nowszych). Oznacza to brak konieczności wdrażania bibliotek `openssl`. TLS 1.0 jest obsługiwany od systemu Windows 7, jednak jeśli wymagane są nowsze implementacje, takie jak TLS 1.2 w systemie Windows 7, należy włączyć TLS 1.1 i TLS 1.2 w rejestrze systemu Windows.

Wymaga Delphi 2010 Professional Edition (lub Enterprise Edition dla Delphi 7, 2007 i 2009).

# HeartBeat

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

[TsgcWebSocketClient](#)

W przypadku komponentów serwera automatycznie wysyła ping do wszystkich aktywnych połączeń WebSocket co x sekund.

W komponentach klienta automatycznie wysyła sygnał ping do serwera co określoną liczbę sekund.

HeartBeat ma następujące właściwości:

- **Enabled:** jeśli true, wysyła ping
- **Interval:** to wartość w sekundach, co ile zostanie wysłany sygnał ping. Przykład: jeśli wartość wynosi 10, sygnał ping będzie wysyłany co 10 sekund
- **Timeout:** czas oczekiwania na odpowiedź z serwera. Przykład: jeśli wartość wynosi 30, będzie czekać 30 sekund na odpowiedź przed zamknięciem połączenia.
- **HeartBeatType:** pozwala dostosować sposób działania HeartBeat
  - **hbtAlways:** wysyła ping co x sekund określonych w Interval.
  - **hbtOnlyIfNoMsgRcvInterval:** wysyła ping co x sekund tylko wtedy, gdy przez ostatnie x sekund określone we właściwości Interval nie odebrano żadnych wiadomości.

## Dostosowywanie HeartBeat

Komponenty klienta i serwera umożliwiają dostosowanie mechanizmu HeartBeat do wysyłania niestandardowych pingów i sprawdzania, czy połączenie jest nadal aktywne. Zdarzenie **OnBeforeHeartBeat** zostało stworzone właśnie w tym celu — pozwala wysłać niestandardową wiadomość i/lub pominąć standardowy ping.

**Przykład:** wysyłaj wiadomość tekstową jako ping co 30 sekund.

```
void OnBeforeHeartBeat(TObject *Sender; const TsgcWSConnection *Connection; ref bool Handled)
{
    Connection->WriteData("ping");
    Handled = true;
}
```



# WatchDog

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

[TsgcWebSocketClient](#)

## Serwer

Na komponentach serwera należy automatycznie restartować serwer po nieoczekiwanym zamknięciu. Aby sprawdzać, czy serwer jest aktywny co 60 sekund, należy ustawić następujące właściwości:

```
WatchDog.Enabled = true;
```

```
WatchDog.Interval = 60;
```

```
WatchDog.Attempts = 0;
```

WatchDog.Monitor umożliwia weryfikację, czy nowi klienci mogą łączyć się z serwerem. Odbywa się to przez wewnętrzny klient, który próbuje otworzyć połączenie WebSocket z serwerem; jeśli próba się nie powiedzie, serwer jest restartowany. Aby monitorować, czy klienci mogą łączyć się z serwerem z limitem czasu 10 sekund, należy ustawić następujące właściwości:

```
WatchDog.Enabled = true;
```

```
WatchDog.Interval = 60;
```

```
WatchDog.Attempts = 0;
```

```
WatchDog.Monitor.Enabled = true;
```

```
WatchDog.Monitor.Timeout = 10;
```

## Klient

W komponentach klienckich automatyczne ponowne połączenie z serwerem po nieoczekiwanym rozłączeniu. Aby ponownie łączyć się co 10 sekund po rozłączeniu, wystarczy ustawić następujące właściwości:

```
WatchDog.Enabled = true;
```

```
WatchDog.Interval = 10;
```

```
WatchDog.Attempts = 0;
```

# Logi

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

To przydatna funkcja umożliwiająca debugowanie połączeń WebSocket. Aby ją włączyć, należy uzyskać dostęp do następującej właściwości:

### **LogFile/ Enabled**

Po włączeniu, każde nowe połączenie będzie rejestrowane w pliku tekstowym. W komponencie serwera, jeśli plik nie istnieje, zostanie utworzony, ale nie można uzyskać do niego dostępu do momentu zamknięcia serwera. Jeśli chcesz otworzyć plik dziennika podczas aktywności serwera, plik dziennika musi zostać utworzony przed uruchomieniem serwera.

### **Przykład:**

```
127.0.0.1:49854 Stat Connected.
```

```
127.0.0.1:49854 Recv 09/11/2013 11:17:03: GET / HTTP/1.1
```

```
Upgrade: websocket
```

```
Connection: Upgrade
```

```
Host: 127.0.0.1:5414
```

```
Origin: http://127.0.0.1:5414
```

```
Pragma: no-cache
```

```
Cache-Control: no-cache
```

```
Sec-WebSocket-Key: 1n598ldHs9SdRfxUK8u4Vw==
```

```
Sec-WebSocket-Version: 13
```

```
Sec-WebSocket-Extensions: x-webkit-deflate-frame
```

```
127.0.0.1:49854 Sent 09/11/2013 11:17:03: HTTP/1.1 101 Switching Protocols
```

```
Upgrade: websocket
```

```
Connection: Upgrade
```

```
Sec-WebSocket-Accept: gDuzFRzwHBc18P1CfinlvKv1BJc=
```

```
127.0.0.1:49854 Stat Disconnected.
```

```
0.0.0.0:0 Stat Disconnected.
```

### **Wiadomości WebSocket**

Ramki WebSocket mogą być maskowane, co oznacza, że zarejestrowana wiadomość nie może być odczytana.

Gdy właściwość **LogFile.UnMaskFrames** = True (domyślnie true)

- Wiadomości **wysłane** przez **klienta WebSocket** są zapisywane jako **niezamaskowane**.
- Wiadomości **odebrane** przez **serwer WebSocket** są zapisywane **zamaskowane** i **odmaskowane** (powodem jest to, że podczas odczytu bufora przez gniazdo protokół wiadomości nie jest jeszcze znany, dlatego zapisywane są obie wersje).

# HTTP

Obsługiwane przez

[TsgcWebSocketHTTPServer](#)

**TsgcWebSocketHTTPServer** to komponent umożliwiający obsługę połączeń WebSocket i HTTP na tym samym porcie. Jest to bardzo przydatne w przypadku konieczności skonfigurowania serwera, na którym włączony jest wyłącznie port HTTP (zazwyczaj port 80). Komponent ten obsługuje wszystkie funkcje [TsgcWebSocketServer](#) i umożliwia serwowanie stron HTML.

Możliwe jest **serwowanie statycznych stron HTML** przy użyciu właściwości **DocumentRoot** — na przykład: jeśli plik test.html zostanie zapisany w katalogu "C:\inetpub\wwwroot" i właściwość **DocumentRoot** zostanie ustawiona na "C:\inetpub\wwwroot", dostęp klienta do pliku test.html będzie obsługiwany automatycznie, na przykład:

<http://localhost/test.html>

Można też **dynamicznie serwować HTML lub inne zasoby** za pomocą kodu. W tym celu istnieje zdarzenie **OnCommandGet**, wywoływane za każdym razem, gdy klient żąda nowej strony HTML, obrazu, pliku javascript itp. Należy sprawdzić, który dokument żąda klient (używając `ARequestInfo.Document`), i wysłać odpowiedź do klienta (używając `AResponseInfo.ContentText` dla treści odpowiedzi, `AResponse.ContentType` dla typu odpowiedzi oraz `AResponseInfo.ResponseNo` z kodem odpowiedzi, zwykle 200). Przykład:

```
void WSServerCommandGet(TIdContext *AContext, TIdHTTPRequestInfo *ARequestInfo,
    TIdHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/myfile.js")
    {
        AResponseInfo->ContentText = "<script type='text/javascript'>alert('Hello!');</script>";
        AResponseInfo->ContentType = "text/javascript";
        AResponseInfo->ResponseNo = 200;
    }
}
```

# Broadcast i kanały

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

Metoda **Broadcast** domyślnie wysyła wiadomość do **wszystkich połączonych klientów**, jednak można użyć argumentu **channels** w celu filtrowania i **wysyłania wiadomości wyłącznie do klientów zasubskrybowanych** do danego kanału.

**Przykład:** serwer obsługuje 2 typy połączonych klientów — urządzenia stacjonarne i mobilne — dlatego można utworzyć 2 kanały: "desktop" i "mobile".

Jeśli w zdarzeniu OnConnect serwera można zidentyfikować, czy klient jest mobilny, można postąpić w sposób następujący:

```
void OnServerConnect(TsgcWSConnection *Connection)
{
    if (desktop == true)
    {
        dynamic_cast(Connection)->DoSubscribe("desktop");
    }
}
```

Należy najpierw rzutować Connection na TsgcWSConnectionServer, aby uzyskać dostęp do metod subskrypcji. Jeśli połączenie spełnia warunki filtru, zostanie zasubskrybowane do kanału desktop. Subskrypcji do kanału można dokonać w dowolnym zdarzeniu. Na przykład można poprosić klienta o przesłanie informacji, czy jest urządzeniem mobilnym, a następnie wysłać wiadomość od klienta do serwera z informacjami o kliencie. Wówczas można wysyłać broadcast wyłącznie do połączeń desktop:

```
Server->Broadcast("Your text message", "desktop");
```

Jeśli istnieje 100 połączeń, z których 30 to urządzenia mobilne, wiadomość zostanie wysłana wyłącznie do pozostałych 70.

# Powiązania

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

Serwery mają zazwyczaj więcej niż jeden adres IP. Gdy serwer WebSocket zostaje włączony z portem nasłuchiwania 80, podczas uruchamiania próbuje nasłuchiwać na porcie 80 na WSZYSTKICH adresach IP. Jeśli serwer ma 3 adresy IP, port 80 zostanie powiązany z każdym z nich.

Powiązania (Bindings) umożliwiają zdefiniowanie dokładnych adresów IP i portów używanych przez serwer. Na przykład, jeśli wymagane jest nasłuchiwanie na porcie 80 dla IP 127.0.0.1 (adres wewnętrzny) i 80.254.21.11 (adres publiczny), można to skonfigurować przed aktywowaniem serwera:

```
bind = WSServer->Bindings->Add();
{
    bind->Port = 80;
    bind->IP = "127.0.0.1";
}
bind = WSServer->Bindings->Add();
{
    bind->Port = 80;
    bind->IP = "80.254.21.11";
}
```

# Wysyłanie dużych plików

Obsługiwane przez

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

Gdy klient HTTP wysyła strumień **multipart/form-data**, strumień jest zapisywany przez serwer w pamięci. W przypadku dużych plików serwer może zgłosić wyjątek **braku pamięci**. Aby uniknąć tych wyjątków, serwer posiada właściwość `HTTPUploadFiles`, w której można skonfigurować sposób obsługi strumieni POST: w pamięci lub jako strumienie plikowe. Jeśli strumienie są obsługiwane jako strumienie plikowe, odebrane strumienie są zapisywane bezpośrednio na dysku twardym, co pozwala uniknąć problemów z pamięcią.

Aby skonfigurować serwer do zapisywania strumieni multipart/form-data jako strumieni plikowych, należy wykonać następujące kroki:

1. Ustawić właściwość `HTTPUploadFiles.StreamType = pstFileStream`. Przy tej konfiguracji serwer będzie zapisywał strumienie na dysku twardym.
2. Można skonfigurować **minimalny rozmiar w bajtach**, od którego pliki będą zapisywane jako strumień plikowy. Domyślna wartość wynosi zero, co oznacza, że wszystkie strumienie będą zapisywane jako strumień plikowy.
3. Folder, w którym strumienie są zapisywane, określa właściwość `SaveDirectory`. Jeśli nie zostanie ustawiona, strumienie będą zapisywane w tym samym folderze, w którym znajduje się aplikacja.
4. Gdy klient wysyła multipart/form-data, zawartość jest zakodowana wewnątrz granic (boundaries). Jeśli właściwość `RemoveBoundaries` jest włączona, zawartość granic zostanie wyodrębniona automatycznie po odebraniu pełnego strumienia.

## Przykładowy kod

Najpierw należy utworzyć nową instancję serwera i ustawić zapisywanie strumieni jako strumienie plikowe.

```
TsgcWebSocketHTTPServer *oServer = new TsgcWebSocketHTTPServer();
oServer->Port = 5555;
oServer->HTTPUploadFiles->StreamType = pstFileStream;
oServer->Active = true;
```

Następnie należy utworzyć nowy plik HTML o następującej konfiguracji

```
<html>

<head><title>sgcWebSockets - Upload Big File</title></head>

<body>

<form action="http://127.0.0.1:5555/file" method="post"
enctype="multipart/form-data" accept-charset="UTF-8">

<input type="file" name="file_1" />
```



```
<input type="submit" />

</form>

</body>

</html>
```

Na koniec należy otworzyć plik HTML w przeglądarce internetowej i wysłać plik do serwera. Serwer utworzy nowy strumień plikowy z rozszerzeniem ".sgc\_ps", a po pełnym odebraniu strumienia wyodrębni plik z granic.

## Zdarzenia

Dostępne są 2 zdarzenia umożliwiające dostosowanie procesu przesyłania pliku (wymaga włączenia właściwości `HTTPUploadFiles.RemoveBoundaries`)

### OnHTTPUploadBeforeSaveFile

Zdarzenie jest wywoływane PRZED zapisaniem pliku i umożliwia dostosowanie nazwy odebranego pliku.

```
void OnHTTPUploadBeforeSaveFileEvent(TObject *Sender, String &aFileName, String &aFilePath)
{
    if (aFileName == "test.jpg")
    {
        aFileName = "custom_test.jpg";
    }
}
```

### OnHTTPUploadAfterSaveFile

Zdarzenie jest wywoływane PO zapisaniu pliku i pozwala uzyskać informację o nazwie zapisanego pliku.

```
void OnHTTPUploadBeforeSaveFileEvent(TObject *Sender, string aFileName, string aFilePath)
{
    DoLog("File Received: " + aFileName);
}
```

### OnHTTPUploadReadInput

Zdarzenie jest wywoływane, gdy dekodery odczytuje wartość wejściową inną niż wejście plikowe (na przykład gdy formularz zawiera zmienne takie jak nazwa, data...).

```
void OnHTTPUploadReadInputEvent(TObject *Sender, string aName, string aValue)
{
    DoLog("Input value Received: " + aName + ":" + aValue);
}
```

# Kompresja

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

Przeglądarki internetowe, takie jak Chrome

Jest to funkcja, która działa bardzo dobrze w sytuacjach, gdy trzeba wysyłać duże ilości danych, zwykle jako wiadomość binarną, ponieważ kompresuje wiadomość WebSocket przy użyciu protokołu „PerMessage\_Deflate”, który jest obsługiwany przez niektóre przeglądarki, takie jak Chrome.

Aby włączyć tę funkcję, należy aktywować następującą właściwość:

**Extensions/ PerMessage\_Deflate / Enabled**

Gdy klient próbuje połączyć się z serwerem WebSocket i ta właściwość jest włączona, wysyła nagłówek z tą właściwością. Jeśli serwer obsługuje tę funkcję, odsyła klientowi odpowiedź z aktywowanym protokołem i wszystkie wiadomości będą kompresowane. Jeśli serwer nie obsługuje tej funkcji, wszystkie wiadomości będą wysyłane bez kompresji.

W przeglądarkach internetowych nie ma potrzeby wykonywania żadnych czynności. Jeśli to rozszerzenie jest obsługiwane, zostanie użyte automatycznie; w przeciwnym razie komunikaty będą wysyłane bez kompresji.

Jeśli wiadomości WebSocket są małe, lepiej nie włączać tej właściwości, gdyż kompresja/dekompresja wiadomości zużywa cykle procesora. Jednak przy transferze dużych ilości danych zauważalne będzie zwiększenie szybkości wymiany wiadomości.

# Flash

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

WebSockets są natywnie obsługiwane przez szeroką gamę przeglądarek internetowych (zob. <http://caniuse.com/websockets>), jednak niektóre starsze wersje nie implementują WebSockets (np. Internet Explorer 6, 7, 8 lub 9). Dla wszystkich tych przeglądarek, które nie obsługują WebSockets, można włączyć **Flash Fallback**.

Prawie wszystkie inne lub starsze przeglądarki obsługują Flash po zainstalowaniu Adobe Flash Player. Aby obsługiwać połączenie Flash, należy **otworzyć port 843** na serwerze, ponieważ Flash używa tego portu ze względów bezpieczeństwa w celu sprawdzenia dostępu między domenami. Jeśli port 843 jest niedostępny, komponent czeka 3 sekundy i próbuje połączyć się z domyślnym portem serwera.

Flash jest stosowany tylko wtedy, gdy przeglądarka nie obsługuje natywnie WebSockets. Jeśli więc Flash Fallback jest włączony po stronie serwera, a przeglądarka obsługuje WebSockets natywnie, nadal będzie używać WebSockets jako transportu.

Aby włączyć Flash Fallback, należy uzyskać dostęp do właściwości **FallBack** / **Flash** na serwerze i ją **włączyć**. Dostępne są 2 dodatkowe właściwości:

**1. Domena:** jeśli wymagane jest ograniczenie połączeń flash do jednej lub wielu domen (domyślnie dozwolone są wszystkie domeny). Przykład: zezwolenie na dostęp do domeny swf.example.com

swf.example.com

**2. Porty:** jeśli konieczne jest ograniczenie połączeń flash do jednego lub wielu portów (domyślnie dozwolone są wszystkie porty). Przykład: poniższe ustawienie zezwala na dostęp do portów 123, 456, 457 i 458

123,456-458

Połączenia Flash obsługują wyłącznie wiadomości tekstowe, wiadomości binarne nie są obsługiwane.

# Obiekty niestandardowe

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

[TsgcWebSocketClient](#)

Za każdym razem, gdy nawiązywane jest nowe połączenie WebSocket, sgcWebSockets tworzy klasę [TsgcWSCConnection](#), w której można uzyskać dostęp do właściwości takich jak identyfikator, liczba odebranych/wysłanych bajtów, adres IP klienta itd. Dostępna jest też właściwość **Data**, w której można przechowywać obiekty w pamięci, takie jak obiekty dostępu do bazy danych czy sesji.

```
// You can create a new class called MyClass and create some properties, example:
class TMyClass
{
private:
    bool FRegistered;
    String FUser;
public:
    __property bool Registered = {read=FRegistered, write=FRegistered};
    __property String User = {read=FUser, write=FUser};
};

// Then, when a new client connects, OnConnect Event, create a new TMyClass and Assign to Data:
void WSServerConnect(TsgcWSCConnection *Connection)
{
    Connection->Data = new TMyClass();
}

// Every time a new message is received by the server, you can access your custom object using
// Connection.Data property.
void WSServerMessage(TsgcWSCConnection *Connection, string Text)
{
    if (dynamic_cast<TMyClass*>(Connection->Data)->Registered == true)
    {
        DoSomeStuff();
    }
}

// When a connection is closed, you may free your object:
void TfrmServerChat::WSServerDisconnect(TsgcWSCConnection *Connection, int Code)
{
    TMyClass *oMyClass = dynamic_cast<TMyClass*>(Connection->Data);
    if (oMyClass != 0)
    {
        delete oMyClass;
        Connection->Data = NULL;
    }
}
```

}  
}

# Grupy

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

**sgcWebSockets** zapewnia zaawansowaną metodę **rozsyłania wiadomości do określonych podzbiorów połączonych klientów**. Grupa może mieć dowolną liczbę klientów, a klient może należeć do dowolnej liczby grup. Grup nie trzeba tworzyć jawnie; w praktyce grupa jest tworzona automatycznie przy pierwszym podaniu jej nazwy w wywołaniu metody **Groups.Add**.

Po dodaniu użytkownika do grupy za pomocą metody **Groups.Add** użytkownik odbiera wiadomości kierowane do tej grupy przez czas trwania bieżącego połączenia.

## Dodawanie i usuwanie użytkowników

Aby dodać lub usunąć użytkowników z grupy, należy wywołać metody **Add** lub **Remove**, przekazując nazwę grupy i klasę **TsgcWSConnection**. Nie ma potrzeby ręcznego usuwania użytkownika z grupy po zakończeniu połączenia.

Poniższy przykład przedstawia metodę **Groups.Add**.

```
void OnConnect(TsgcWSConnection *Connection)
{
    TsgcWebSocketServer1->Groups->Add("Room1", Connection);
}
```

## Wysyłanie wiadomości do grupy

Wiadomość można wysłać do wszystkich członków grupy, jak pokazano w poniższym przykładzie.

```
TsgcWebSocketServer1->Groups->Group["Room1"]->Broadcast("Hello Members of Room1");
```

Można też wysłać wiadomość do wszystkich grup, których nazwa zaczyna się od „Room” (jeśli istnieją Room1, Room2, Room3..., ich użytkownicy otrzymają wiadomość).

```
TsgcWebSocketServer1->Groups->Broadcast("Room*", "Hello Members of Room");
```

## Zdarzenia

Dostępne są 2 zdarzenia służące do obsługi grup i klientów za każdym razem, gdy nowy klient jest dodawany do grupy lub gdy jest z niej usuwany:

OnClientAdded

OnClientRemoved

Przykład: wysyłanie wiadomości do grupy po opuszczeniu jej przez członka.

```
TsgcWebSocketServer1->Groups->OnClientRemoved() = OnClientRemovedEvent();

void OnClientRemovedEvent(TObject *Sender, const TsgcWSServerGroupItem *aGroup,
    const TsgcWSConnection *aConnection)
{
    aGroup->Broadcast("Client " + aConnection->Guid + " has disconnected");
}
```



# IOCP

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

**\*Wymaga niestandardowej wersji Indy.**

IOCP dla systemu Windows to API umożliwiające obsługę tysięcy połączeń przy użyciu ograniczonej puli wątków, zamiast przydzielania jednego wątku na połączenie, jak robi to Indy domyślnie.

Aby włączyć IOCP dla serwerów Indy, przejdź do właściwości **IOHandlerOptions** i wybierz **iohIOCP** jako typ IOHandler.

```
Server->IOHandlerOptions->IOHandlerType = iohIOCP;  
Server->IOHandlerOptions->IOCP->IOCPThreads = 0;  
Server->IOHandlerOptions->IOCP->WorkOpThreads = 0;
```

**IOCPThreads** to wątki używane do asynchronicznych żądań IOCP (operacje nakładkowe); domyślnie wartość wynosi zero, co oznacza, że liczba wątków jest obliczana na podstawie liczby procesorów (z wyjątkiem Delphi 7 i 2007, gdzie liczba wątków jest ustawiona na 32, ponieważ funkcja `cpucount` nie jest obsługiwana).

Właściwość **WorkOpThreads** powinna być włączona wyłącznie wtedy, gdy połączenia mają być zawsze przetwarzane w tym samym wątku. Przy korzystaniu z IOCP żądania są przetwarzane przez pulę wątków i każde żądanie (dla tego samego połączenia) może być obsługiwane przez inny wątek. Jeśli wymagane jest obsługiwanie każdego połączenia w tym samym wątku, należy w **WorkOpThreads** ustawić liczbę wątków używanych do obsługi tych żądań. Wpływa to na wydajność serwera i zaleca się ustawienie wartości większej od zera wyłącznie w przypadku, gdy funkcja ta jest niezbędna.

Włączenie IOCP dla serwerów Windows jest zalecane, gdy wymagana jest obsługa tysięcy połączeń. Jeśli serwer obsługuje maksymalnie 100 jednoczesnych połączeń, można pozostać przy domyślnym modelu wątkowym Indy.

**Zdarzenie OnDisconnect nie jest wywoływane**

IOCP działa inaczej niż domyślny IOHandler biblioteki Indy. W przypadku domyślnego IOHandler Indy każde połączenie jest obsługiwane przez osobny wątek, który działa przez cały czas, sprawdzając, czy połączenie jest aktywne; w razie rozłączenia zostaje ono zgłoszone w krótkim czasie.

IOCP działa inaczej: istnieje pula wątków obsługująca wszystkie połączenia, zamiast modelu 1 wątek = 1 połączenie, jak to robi domyślnie Indy. W IOCP jedynym sposobem wykrycia aktywności połączenia jest próba zapisu do gniazda. Jeśli wystąpi błąd, oznacza to, że połączenie zostało zamknięte. Dostępne są 2 opcje wykrywania rozłączeń:

1. Jeśli jest używany komponent **TsgcWebSocketClient**, można go włączyć we właściwości Options, ustawiając **CleanDisconnect := True** (domyślnie wyłączone). Po włączeniu, przed rozłączeniem klienta wysyłana jest wiadomość informująca serwer o rozłączeniu, dzięki czemu serwer odbierze tę wiadomość i zostanie wywołane zdarzenie OnDisconnect.
2. Można włączyć funkcję **heartbeat** po stronie **serwera**, na przykład co 60 sekund — serwer będzie wówczas próbował wysyłać ping do wszystkich podłączonych klientów, a w przypadku wykrycia rozłączonego klienta zostanie wywołane zdarzenie OnDisconnect.

# EPOLL

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

**\*Wymaga sgcWebSockets Enterprise Edition.**

EPOLL dla systemu Linux to API umożliwiające obsługę tysięcy połączeń przy użyciu ograniczonej puli wątków zamiast stosowania jednego wątku na połączenie, jak domyślnie działa Indy.

Aby włączyć EPOLL dla serwerów Indy, należy przejść do właściwości **IOHandlerOptions** i wybrać **ioHIEPOLL** jako typ IOHandler.

```
Server->IOHandlerOptions->IOHandlerType = ioHIEPOLL;  
Server->IOHandlerOptions->EPOLL->EPOLLThreads = 0;  
Server->IOHandlerOptions->EPOLL->WorkOpThreads = 0;
```

**EPOLLThreads** to wątki używane do asynchronicznych żądań EPOLL (operacje nakładające się); domyślnie wartość wynosi zero, co oznacza, że liczba wątków jest obliczana na podstawie liczby procesorów (z wyjątkiem Delphi 7 i 2007, gdzie liczba wątków wynosi 32, ponieważ funkcja cpucount nie jest obsługiwana). Liczbę wątków można dostosować ręcznie.

**WorkOpThreads** należy włączyć wyłącznie wtedy, gdy wymagane jest, aby połączenia były zawsze przetwarzane w tym samym wątku. W przypadku korzystania z EPOLL żądania są przetwarzane przez pulę wątków i każde żądanie (dla tego samego połączenia) może być przetwarzane w innym wątku. Jeśli konieczna jest obsługa każdego połączenia w tym samym wątku, należy w polu WorkOpThreads ustawić liczbę wątków używanych do obsługi tych żądań. Ma to wpływ na wydajność serwera i zalecane jest ustawienie wartości większej od zera tylko wtedy, gdy ta funkcja jest wymagana.

Włączenie EPOLL dla serwerów Linux jest zalecane, gdy wymagana jest obsługa tysięcy połączeń. Jeśli serwer obsługuje maksymalnie 100 równoczesnych połączeń, można pozostać przy domyślnym modelu wątkowym Indy.

**Zdarzenie OnDisconnect nie jest wywoływane**

EPOLL działa inaczej niż domyślny IOHandler Indy. Przy domyślnym IOHandler Indy każde połączenie działa w osobnym wątku, a wątki te działają przez cały czas i sprawdzają, czy połączenie jest aktywne, dlatego rozłączenie jest zgłaszane w krótkim czasie.

EPOLL działa inaczej — istnieje pula wątków obsługująca wszystkie połączenia, zamiast modelu 1 wątek = 1 połączenie stosowanego domyślnie przez Indy. W przypadku EPOLL jedynym sposobem wykrycia, czy połączenie jest nadal aktywne, jest próba zapisu do gniazda sieciowego; błąd oznacza, że połączenie zostało zamknięte. Istnieją 2 opcje wykrywania rozłączeń:

1. Jeśli jest używany komponent **TsgcWebSocketClient**, można go włączyć we właściwości Options, ustawiając **CleanDisconnect := True** (domyślnie wyłączone). Po włączeniu, przed rozłączeniem klienta wysyłana jest wiadomość informująca serwer o rozłączeniu, dzięki czemu serwer odbierze tę wiadomość i zostanie wywołane zdarzenie OnDisconnect.
2. Można włączyć funkcję **heartbeat** po stronie **serwera**, na przykład co 60 sekund — serwer będzie wówczas próbował wysyłać ping do wszystkich podłączonych klientów, a w przypadku wykrycia rozłączonego klienta zostanie wywołane zdarzenie OnDisconnect.

## Limit połączeń Linux

Aby zwiększyć liczbę jednocześnie otwartych połączeń, należy użyć następującego polecenia

```
ulimit -n 10000
```

Poprzednie polecenie ustawia maksymalną liczbę otwartych deskryptorów plików na 10000

# ALPN

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

**\*Wymaga niestandardowej wersji Indy.**

Application-Layer Protocol Negotiation (ALPN) to rozszerzenie Transport Layer Security (TLS) do negocjacji protokołów warstwy aplikacji. ALPN umożliwia warstwie aplikacji negocjację protokołu, który ma być używany przez bezpieczne połączenie, bez dodatkowych rund i niezależnie od protokołów warstwy aplikacji. Jest wymagane przez bezpieczne połączenia HTTP/2, które poprawiają kompresję stron internetowych i zmniejszają opóźnienia w porównaniu z HTTP/1.x.

## Klient

W opcji `TLSOptions.ALPNProtocols` można skonfigurować protokoły obsługiwane przez klienta. Gdy klient łączy się z serwerem, protokoły te są wysyłane podczas początkowego uzgadniania TLS „Client Hello”; klient podaje listę obsługiwanych przez siebie protokołów, a serwer wybiera, który protokół zostanie użyty, jeśli którykolwiek.

Wybrany przez serwer protokół można sprawdzić, uzyskując dostęp do właściwości `ALPNProtocol` obiektu `TsgcWSConnectionClient`.

## Server

Gdy pojawi się nowe połączenie TLS, wywoływane jest zdarzenie `OnSSLALPNSelect`, w którym można uzyskać dostęp do listy protokołów obsługiwanych przez klienta i wybrać jeden z nich po stronie serwera.

Jeśli żaden protokół nie jest obsługiwany, parametr `aProtocol` może pozostać pusty.

```
// Client
void OnClientConnect(TsgcWSConnection *Connection)
{
    string vProtocol = "";
    vProtocol = dynamic_cast<TsgcWSConnectionClient*>(Connection)->ALPNProtocol;
}

// Server
void OnSSLALPNSelect(TObject *Sender, TStringList *aProtocols, String &aProtocol)
{
    for (int i = 0; i < aProtocols->Count; i++)
    {
        if (aProtocols->Strings[i] == "h2")
        {
            aProtocol = "h2";
            break;
        }
    }
}
```

# Przekazywanie żądań HTTP

## Obsługiwane przez

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

[TsgcWSHTTPWebBrokerBridgeServer](#)

[TsgcWSHTTP2WebBrokerBridgeServer](#)

[TsgcWSServer\\_HTTPAPI\\_WebBrokerBridge](#)

Można skonfigurować serwer do przekazywania wybranych żądań HTTP do innego serwera. Jest to bardzo przydatne, gdy działa więcej niż jeden serwer i tylko jeden z nich nasłuchuje pod publicznym adresem.

**Przykład:** można skonfigurować serwer tak, aby przekazywał do innego serwera wszystkie żądania kierowane na /internal, podczas gdy pozostałe żądania obsługuje serwer sgcWebSockets.

Użyj zdarzenia **OnBeforeForwardHTTP**, aby sprawdzić, czy żądany adres URL powinien zostać przekierowany, a jeśli tak, ustaw adres URL przekierowania.

**Przykład:** jeśli chcesz przekazać wszystkie żądania dotyczące dokumentu "/internal" do serwera "localhost:8080", wykonaj następujące czynności:

```
void OnBeforeForwardHTTP(TsgcWSConnection *Connection, TIdHTTPRequestInfo *ARequestInfo,
    TsgcWSServerForwardHTTP *aForward)
{
    if (ARequestInfo->Document == "/internal")
    {
        aForward->Enabled = true;
        aForward->URL = "http://localhost:8080";
    }
}
```

## Inne opcje

Przy przekazywaniu żądania HTTP dostępne są następujące dodatkowe opcje:

1. Domyślnie żądanie jest przekazywane przy użyciu oryginalnego dokumentu. **Przykład:** jeśli żądanie `http://localhost:8080/internal` jest przekazywane do serwera wewnętrznego `http://localhost:5555`, przekazany adres URL będzie miał postać `http://localhost:5555/internal`. Można jednak zmodyfikować dokument, używając właściwości **Document** obiektu **Forward** (domyślnie używa tego samego dokumentu co oryginalne żądanie).

```
aForward.Document = "/NewInternal"
```

2. Jeśli przekazywane jest bezpieczne połączenie HTTP (HTTPS), można dostosować opcje SSL/TLS we właściwości **TLSTOptions** obiektu **Forward**. **Przykład:** ustawienie wersji TLS

```
aForward.TLSTOptions.Version = tls1_2
```

3. Następujące właściwości mogą być użyte do dostosowania żądania HTTP:

- **QueryParams:** parametry po adresie dokumentu, na przykład: `'id=1&user=2'`.
- **Host:** określa hosta i numer portu serwera, do którego kierowane jest żądanie. Przykład: `www.esegece.com:443`
- **Origin:** źródło żądania (schemat, nazwa hosta i port). Przykład: `https://www.esegece.com/document`.
- **LogFilename:** nazwa pliku, w którym będą przechowywane żądania/odpowiedzi.
- **NoCache:** jeśli żądanie nie może używać pamięci podręcznej przeglądarki, domyślnie włączone.
- **CustomHeaders:** lista niestandardowych nagłówków, które zostaną dodane do żądania. Przykład: `CustomHeaders.Add('X-ReverseProxy-Host: http://127.0.0.1:8888/test');`



# Quality Of Service

Obsługiwane przez

TsgcWSPServer\_sgc

TsgcWSPClient\_sgc

TsgcWSPClient\_MQTT

JavaScript

SGC Default Protocol i MQTT implementują QoS (Quality of Service) dla dostarczania wiadomości. Istnieją 3 różne typy:

**Poziom 0:** „Co najwyżej raz”, gdzie wiadomości są dostarczane zgodnie z najlepszymi możliwościami sieci TCP/IP. Możliwa jest utrata lub duplikacja wiadomości. Poziom ten może być stosowany na przykład w przypadku danych z czujników otoczenia, gdzie pominięcie pojedynczego odczytu nie ma znaczenia, ponieważ kolejny zostanie opublikowany wkrótce potem.

**Poziom 1:** „Co najmniej raz” — wiadomości są gwarantowanie dostarczane, jednak mogą wystąpić duplikaty.

**Poziom 2:** „Dokładnie raz”, gdzie wiadomości mają zagwarantowane dotarcie dokładnie jeden raz. Tego poziomu można używać na przykład w systemach rozliczeniowych, gdzie zduplikowane lub utracone wiadomości mogłyby prowadzić do naliczenia nieprawidłowych opłat.

## Poziom 0

Wiadomość jest dostarczana zgodnie z najlepszymi możliwościami sieci TCP/IP. Odpowiedź nie jest oczekiwana i protokół nie definiuje semantyki ponawiania. Wiadomość dociera do serwera raz lub wcale.

Poniższa tabela przedstawia przepływ protokołu dla poziomu QoS 0.

| Klient  | Wiadomość i kierunek | Serwer                                                 |
|---------|----------------------|--------------------------------------------------------|
| QoS = 0 | PUBLISH<br>----->    | <b>Akcja:</b> Publikowanie wiadomości do subskrybentów |

## Poziom 1

Odebranie wiadomości przez serwer jest potwierdzane wiadomością ACKNOWLEDGEMENT. W przypadku zidentyfikowanej awarii łącza komunikacyjnego lub urządzenia nadającego albo gdy wiadomość potwierdzająca nie zostanie odebrana po określonym czasie, nadawca ponownie wysyła wiadomość. Wiadomość dociera do serwera co najmniej raz.

Wiadomość z poziomem QoS 1 ma w wiadomości identyfikator.

Poniższa tabela przedstawia przepływ protokołu QoS poziomu 1.

| Klient                                                           | Wiadomość i kierunek    | Serwer                                                                                                                                                   |
|------------------------------------------------------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| QoS = 1<br>Message ID = x<br><b>Action:</b> Przechowaj wiadomość | PUBLISH<br>----->       | <b>Akcje:</b> <ul style="list-style-type: none"><li>• Zapisz wiadomość</li><li>• Opublikuj wiadomość do subskrybentów</li><li>• Usuń wiadomość</li></ul> |
| <b>Akcja:</b> Odrzuć wiadomość                                   | POTWIERDZENIE<br><----- |                                                                                                                                                          |

Jeśli klient nie otrzyma komunikatu ACKNOWLEDGMENT (w czasie zdefiniowanym w aplikacji lub po wykryciu awarii i ponownym uruchomieniu sesji komunikacyjnej), może ponownie wysłać wiadomość PUBLISH.

## Poziom 2

Dodatkowe przepływy protokołu powyżej poziomu QoS 1 zapewniają, że zduplikowane wiadomości nie są dostarczane do aplikacji odbierającej. Jest to najwyższy poziom dostarczania, stosowany gdy zduplikowane wiadomości są niedopuszczalne. Wiąże się to ze wzrostem ruchu sieciowego, który jest jednak zazwyczaj akceptowalny ze względu na wagę treści wiadomości.

Wiadomość z poziomem QoS 2 posiada identyfikator Message ID w treści wiadomości.

Poniższa tabela przedstawia przepływ protokołu MQTT QoS poziomu 2. Dla sposobu obsługi przepływu PUBLISH przez odbiorcę dostępne są dwie semantyki.

| Klient                    | Wiadomość i kierunek | Serwer                             |
|---------------------------|----------------------|------------------------------------|
| QoS = 2<br>Message ID = x | PUBLISH<br>----->    | <b>Akcja:</b> Przechowaj wiadomość |

|                                     |                         |                                                                                                                               |
|-------------------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>Action:</b> Przechowaj wiadomość |                         |                                                                                                                               |
|                                     | PUBREC<br><-----        | Identyfikator wiadomości = x                                                                                                  |
| Identyfikator wiadomości = x        | PUBREL<br>----->        | <b>Akcje:</b> <ul style="list-style-type: none"> <li>• Opublikuj wiadomość subskrybentom</li> <li>• Usuń wiadomość</li> </ul> |
| <b>Akcja:</b> Odrzuć wiadomość      | POTWIERDZENIE<br><----- | Identyfikator wiadomości = x                                                                                                  |

W przypadku wykrycia błędu lub po upływie zdefiniowanego okresu, przepływ protokołu jest ponawiany od ostatniej niepotwierdzonej wiadomości protokołu. Dodatkowe przepływy protokołu zapewniają, że wiadomość jest dostarczana do subskrybentów dokładnie raz.

# Kolejki

Obsługiwane przez

TsgcWSPServer\_sgc

TsgcWSPClient\_sgc

JavaScript

Domyślny protokół SGC implementuje kolejki w celu dodania trwałości do publikowanych wiadomości (dostępne tylko dla wiadomości opublikowanych)

Poziom 0: Wiadomości nie są kolejgowane na serwerze

Poziom 1: na serwerze kolejgowana jest tylko ostatnia wiadomość i jest ona wysyłana za każdym razem, gdy klient subskrybuje nowy kanał lub łączy się z serwerem.

Poziom 2: wszystkie wiadomości są kolejgowane na serwerze i są wysyłane za każdym razem, gdy klient subskrybuje nowy kanał lub nawiązuje połączenie z serwerem.

## Poziom 0

Wiadomość nie jest kolejgowana przez serwer

Poniższa tabela przedstawia przepływ protokołu na poziomie 0 kolejki.

| Klient    | Wiadomość i kierunek | Serwer                                                 |
|-----------|----------------------|--------------------------------------------------------|
| Queue = 0 | PUBLISH<br>----->    | <b>Akcja:</b> Publikowanie wiadomości do subskrybentów |

## Poziom 1

Wiadomość z poziomem kolejki 1 jest przechowywana na serwerze, a jeśli dla tego kanału istnieją inne przechowywane wiadomości, są one usuwane.

Poniższa tabela przedstawia przepływ protokołu Queue na poziomie 1.

| Klient                            | Wiadomość i kierunek | Serwer                                                                                                                                                        |
|-----------------------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Queue = 1                         | PUBLISH<br>----->    | <b>Akcje:</b> <ul style="list-style-type: none"><li>• Usuwa wszystkie wiadomości tego kanału</li><li>• Przechowuje ostatnią wiadomość według kanału</li></ul> |
| <b>Akcja:</b> Przetwórz wiadomość | NOTIFY<br><-----     | <b>Działanie:</b> Za każdym razem, gdy nowy klient subskrybuje ten kanał, wysyłana jest ostatnia wiadomość.                                                   |

Jest to przydatne w sytuacjach, gdy wydawcy wysyłają wiadomości na zasadzie "raportowania wyjątków", gdy między wiadomościami może upłynąć pewien czas. Dzięki temu nowi subskrybenci mogą natychmiast otrzymywać dane z zachowaną wartością lub ostatnią znaną prawidłową wartością.

Poziom 2

Wszystkie wiadomości z poziomem kolejki 2 są przechowywane na serwerze.

Poniższa tabela przedstawia przepływ protokołu kolejki na poziomie 2.

| Klient                            | Wiadomość i kierunek | Serwer                                                                                                  |
|-----------------------------------|----------------------|---------------------------------------------------------------------------------------------------------|
| Queue = 2                         | PUBLISH<br>----->    | <b>Akcja:</b> Przechowaj wiadomość                                                                      |
| <b>Akcja:</b> Przetwórz wiadomość | NOTIFY<br><-----     | <b>Akcja:</b> Za każdym razem, gdy nowy klient subskrybuje ten kanał, wysyłane są WSZYSTKIE wiadomości. |

# Transakcje

## Obsługiwane przez

TsgcWSPServer\_sgc

TsgcWSPClient\_sgc

JavaScript

Protokół SGC w sgcWebSockets obsługuje transakcyjne przesyłanie wiadomości. Gdy klient zatwierdza transakcję, wszystkie wiadomości wysłane przez klienta są przetwarzane po stronie serwera. Klient wywołuje 3 metody:

### StartTransaction

Tworzy nową transakcję po stronie serwera; wszystkie wiadomości wysyłane od klienta do serwera po wywołaniu tej metody są kolejgowane po stronie serwera do momentu wywołania przez klienta Commit lub Rollback.

| Klient                                                            | Wiadomość i kierunek       | Serwer                                                                                        |
|-------------------------------------------------------------------|----------------------------|-----------------------------------------------------------------------------------------------|
| Kanał = X                                                         | STARTTRANSACTION<br>-----> | <b>Action:</b> Tworzy nową kolejkę do przechowywania wszystkich wiadomości określonego kanału |
| Kanał = X                                                         | PUBLISH<br>----->          | <b>Akcja:</b> Wiadomość jest przechowywana po stronie serwera.                                |
| <b>Action:</b> Klient otrzymuje potwierdzenie wysłanej wiadomości | ACKNOWLEDGEMENT<br><-----  | <b>Action:</b> Serwer zwraca potwierdzenie do klienta, ponieważ wiadomość jest przechowywana. |
| ....                                                              | ....                       | ....                                                                                          |

### Commit

Gdy klient wywoła Commit, wszystkie wiadomości umieszczone w kolejce przez serwer są przetwarzane.

| Klient | Wiadomość i kierunek | Serwer |
|--------|----------------------|--------|
|--------|----------------------|--------|

|              |                  |                                                                            |
|--------------|------------------|----------------------------------------------------------------------------|
| Kanał =<br>X | COMMIT<br>-----> | <b>Akcja:</b> Przetwórz wszystkie wiadomości kolejkowane przez Transaction |
|--------------|------------------|----------------------------------------------------------------------------|

## RollBack

Gdy klient wywołuje RollBack, wszystkie wiadomości kolejkowane przez serwer są usuwane i nie są przetwarzane po stronie serwera.

| Klient       | Wiadomość i kierunek | Serwer                                                                         |
|--------------|----------------------|--------------------------------------------------------------------------------|
| Kanał =<br>X | ROLLBACK<br>----->   | <b>Akcja:</b> Usuń wszystkie wiadomości umieszczone w kolejce przez transakcję |

# TCP Connections

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

Domyślnie sgcWebSocket używa WebSocket jako protokołu, ale w komponentach klienta i serwera można używać zwykłego protokołu TCP.

## Klient Komponent

Wyłącz protokół WebSocket.

```
Client->Specifications->RFC6455 = false;
```

## Komponent serwerowy

Obsłuż zdarzenie OnUnknownProtocol, ustaw Transport na trpTCP i zaakceptuj połączenie.

```
void OnUnknownProtocol(TsgcWSConnection *Connection, ref bool Accept)
{
    Connection->Transport = trpTCP;
    Accept = true;
}
```

Gdy klient łączy się z serwerem, połączenie to zostanie zdefiniowane jako TCP i będzie używać protokołu TCP zamiast WebSockets. Połączenia TCP nie rozróżniają wiadomości tekstowych i binarnych, więc wszystkie odebrane wiadomości są obsługiwane przez zdarzenie OnBinary.

## Koniec wiadomości

Jeśli wiadomości są duże, mogą być czasami odbierane w fragmentach. Istnieje metoda wykrywania końca wiadomości poprzez określenie bajtów do wyszukania. Przykład: w protokole STOMP wszystkie wiadomości kończą się bajtami 0 i 10.

```
void OnWSClientConnect(TsgcWSConnection *Connection)
{
    Connection->TCPEndOfFrameScanBuffer = eofScanAllBytes;
    Connection->AddTCPEndOfFrame(0);
}
```



```
Connection->AddTCPEndOfFrame(10);  
}
```

# SubProtocol

## Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

WebSocket zapewnia prostą negocjację podprotokołu. W nagłówku żądania dodawane są nazwy protokołów obsługiwanych przez klienta. Po ich odebraniu, jeśli odbiorca obsługuje jeden z nich, wysyła odpowiedź z obsługiwany podprotokołem.

sgcWebSockets obsługuje kilka protokołów podrzędnych: [MQTT](#), [WAMP](#)... i inne. Można implementować własne protokoły podrzędne za pomocą prostej metody — wystarczy wywołać `RegisterProtocol` i przekazać nazwę protokołu jako argument.

**Przykład:** należy połączyć się z serwerem implementującym podprotokół "Test 1.0"

```
Client = new TsgcWebSocketClient();
Client->Host = "server host";
Client->Port = server.port;
Client->RegisterProtocol("Test 1.0");
Client->Active = true;
```

Aby **używać więcej niż jednego protokołu w jednym połączeniu**, można użyć komponentów **Broker Protocol** (serwer i klient). Wystarczy umieścić Broker między klientem/serwerem a protokołami. **Przykład:** używanie protokołów SGC i Files w jednym połączeniu.

```
// ... server
TsgcWebSocketServer *oServer = new TsgcWebSocketServer();
TsgcWSPServer_Broker *oServerBroker = new TsgcWSPServer_Broker();
oServerBroker->Server = oServer;
TsgcWSPServer_sgc *oServerSGC = new TsgcWSPServer_sgc();
oServerSGC->Broker = oServerBroker;
TsgcWSPServer_files *oServerFiles = new TsgcWSPServer_files();
oServerFiles->Broker = oServerBroker;
// ... client
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSPClient_Broker *oClientBroker = new TsgcWSPClient_Broker();
oClientBroker->Client = oClient;
TsgcWSPClient_sgc *oClientSGC = new TsgcWSPClient_sgc();
oClientSGC->Broker = oClientBroker;
TsgcWSPClient_files *oClientFiles = new TsgcWSPClient_files();
```

```
oClientFiles->Broker = oClientBroker;
```

Gdy broker protokołów jest dołączony między serwerem/klientem a protokołem, zdarzenia **OnConnect** i **OnDisconnect** są wywoływane w składniku Broker (zamiast w składnikach serwera lub klienta).

# Throttle

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

Ograniczanie przepustowości jest obsługiwane przez komponenty serwera i klienta; jeśli jest włączone, może ograniczyć liczbę bitów na sekundę wysyłanych/odbieranych przez gniazdo. Indy używa metody blokującej, więc jeśli klient ogranicza odczyt, nieprzeczytane dane pozostaną w gnieździe klienta, a serwer zostanie zablokowany przed zapisem nowych danych do klienta. Im wolniej klient odczytuje dane, tym wolniej serwer może zapisywać nowe dane.

# Server-sent Events (Powiadomienia push)

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

JavaScript

SSE nie są częścią WebSockets; definiują API do otwierania połączenia HTTP w celu odbierania powiadomień push z serwera.

SSE są wysyłane przez tradycyjne HTTP. Oznacza to, że nie wymagają specjalnego protokołu ani implementacji serwera. Ponadto Server-Sent Events oferują szereg funkcji, których brakuje WebSockets z założenia, takich jak automatyczne ponowne łączenie, identyfikatory zdarzeń oraz możliwość wysyłania dowolnych zdarzeń.

## Zdarzenia

- **Open:** gdy nowe połączenie SSE zostaje otwarte.
- **Message:** gdy klient odbierze nową wiadomość.
- **Error:** gdy wystąpi błąd połączenia, taki jak rozłączenie.

## JavaScript API

Aby subskrybować strumień zdarzeń, należy utworzyć obiekt EventSource i przekazać mu adres URL strumienia:

```
var sse = new EventSource('sse.html');

sse.addEventListener('message', function(e)
{console.log(e.data);
}, false);

sse.addEventListener('open', function(e) {
```

```
// Connection was opened.
}, false);

sse.addEventListener('error', function(e) {
  if (e.readyState == EventSource.CLOSED) {
    // Connection was closed.
  }
}, false);
```

Gdy aktualizacje są wypychane z serwera, procedura obsługi `onmessage` jest wywoływana i nowe dane są dostępne we właściwości `e.data`. Jeśli połączenie zostanie zamknięte, przeglądarka automatycznie ponownie połączy się ze źródłem po około 3 sekundach (jest to domyślny interwał ponownych prób, który można zmienić po stronie serwera).

## Pola

Poniżej zdefiniowano następujące nazwy pól zgodnie ze specyfikacją:

### zdarzenie

Typ zdarzenia. Jeśli jest określony, w przeglądarce zostanie wywołane zdarzenie dla słuchacza o podanej nazwie; witryna internetowa użyłaby `addEventListener()` do nasłuchiwanie nazwanych zdarzeń. Procedura obsługi `onmessage` jest wywoływana, jeśli dla wiadomości nie podano nazwy zdarzenia.

### dane

Pole danych dla wiadomości. Gdy `EventSource` odbiera wiele kolejnych wierszy zaczynających się od `data:`, łączy je, wstawiając między nimi znak nowego wiersza. Końcowe znaki nowego wiersza są usuwane.

### id

Identyfikator zdarzenia używany do ustawienia wartości ostatniego identyfikatora zdarzenia obiektu EventSource.

retry

Czas ponownego połączenia używany podczas próby wysłania zdarzenia. Musi być liczbą całkowitą określającą czas ponownego połączenia w milisekundach. Jeśli podana zostanie wartość niebędąca liczbą całkowitą, pole zostanie zignorowane.

Wszystkie pozostałe nazwy pól są ignorowane.

W przypadku ciągów wieloliniowych należy używać #10 jako znaku nowej linii.

### **Przykłady użycia:**

Aby wysłać wiadomość do klienta, należy użyć metody WriteData.

```
// If you need to send a message to a client, just use WriteData method.
Connection->WriteData("Notification from server");

// To send a message to all Clients, use Broadcast method.
Connection->Broadcast("Notification from server");

// To send a message to all Clients using url 'sse.html', use Broadcast method and Channel parameter:
Connection->Broadcast("Notification from server", "/sse.html");

// You can send a unique id with an stream event by including a line starting with "id:":
Connection->WriteData("id: 1 \r data: Notification from server");

// If you need to specify an event name:
Connection->WriteData("event: notifications \r data: Notification from server");
```

javascript kod do nasłuchiwanie kanału "notifications":

```
sse.addEventListener('notifications', function(e) {  
  console.log('notifications:' + e.data);  
}, false);
```



# LoadBalancing

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketLoadBalancerServer](#)

Równoważenie obciążenia umożliwia dystrybucję pracy między kilkoma serwerami backendowymi. Za każdym razem, gdy nowy klient żąda połączenia, łączy się z serwerem równoważenia obciążenia (który jest połączony z serwerami backendowymi) i otrzymuje ciąg połączenia z informacjami o hoście, porcie itp., który jest używany przez klienta do połączenia z serwerem. Przy 4 serwerach wszystkie serwery będą miały mniej więcej tę samą liczbę połączeń, a obciążenie będzie podobne.

Jeśli klient chce wysłać wiadomość do wszystkich klientów wszystkich serwerów, wystarczy użyć metody Broadcast, a wiadomość zostanie rozesłana do wszystkich serwerów połączonych z serwerem równoważenia obciążenia.

Aby włączyć tę funkcję:

1. Upuść komponent [TsgcWebSocketLoadBalancerServer](#), ustaw port nasłuchiwania i ustaw właściwość active na True.
2. Komponenty serwera i klienta posiadają właściwość LoadBalancer, w której należy ustawić host i port serwera Load Balancer oraz ustawić wartość Enabled na True.

Komponent umożliwia równoważenie obciążenia protokołów **WebSocket** i **HTTP**.

# Pliki

## Obsługiwane przez

[TsgcWSPServer\\_sgc](#)

[TsgcWSPClient\\_sgc](#)

Ten protokół umożliwia przesyłanie plików od klienta do serwera i od serwera do klienta w prosty sposób. Obsługuje pliki zarówno bardzo małe, jak i duże, przy niskim zużyciu pamięci. Dostępne ustawienia:

1. Rozmiar pakietu w bajtach.
2. Należy użyć niestandardowych kanałów, aby wysyłać pliki wyłącznie do subskrybujących klientów.
3. Postęp wysyłania i odbierania pliku.
4. Autoryzacja odebranych plików.
5. Potwierdzenie odebranych pakietów.

# Proxy

Obsługiwane przez

[TsgcWebSocketClient](#)

Klienckie komponenty WebSocket obsługują połączenia WebSocket przez proxy HTTP. Aby włączyć połączenie przez proxy, należy aktywować następujące właściwości:

## Proxy / Włączony

Po ustawieniu na True można skonfigurować:

**Host:** Adres serwera proxy

**Port:** port serwera proxy

**UserName/Password:** Uwierzytelnianie do połączenia z serwerem proxy, wyłącznie jeśli jest wymagane.

**ProxyType:** obsługiwane są następujące typy proxy:

- HTTP
- Socks4
- Socks4A
- Socks5

Można skonfigurować proxy SOCKS, uzyskując dostęp do właściwości SOCKS i ustawiając Enabled na True.

# Fragmentowane wiadomości

Obsługiwane przez

[TsgcWebSocketServer](#)

[TsgcWebSocketHTTPServer](#)

[TsgcWebSocketClient](#)

[TsgcWebSocketServer\\_HTTPAPI](#)

Domyślnie, gdy strumień jest wysyłany przy użyciu biblioteki `sgcWebSockets`, wszystkie dane są wysyłane w jednym pakiecie lub buforowane, a po odebraniu ostatniego pakietu wywoływane jest zdarzenie `OnBinary message`.

To zachowanie można dostosować za pomocą właściwości **Options.FragmentedMessages**, która przyjmuje następujące wartości:

1. `frgOnlyBuffer`: jest to wartość domyślna. Oznacza, że wiadomości pakietowe będą buforowane i dopiero po odebraniu całego strumienia zostanie wywołane zdarzenie wiadomości `OnBinary`.
2. `frgOnlyFragmented`: oznacza, że zdarzenie `OnFragmented` będzie wywoływane wyłącznie dla każdego odebranego pakietu.
3. `frgAll`: oznacza to, że zdarzenie `OnFragmented` będzie wywoływane dla każdego odebranego pakietu oraz po odebraniu pełnego strumienia.

Zdarzenie **OnFragmented** jest przydatne, gdy konieczne jest wysyłanie dużych strumieni, a odbiorca musi wyświetlać postęp transferu.

**Przykład:** klient musi wysłać strumień o rozmiarze 1 000 000 bajtów do serwera, a serwer chce wyświetlać postęp dla każdych odebranych 1000 bajtów.

Klient wyśle strumień metodą `writedata` z rozmiarem pakietu równym 1000

```
Client->WriteData(stream, 1000);
```

Serwer ustawi `Options.FragmentedMessages := frgAll` i będzie obsługiwał zdarzenie `OnFragmented`, aby odbierać postęp strumieni

```
void OnFragmented(TsgcWSConnection *Connection, const TMemoryStream *Data, const TOpCode OpCode, const boolean Continuation)
{
    ShowProgress(Data->Size);
    if (Continuation == false)
    {
        SaveStream(Data);
    }
}
```

# TsgcWebSocketClient

Komponent klienta WebSocket — łączy się z dowolnym serwerem WebSocket zgodnym z RFC 6455 i wysyła/odbiera wiadomości tekstowe i binarne.

## Wprowadzenie

TsgcWebSocketClient implementuje komponent klienta WebSocket i umożliwia połączenie z serwerem WebSocket. Poniższe kroki opisują sposób konfiguracji tego komponentu:

1. Upuść komponent **TsgcWebSocketClient** na formularz.
2. Należy ustawić **Host** i **Port** (domyślnie 80), aby nawiązać połączenie z dostępnym serwerem WebSocket. Można ustawić właściwość **URL**, a Host, Port, Parameters... zostaną zaktualizowane na podstawie adresu URL. **Przykład:** wss://127.0.0.1:8080/ws/ spowoduje:

```
oClient = new TsgcWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClient->TLS = true;
oClient->Options->Parameters = "/ws/";
```

3. Można wybrać, czy wymagane jest **TLS** (bezpieczne połączenie), czy nie — domyślnie opcja ta jest nieaktywna.

4. Można połączyć się przez serwer proxy HTTP; należy zdefiniować właściwości proxy:

**Host:** nazwa hosta serwera proxy.

**Port:** numer portu serwera proxy.

**Username:** nazwa użytkownika do uwierzytelnienia; pozostaw puste w przypadku dostępu anonimowego.

**Password:** hasło do uwierzytelnienia; pozostaw puste w przypadku dostępu anonimowego.

5. Jeśli serwer obsługuje **kompresję**, można włączyć kompresję przesyłanych wiadomości.

6. Ustaw dozwolone **specyfikacje**; domyślnie wszystkie specyfikacje są włączone.

**RFC6455:** jest standardową i zalecaną specyfikacją WebSocket.

**Hixie76:** zawsze false

7. Ustaw `Active = true`, aby nawiązać nowe połączenie WebSocket.

## Dokumentacja

- Właściwości — 23 właściwości dostępne w komponencie.
- Metody — 22 metody publiczne.
- Zdarzenia — 16 zdarzeń do obsłużenia.

# Przewodniki

## Najczęstsze zastosowania

- **Połączenie**
  - [Jak połączyć się z serwerem WebSocket](#)
  - [Otwórz połączenie klienta](#)
  - [Zamknij połączenie klienta](#)
  - [Utrzymuj połączenie aktywne](#)
  - [Nieoczekiwane rozłączenia](#)
  - [Połącz serwer TCP](#)
  - [Przekierowania WebSocket](#)
- **Bezpieczne serwery**
  - [Połącz z bezpiecznym serwerem](#)
  - [Certyfikaty OpenSSL](#)
  - [Certyfikaty SChannel](#)
  - [SChannel: informacje o połączeniu](#)
- **Wysyłanie wiadomości**
  - [Wyślij wiadomość tekstową](#)
  - [Wyślij wiadomość binarną](#)
- **Odbieranie wiadomości**
  - [Odbieranie wiadomości tekstowych](#)
  - [Odbieranie wiadomości binarnych](#)
- **Uwierzytelnianie**
  - [Uwierzytelnianie klienta](#)
- **Inne**
  - [Wyjątki klienta](#)
  - [Uzgadnianie WebSocket klienta](#)
  - [Rejestracja protokołu klienta](#)
  - [Serwery proxy klienta](#)

# TsgcWebSocketClient | Połączenie z serwerem WebSocket

## Właściwość URL

Najprostszym sposobem połączenia z serwerem WebSocket jest użycie właściwości **URL** i ustawienie **Active = true**.

**Przykład:** połącz z [www.esegece.com](http://www.esegece.com) przy użyciu bezpiecznego połączenia.

```
oClient = new TsgcWebSocketClient();  
oClient->URL = "wss://www.esegece.com:2053";  
oClient->Active = true;
```

## Host, Port i parametry

Połączenie z serwerem WebSocket można nawiązać, używając właściwości Host i port.

**Przykład:** połącz się z [www.esegece.com](http://www.esegece.com) przy użyciu bezpiecznych połączeń

```
oClient = new TsgcWebSocketClient();  
oClient->Host = "www.esegece.com";  
oClient->Port = 2053;  
oClient->TLS = true;  
oClient->Active = true;
```



# TsgcWebSocketClient | Client Open Connection

Po skonfigurowaniu klienta do połączenia z serwerem dostępne są 3 różne opcje otwarcia nowego połączenia.

## Właściwość Active

Najprostszym sposobem otwarcia nowego połączenia jest ustawienie właściwości Active na True. Spowoduje to próbę połączenia z serwerem przy użyciu konfiguracji komponentu.

Ustawienie właściwości Active na false spowoduje zamknięcie połączenia, jeśli jest aktywne.

Metoda ta jest wykonywana w tym samym wątku co wywołujący. Jeśli zostanie wywołana z wątku głównego, metoda zostanie wykonana w wątku głównym aplikacji.

## Otwórz połączenie

```
oClient = new TsgcWebSocketClient();  
...  
oClient->Active = true;
```

Po ustawieniu Active = true **nadal nie można wysłać danych do serwera**, ponieważ klient może jeszcze nawiązywać połączenie. Należy najpierw poczekać na zdarzenie OnConnect, a dopiero potem można rozpocząć wysyłanie wiadomości do serwera.

## Zamknij połączenie

```
oClient->Active = false;
```

Po ustawieniu Active = false **nie można mieć pewności, że połączenie jest już zamknięte** bezpośrednio po tym kodzie, dlatego należy poczekać, aż zostanie wywołane zdarzenie OnDisconnect.

## Metody Start/Stop

Wywołanie metody Start() lub Stop() w celu nawiązania lub zakończenia połączenia z serwerem jest wykonywane w wątku pomocniczym, dlatego nie blokuje wątku, z którego zostało wywołane. Należy użyć tej metody, gdy wymagane jest nawiązanie połączenia z serwerem bez wstrzymywania wykonywania kodu poniżej.

## Otwórz połączenie

```
oClient = new TsgcWebSocketClient();
...
oClient->Start();
```

Po wywołaniu Start(), **nadal nie można wysyłać danych do serwera**, ponieważ klient może być jeszcze w trakcie nawiązywania połączenia. Należy najpierw poczekać na wywołanie zdarzenia OnConnect, a dopiero potem można rozpocząć wysyłanie wiadomości do serwera.

## Zamknij połączenie

```
oClient->Stop();
```

Po wywołaniu Stop(), **nie można mieć pewności, że połączenie jest już zamknięte** bezpośrednio po wykonaniu tego kodu, dlatego należy poczekać na wyzwolenie zdarzenia OnDisconnect.

## Metody Connect/Disconnect

Wywołanie metody Connect() lub Disconnect() w celu otwarcia lub zamknięcia połączenia z serwerem jest wykonywane w tym samym wątku, w którym zostało wywołane, ale oczekuje na zakończenie operacji. Należy ustawić wartość Timeout, która określa maksymalny czas oczekiwania na zakończenie operacji (domyślnie 10 sekund).

**Przykład:** połączenie z serwerem i oczekiwanie maksymalnie 5 sekund

```
oClient = new TsgcWebSocketClient();
...
if (oClient->Connect(5000) == true)
{
    oClient->WriteData("Hello from client");
}
else
{
    Error();
}
```

Jeśli metoda Connect() zwróci pomyślny wynik, można już wysłać wiadomość do serwera, ponieważ połączenie jest aktywne.

**Przykład:** rozłącz z serwerem i czekaj do 10 sekund

```
if (oClient->Disconnect(10000) == true)
{
    ShowMessage("Disconnected");
}
else
```

```
{  
    ShowMessage("Not Disconnected");  
}
```

Jeżeli metoda Disconnect() zwróci wynik powodzenia, oznacza to, że połączenie jest już zamknięte.

Zdarzenie **OnBeforeConnect** może być użyte do dostosowania właściwości połączenia serwera, zanim klient spróbuje się z nim połączyć.

## TsgcWebSocketClient | Zamykanie połączenia klienta

Połączenie można zamknąć za pomocą właściwości `Active` lub metod `Stop` i `Disconnect`. Więcej informacji można znaleźć w sekcji [Otwieranie połączenia klienta](#).

### CleanDisconnect

Gdy połączenie jest zamykane, można powiadomić drugiego peera o zamknięciu połączenia przez wysłanie wiadomości zamknięcia. Aby włączyć tę funkcję, ustaw właściwość `Options.CleanDisconnect` na `true`.

Jeśli ta właściwość jest włączona, przed zamknięciem połączenia zostanie wysłana do serwera wiadomość `Close` informująca o zamknięciu połączenia przez klienta.

### Rozłącz

[TsgcWSConnection](#) posiada metodę `Disconnect()`, która umożliwia rozłączenie połączenia na poziomie gniazda sieciowego. Po wywołaniu tej metody gniazdo zostanie rozłączone bezpośrednio, bez oczekiwania na odpowiedź serwera. Wraz z metodą można przesłać kod zamknięcia.

### Close

[TsgcWSConnection](#) posiada metodę `Close()`, która pozwala wysłać do serwera wiadomość z żądaniem zamknięcia połączenia. Jeśli serwer odbierze tę wiadomość, musi zamknąć połączenie, a klient otrzyma powiadomienie o zamknięciu połączenia. Wraz z tą metodą można przesłać kod zamknięcia.

# TsgcWebSocketClient | Utrzymywanie połączenia otwartego

Po nawiązaniu połączenia klienta z serwerem połączenie może niekiedy zostać zerwane z powodu słabego sygnału, błędów połączenia itp. Dostępne są 2 właściwości pomagające utrzymać aktywne połączenie.

## HeartBeat

**HeartBeat** to właściwość umożliwiająca **wysyłanie sygnału Ping** co **X sekund**, aby **utrzymać połączenie aktywne**. Niektóre serwery zamykają połączenia TCP, gdy między partnerami nie są wymieniane żadne dane. HeartBeat rozwiązuje ten problem, wysyłając ping w określonym interwale. Zazwyczaj jest to wystarczające do utrzymania aktywnego połączenia, ale można ustawić interwał Timeout, jeśli połączenie ma być zamykane, gdy odpowiedź serwera nie zostanie odebrana po X sekundach.

**Przykład:** wysyłanie pinga co 30 sekund

```
oClient = new TsgcWebSocketClient();
oClient->HeartBeat->Interval = 30;
oClient->HeartBeat->Timeout = 0;
oClient->HeartBeat->Enabled = true;
oClient->Active = true;
```

Istnieje zdarzenie o nazwie **OnBeforeHeartBeat**, które umożliwia dostosowanie zachowania HeartBeat. Domyślnie, jeśli HeartBeat jest włączony, klient wysyła ping WebSocket co X sekund zgodnie z wartością właściwości HeartBeat.Interval.

**OnBeforeHeartBeat** posiada parametr o nazwie Handled, domyślnie ustawiony na false, co oznacza, że przebieg jest kontrolowany przez komponent TsgcWebSocketClient. Ustawienie tej wartości na True powoduje, że ping nie zostanie wysłany, a zamiast tego można wysłać własną wiadomość przy użyciu klasy Connection.

## WatchDog

Jeśli WatchDog jest włączony, gdy klient wykryje rozłączenie, WatchDog próbuje ponownie nawiązać połączenie co X sekund, dopóki połączenie nie będzie ponownie aktywne.

**Przykład:** ponowne połączenie co 10 sekund po rozłączeniu z nieograniczoną liczbą prób.

```
oClient = new TsgcWebSocketClient();
oClient->WatchDog->Interval = 10;
oClient->WatchDog->Attempts = 0;
oClient->WatchDog->Enabled = true;
oClient->Active = true;
```

Zdarzenia **OnBeforeWatchDog** można użyć do zmiany serwera, z którym klient spróbuje nawiązać połączenie. **Przykład:** po 3 próbach, jeśli klient nie może połączyć się z serwerem, podejmie próbę połączenia z serwerem zapasowym.

Właściwość **Handled**, jeśli ustawiona na True, oznacza, że klient nie będzie próbował ponownie nawiązać połączenia.

# TsgcWebSocketClient | Dropped Disconnections

Po nawiązaniu połączenia, jeśli żaden peer nie wysyła żadnych danych, przez sieć nie są przesyłane żadne pakiety. TCP jest protokołem beczynnym, dlatego zakłada, że połączenie jest nadal aktywne.

## Przyczyny rozłączenia

- **Zamknięcie aplikacji:** gdy proces kończy działanie, zazwyczaj wysyła pakiet FIN, który informuje drugi węzeł o zamknięciu połączenia. Jeśli jednak proces ulegnie awarii, nie ma gwarancji, że ten pakiet zostanie wysłany do drugiego węzła.
- **Zamknięcie urządzenia:** w przypadku zamknięcia urządzenia najprawdopodobniej nie zostanie wysłane żadne powiadomienie na ten temat.
- **Odłączony kabel sieciowy:** odłączenie kabla sieciowego ma taki sam skutek jak zamknięcie routera; nie są przesyłane żadne dane, więc połączenie nie jest zamykane.
- **Utrata sygnału z routera:** jeśli aplikacja utraci sygnał z routera, połączenie pozostanie aktywne.

## Wykrywanie połączeń w stanie półotwartym

Rozłączenia można próbować wykryć przy użyciu następujących metod

### Drugie połączenie

Można spróbować otworzyć drugie połączenie i podjąć próbę nawiązania go, jednak rozwiązanie to ma pewne wady: zużywa więcej zasobów, tworzy nowe wątki itp. Ponadto, jeśli drugi peer został zrestartowany, drugie połączenie będzie działać, ale pierwsze nie.

### Wyślij ping do drugiego peera

Przy próbie wysłania ping lub jakiegokolwiek wiadomości przez połączenie w stanie half-open nie zostanie zwrócony żaden błąd.

## Włącz KeepAlive na poziomie gniazda TCP

Pakiet TCP keep-alive to po prostu ACK z numerem sekwencji ustawionym o jeden mniej niż bieżący numer sekwencji połączenia. Host odbierający jedno z tych ACK odpowiada ACK dla bieżącego numeru sekwencji. Pakiety keep-alive mogą być używane do weryfikacji, czy komputer na zdalnym końcu połączenia jest nadal dostępny. Pakiety TCP keep-alive mogą być wysyłane raz na `TCPKeepAlive.Time` (domyślnie 7 200 000 milisekund, czyli dwie godziny), jeśli przez połączenie TCP nie były przesyłane inne dane ani pakiety keep-alive wyższego poziomu. Jeśli nie ma odpowiedzi na keep-alive, jest on powtarzany co `TCPKeepAlive.Interval` sekund. `KeepAliveInterval` domyślnie wynosi 1000 milisekund.

Można włączyć funkcję KeepAlive dla poszczególnych połączeń i umożliwić protokołowi TCP sprawdzenie, czy połączenie jest aktywne. Jest to preferowana metoda, gdy wymagane jest wykrywanie zerwanych połączeń (na przykład: przy odłączeniu kabla sieciowego).

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);  
oClient->TCPKeepAlive->Enabled = true;  
oClient->TCPKeepAlive->Time = 5000;  
oClient->TCPKeepAlive->Interval = 1000;
```



# TsgcWebSocketClient | Połącz z serwerem TCP

TsgcWebSocketClient może łączyć się z serwerami WebSocket, ale może też łączyć się z serwerami zwykłego TCP.

## Właściwość URL

Najprostszym sposobem nawiązania połączenia z serwerem TCP jest użycie właściwości **URL** i ustawienie wartości **Active = true**.

**Przykład:** połączenie z 127.0.0.1 port 5555

```
oClient = new TsgcWebSocketClient();  
oClient->URL = "tcp://127.0.0.1:5555";  
oClient->Active = true;
```

## Host, Port i parametry

Połączenie z serwerem TCP jest możliwe przy użyciu właściwości Host i port.

**Przykład:** połączenie z 127.0.0.1 port 5555

```
oClient = new TsgcWebSocketClient();  
oClient->Specifications->RFC6455 = false;  
oClient->Host = "127.0.0.1";  
oClient->Port = 5555;  
oClient->Active = true;
```

## TsgcWebSocketClient | Połączenia TIME\_WAIT

Gdy klient inicjuje rozłączenie z serwerem, między klientem a serwerem odbywa się wymiana komunikatów dotycząca stanu rozłączenia. Po zakończeniu tego procesu gniazdo klienta przechodzi w stan TIME\_WAIT na czas o zmiennej długości. Jest to normalne zachowanie; w systemach operacyjnych Windows czas ten wynosi domyślnie około 4 minut.

Można ograniczyć lub wyeliminować to zachowanie, choć należy robić to ostrożnie, korzystając z następujących alternatyw.

### REGEDIT

Wartość TIME\_WAIT można zmniejszyć przy użyciu Edytora rejestru systemu Windows (Regedit).

1. Otwórz Regedit i przejdź do podkluczy rejestru  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters.
2. Utwórz nową wartość REG\_DWORD o nazwie **TcpTimedWaitDelay**
3. Należy ustawić wartość w sekundach. Przykład: wartość 5 oznacza, że TIME\_WAIT będzie czekać maksymalnie 5 sekund.
4. Zapisz i uruchom ponownie system.

### LINGER

Inną opcją pozwalającą uniknąć stanu TIME\_WAIT jest użycie opcji gniazda SO\_LINGER. Jeśli jest włączona, zamiast łagodnego zamknięcia połączenia klient resetuje połączenie, dzięki czemu stan TIME\_WAIT jest pomijany.

Tę opcję można włączyć za pomocą właściwości **LingerState**, która domyślnie ma wartość -1. Ustawienie wartości zero spowoduje zresetowanie połączenia podczas rozłączania z gniazdem bez limitu czasu.

Ta opcja jest prawdopodobnie najmniej zalecana i powinna być stosowana wyłącznie jako ostateczność.

## TsgcWebSocketClient | Przekierowania WebSocket

Gdy klient łączy się z serwerem WebSocket, serwer może zwrócić kod odpowiedzi HTTP 30x. Kod odpowiedzi 301 oznacza, że zasób został trwale przeniesiony, a nowy adres URL jest podany w nagłówku HTTP Location.

Klient WebSocket obsługuje przekierowania automatycznie: po wykryciu odpowiedzi serwera zawierającej przekierowanie rozłącza bieżące połączenie i próbuje połączyć się z nowym adresem URL Location.

### Przykład

1. Klient najpierw próbuje nawiązać połączenie z adresem ws://127.0.0.1:5000
2. Serwer zwraca kod odpowiedzi 301 i zawiera nagłówek Location z wartością ws://80.50.1.2:3000
3. Klient odczytuje odpowiedź z serwera, wykrywa, że jest to przekierowanie, i odczytuje lokalizację (Location)
  1. Najpierw rozłącza bieżące połączenie.
  2. Aktualizuje właściwość URL wartością nagłówka Location (ws://80.50.1.2:3000)
  3. Nawiązuje połączenie z nowym serwerem.

# TsgcWebSocketClient | Połączenie z bezpiecznym serwerem

TsgcWebSocketClient może nawiązywać połączenia z serwerami WebSocket przy użyciu połączeń zabezpieczonych i niezabezpieczonych.

Można skonfigurować bezpieczne połączenie przy użyciu właściwości URL lub właściwości Host / Port. Zobacz [Połączenie z serwerem WebSocket](#).

## TLSOptions

We właściwości **TLSOptions** znajdują się właściwości do **dostosowywania bezpiecznego połączenia**. Najważniejszą właściwością jest **version**, która określa **wersję protokołu TLS**. Zazwyczaj ustawienie **właściwości TLS na true** i **TLSOptions.Version na tlsUndefined** jest wystarczające dla zdecydowanej większości serwerów WebSocket.

Właściwość **TLSOptions.Version** pozwala ustawić wersję TLS używaną do połączenia z serwerem lub pozostawić klientowi możliwość negocjacji wersji TLS spośród wszystkich dostępnych (jest to wartość domyślna, gdy wartość wynosi **tlsUndefined**).

Jeśli podczas próby **połączenia z serwerem** wystąpi błąd dotyczący protokołu TLS, **najprawdopodobniej** serwer **wymaga nowszej wersji TLS** niż skonfigurowana.

Jeśli właściwość **TLSOptions.IOHandler** jest ustawiona na **iohOpenSSL**, konieczne jest **wdrożenie bibliotek OpenSSL** (bibliotek obsługujących wszystkie operacje TLS). Zapoznaj się z artykułem dotyczącym [OpenSSL](#).

Jeśli właściwość **TLSOptions.IOHandler** jest ustawiona na **iohSChannel**, **nie jest wymagane** wdrażanie żadnej biblioteki (obsługiwany jest wyłącznie system Windows).

## TsgcWebSocketClient | Certyfikaty OpenSSL

Gdy serwer wymaga połączenia klienta przy użyciu certyfikatu SSL, należy użyć właściwości `TLSOptions` komponentu `TsgcWebSocketClient` do ustawienia plików certyfikatu. Certyfikat musi być w formacie PEM, więc jeśli certyfikat ma inny format, należy go najpierw przekonwertować do formatu PEM.

Połączenie przez biblioteki OpenSSL wymaga ustawienia `TLSOptions.IOHandler = iohOpenSSL`.

Skonfiguruj następujące właściwości:

- **CertFile:** ścieżka do certyfikatu w formacie PEM.
- **KeyFile:** ścieżka do klucza prywatnego certyfikatu.
- **RootCertFile:** ścieżka do głównego certyfikatu.
- **Password:** jeśli certyfikat jest chroniony hasłem, należy tu podać sekret.

# TsgcWebSocketClient | Certyfikaty SChannel

Jeśli serwer wymaga, aby klient łączył się przy użyciu certyfikatu SSL, należy użyć właściwości `TLSOptions` komponentu `TsgcWebSocketClient` do ustawienia plików certyfikatu.

Połączenie przez `SChannel` wymaga ustawienia `TLSOptions.IOHandler = iohSChannel`.

`SChannel` obsługuje 2 rodzaje uwierzytelniania certyfikatami:

1. Używanie **certyfikatu PFX**
2. Ustawienie **skrót certyfikatu** dla certyfikatu już zainstalowanego w systemie Windows.

## Certyfikat PFX

Certyfikat PFX to plik zawierający certyfikat i klucz prywatny. Jeśli certyfikat jest w formacie PEM, przed użyciem należy go przekonwertować na format PFX.

Użyj poniższego polecenia `openssl`, aby przekonwertować certyfikat PEM na format PFX

```
openssl pkcs12 -inkey certificate-pem.key -in certificate-pem.crt -export -out certificate.pfx
```

Po przekonwertowaniu certyfikatu do formatu PFX wystarczy wdrożyć certyfikat i ustawić właściwość **`TLSOptions.CertFile`** na jego ścieżkę.

```
TLSOptions.IOHandler = iohSChannel  
TLSOptions.CertFile = <certificate path>  
TLSOptions.Password = <certificate optional password>
```

## Certyfikat Hash

Jeśli certyfikat jest już zainstalowany w magazynie certyfikatów systemu Windows, wystarczy znać odcisk palca certyfikatu i ustawić go we właściwości `TLSOptions.SChannel_Options`.

Znalezienie skrót certyfikatu w **PowerShell** jest tak proste, jak uruchomienie polecenia **`dir`** na kontenerze certyfikatów.

```
dir cert:\localmachine\my
```

Skrót jest szesnastkową wartością **Thumbprint**.

```
Directory: Microsoft.PowerShell.Security\Certificate::localmachine\my
Thumbprint                                     Subject
-----
C12A8FC8AE668F866B48F23E753C93D357E9BE10  CN=*.mydomain.com
```

Po uzyskaniu wartości Thumbprint należy ustawić hash oraz lokalizację certyfikatu we właściwości **TLSOptions.SChannel\_Options**.

```
TLSOptions.IOHandler = iohSChannel
TLSOptions.SChannel_Options.CertHash = <certificate thumbprint>
TLSOptions.SChannel_Options.CertStoreName = <certificate store name>
TLSOptions.SChannel_Options.CertStorePath = <certificate store path>
TLSOptions.Password = <certificate optional password>
```

## TsgcWebSocketClient | SChannel Get Connection Info

Po nawiązaniu przez klienta połączenia z bezpiecznym serwerem można uzyskać informacje o używanej wersji TLS (TLS 1.2, TLS 1.3 itp.), zastosowanym szyfrze, sile klucza i innych parametrach.

Wywołaj funkcję **GetInfo** programu obsługi SChannel, aby uzyskać dostęp do tych informacji. Dostęp do programu obsługi SSL można uzyskać za pomocą metody **OnSSLAfterCreateHandler**, która jest wywoływana po utworzeniu programu obsługi SChannel. Po nawiązaniu połączenia klienta z serwerem, jeśli program obsługi SSL jest przypisany, wywołaj funkcję **GetInfo**, która w przypadku powodzenia zwróci dane połączenia.

```
oClient = new TsgcWebSocketClient();
oClient->URL = "wss://www.esegece.com:2053";
oClient->TLSOptions->Version = tls1_2;
oClient->TLSOptions->IOHandler = iohSChannel;
oClient->OnSSLAfterCreateHandler = OnSSLAfterCreateHandlerEvent;
oClient->OnConnect = OnConnectEvent;
oClient->Active = true;
void OnSSLAfterCreateHandlerEvent(TObject *Sender, TwssSSLHandler aType,
    TIdSSLIOHandlerSocketBase *aSSLHandler)
{
    if (aSSLHandler->ClassType() == __classid(TsgcIdSSLIOHandlerSocketSChannel))
    {
        SSL = dynamic_cast<TsgcIdSSLIOHandlerSocketSChannel*>(aSSLHandler);
    }
}
void OnConnectEvent(TsgcWSConnection *Connection)
{
    if (SSL != NULL)
    {
        TsgcSChannelConnectionInfo oInfo = SSL->GetInfo();
        if (oInfo->Protocol != tls1_2)
        {
            throw Exception("Client cannot connect using TLS 1.2");
        }
    }
}
```



# TsgcWebSocketClient | Klient wysła wiadomość tekstową

Po połączeniu klienta z serwerem możliwe jest wysyłanie wiadomości tekstowych. Aby wysłać wiadomość tekstową, należy wywołać metodę `WriteData()`.

## Wysyłanie wiadomości tekstowej

Wywołaj metodę **`WriteData()`** w celu wysłania wiadomości tekstowej. Metoda jest wykonywana w **tym samym wątku**, z którego jest wywoływana.

```
TsgcWebSocketClient1->WriteData("My First sgcWebSockets Message!.");
```

Jeśli **`QueueOptions.Text`** ma **wartość inną niż `qmNone`**, zamiast być przetwarzana w tym samym wątku, który ją wywołuje, zostanie przetworzona w wątku pomocniczym. Domyślnie ta opcja jest wyłączona.

## Wyślij wiadomość tekstową i poczekaj na odpowiedź

Wywołaj metodę `WriteAndWaitData()`, aby wysłać wiadomość tekstową i oczekiwać na odpowiedź serwera. Funkcja zwraca odebraną wiadomość tekstową.

```
TsgcWebSocketClient1->WriteAndWaitData("My First sgcWebSockets Message!.");
```

## TsgcWebSocketClient | Wysyłanie binarnej wiadomości przez klienta

Po nawiązaniu połączenia klienta z serwerem możliwe jest wysyłanie wiadomości binarnych. Aby wysłać wiadomość binarną, należy wywołać metodę `WriteData()`.

### Wyślij wiadomość binarną

Wywołaj metodę **`WriteData()`** w celu wysłania wiadomości binarnej. Metoda jest wykonywana w **tym samym wątku**, z którego jest wywoływana.

```
TMemoryStream oStream = new TMemoryStream();  
...  
TsgcWebSocketClient1->WriteData(oStream);  
delete oStream;
```

Jeśli właściwość **`QueueOptions.Binary`** ma **wartość różną od `qmNone`**, przetwarzanie odbywa się na wątku pomocniczym zamiast na wątku wywołującym. Domyślnie opcja ta jest wyłączona.

### Wyślij wiadomość binarną i oczekuj na odpowiedź

Wywołaj metodę `WriteAndWaitData()`, aby wysłać wiadomość binarną i oczekiwać na odpowiedź serwera. Funkcja zwraca otrzymaną wiadomość binarną.

```
TsgcWebSocketClient1->WriteAndWaitData(oStream);
```

## TsgcWebSocketClient | Klient wysyła wiadomość tekstową i binarną

Protokół WebSocket dopuszcza tylko dwa typy wiadomości: tekstowe lub binarne. Nie można jednak przesyłać danych binarnych razem z tekstem w tej samej wiadomości.

Jednym ze sposobów rozwiązania tego problemu jest dodanie nagłówka do wiadomości binarnej przed jej wysłaniem i dekodowanie tej wiadomości binarnej po jej odebraniu.

W module `sgcWebSocket_Helpers` dostępne są 2 funkcje, które można wykorzystać do ustawienia krótkiego opisu pakietu binarnego. Dodają one nagłówek do strumienia służący do identyfikacji pakietu binarnego.

**Przed wysłaniem wiadomości binarnej należy wywołać metodę kodowania strumienia.**

```
sgcWSStreamWrite("00001", oStream);  
TsgcWebSocketClient1->WriteData(oStream);
```

**Po odebraniu wiadomości binarnej należy wywołać metodę w celu zdekodowania strumienia.**

```
sgcWSStreamRead(oStream, vID);
```

Jedynym ograniczeniem jest to, że tekst używany do identyfikacji wiadomości binarnej ma maksymalną długość 10 znaków (można to zmienić, jeśli jest dostęp do kodu źródłowego).

## TsgcWebSocketClient | Odbieranie wiadomości tekstowych

Gdy klient otrzyma wiadomość tekstową, wywoływane jest zdarzenie **OnMessage**. Należy odczytać parametr `Text`, aby pobrać ciąg odebranej wiadomości.

```
void OnMessage(TsgcWSConnection *Connection, const string Text)
{
    ShowMessage("Message Received from Server: " + Text);
}
```

Domyślnie klient używa metody **neAsynchronous** do wywoływania zdarzenia `OnMessage`, co oznacza, że **zdarzenie to jest wykonywane w kontekście głównego wątku**, dzięki czemu aktualizacja kontrolek formularza jest bezpieczna wątkowo.

Jeśli klient odbiera dużą liczbę wiadomości lub konieczna jest kontrola synchronizacji z innymi wątkami, należy ustawić właściwość `NotifyEvents` na **neNoSync**. Oznacza to, że zdarzenie `OnMessage` będzie **wykonywane w kontekście wątku połączenia**. Jeśli wymagana jest aktualizacja jakiegokolwiek kontrolki formularza lub dostęp do współdzielonych obiektów, konieczne jest zaimplementowanie własnych metod synchronizacji.

## TsgcWebSocketClient | Odbieranie wiadomości binarnych

Gdy klient odbierze wiadomość binarną, wywoływane jest zdarzenie **OnBinary**. Należy odczytać parametr *Data*, aby pobrać odebraną wiadomość binarną.

```
void OnBinary(TsgcWSConnection *Connection, const TMemoryStream *Data)
{
    oBitmap = new TBitmap();
    oBitmap->LoadFromStream(Data);
    Image1->Picture->Assign(oBitmap);
    Log(
        "#image uncompressed size: " + IntToStr(Data->Size) +
        ". Total received: " + IntToStr(Connection->RecBytes));
    delete oBitmap;
}
```

Domyślnie klient używa metody **neAsynchronous** do wysyłania zdarzenia *OnMessage*, co oznacza, że **to zdarzenie jest wykonywane w kontekście wątku głównego**, więc jest bezpieczne wątkowo aktualizowanie na przykład dowolnej kontrolki formularza.

Jeśli klient odbiera wiele wiadomości lub konieczna jest kontrola synchronizacji z innymi wątkami, należy ustawić właściwość *NotifyEvents* na **neNoSync**. Oznacza to, że zdarzenie *OnMessage* będzie **wykonywane w kontekście wątku połączenia**, więc jeśli wymagana jest aktualizacja kontrolki formularza lub dostęp do współdzielonych obiektów, należy zaimplementować własne metody synchronizacji.

# TsgcWebSocketClient | Uwierzytelnianie klienta

Klient TsgcWebSocket obsługuje 4 typy uwierzytelniania:

- **Basic:** wysyła nagłówek HTTP podczas uzgadniania WebSocket z nazwą użytkownika i hasłem zakodowanymi jako Basic Authentication.
- **Token:** wysyła Token jako nagłówek HTTP podczas uzgadniania WebSocket. Ustaw wymagany token w Authentication.Token.AuthToken zgodnie z wymaganiami serwera.
- **Session:** klient najpierw żąda sesji HTTP od serwera, a jeśli serwer zwraca sesję, jest ona przekazywana w nagłówku HTTP GET uzgodnienia WebSocket. (\* własna metoda uwierzytelniania biblioteki sgcWebSockets).
- **URL:** klient żąda autoryzacji przy użyciu nagłówka HTTP GET uzgodnienia WebSocket. (\* własna metoda uwierzytelniania biblioteki sgcWebSockets).

## Uwierzytelnianie Basic

Jest to prosta metoda autoryzacji, w której nazwa użytkownika i hasło są kodowane i przekazywane jako nagłówek HTTP. Wystarczy ustawić pola User i Password oraz włączyć wyłącznie typ autoryzacji Basic, aby skorzystać z tej metody.

```
oClient = new TsgcWebSocketClient();
oClient->Authentication->Enabled = true;
oClient->Authentication->Basic->Enabled = true;
oClient->Authentication->User = "your user";
oClient->Authentication->Password = "your password";
oClient->Authentication->Token->Enabled = false;
oClient->Authentication->URL->Enabled = false;
oClient->Authentication->Session->Enabled = false;
oClient->Active = true;
```

## Token uwierzytelniający

Umożliwia autoryzację przy użyciu JWT. Wymaga wcześniejszego uzyskania tokenu za pomocą narzędzia zewnętrznego (na przykład połączenia HTTP, OAuth2 itp.).

Jeśli dołączony jest komponent OAuth2, token można uzyskać automatycznie. Więcej informacji na temat [OAuth2](#).

Należy ustawić token AuthToken i włączyć uwierzytelnianie tokenem.

```
oClient = new TsgcWebSocketClient();
oClient->Authentication->Enabled = true;
oClient->Authentication->Token->Enabled = true;
oClient->Authentication->Token->AuthToken = "your token";
oClient->Authentication->Basic->Enabled = false;
oClient->Authentication->URL->Enabled = false;
oClient->Authentication->Session->Enabled = false;
oClient->Active = true;
```

## Sesja uwierzytelniania

Najpierw klient łączy się z serwerem przy użyciu połączenia HTTP, żądając nowej sesji. W przypadku powodzenia serwer zwraca SessionId, a klient wysyła ten SessionId w nagłówku HTTP GET podczas uzgadniania WebSocket.

Wymaga ustawienia nazwy użytkownika i hasła oraz włączenia uwierzytelniania sesji.

```
oClient = new TsgcWebSocketClient();
oClient->Authentication->Enabled = true;
oClient->Authentication->Session->Enabled = true;
oClient->Authentication->User = "your user";
oClient->Authentication->Password = "your password";
oClient->Authentication->Basic->Enabled = false;
oClient->Authentication->URL->Enabled = false;
oClient->Authentication->Token->Enabled = false;
oClient->Active = true;
```

## URL uwierzytelniania

Ta metoda uwierzytelniania przekazuje nazwę użytkownika i hasło w nagłówku HTTP GET podczas uzgadniania WebSocket.

```
oClient = new TsgcWebSocketClient();
oClient->Authentication->Enabled = true;
oClient->Authentication->URL->Enabled = true;
oClient->Authentication->User = "your user";
oClient->Authentication->Password = "your password";
oClient->Authentication->Basic->Enabled = false;
oClient->Authentication->Session->Enabled = false;
oClient->Authentication->Token->Enabled = false;
oClient->Active = true;
```

# TsgcWebSocketClient | Wyjątki klienta

Niekiedy w komunikacji zdarzają się błędy: serwer może rozłączyć połączenie, ponieważ nie jest ono autoryzowane lub wiadomość ma nieprawidłowy format. Istnieją 2 zdarzenia przechwytyjące błędy.

## OnError

To zdarzenie jest wywoływane za każdym razem, gdy wystąpi błąd protokołu WebSocket, taki jak nieprawidłowy typ wiadomości, nieprawidłowy ciąg utf8 itp.

```
void OnError(TsgcWSConnection *Connection, const string Error)
{
    WriteLn("#error: " + Error);
}
```

## OnException

To zdarzenie jest wywoływane za każdym razem, gdy wystąpi wyjątek, na przykład podczas zapisu do nieaktywnego gniazda lub uzyskiwania dostępu do nieistniejącego obiektu.

```
void OnException(TsgcWSConnection *Connection, Exception *E)
{
    WriteLn("#exception: " + E->Message);
}
```

Domyślnie, gdy **połączenie jest zamykane przez serwer, zostaje zgłoszony wyjątek**. Jeśli nie chcesz, aby te wyjątki były zgłaszane, wyłącz **Options.RaiseDisconnectExceptions**.



# TsgcWebSocketClient | Uzgadnianie WebSocket

Protokół WebSocket używa uzgodnienia HTTP do przełączenia z protokołu HTTP na WebSocket. Uzgodnienie to jest obsługiwane wewnętrznie przez komponent klienta TsgcWebSocket, ale można dodawać własne niestandardowe nagłówki HTTP, jeśli serwer wymaga dodatkowych informacji w nagłówkach HTTP.

**Przykład:** jeśli chcesz dodać nagłówek HTTP „Client: sgcWebSockets”

```
void OnHandshake(TsgcWSConnection *Connection, ref TStringList *Headers)
{
    Headers->Add("Client: sgcWebSockets");
}
```

## TsgcWebSocketClient | Protokół rejestracji klienta

Domyślnie TsgcWebSocketClient nie używa żadnego podprotokołu. Podprotokoły WebSocket są zbudowane na bazie protokołu WebSocket i definiują niestandardowy protokół wiadomości. Przykłady podprotokołów WebSocket to MQTT, STOMP itp.

Nazwa podprotokołu WebSocket jest wysyłana jako nagłówek HTTP podczas uzgadniania WebSocket. Nagłówek ten jest przetwarzany przez serwer, który w przypadku obsługi tego podprotokołu zaakceptuje połączenie. Jeśli podprotokół nie jest obsługiwany, połączenie zostanie automatycznie zamknięte.

**Przykład:** połączenie z serwerem WebSocket z nazwą SubProtocol 'myprotocol'

```
Client = new TsgcWebSocketClient();
Client->Host = "server host";
Client->Port = server.port;
Client->RegisterProtocol("myprotocol");
Client->Active = true;
```

## TsgcWebSocketClient | Proxy klienta

Klient TsgcWebSocket obsługuje połączenia przez serwery proxy. Aby skonfigurować połączenie przez serwer proxy, należy wypełnić właściwości **Proxy** klienta TsgcWebSocket.

```
Client = new TsgcWebSocketClient();
Client->Proxy->Enabled = true;
Client->Proxy->Username = "user";
Client->Proxy->Password = "secret";
Client->Proxy->Host = "80.55.44.12";
Client->Proxy->Port = 8080;
Client->Active = true;
```

# TsgcTCPClient

TsgcTCPClient to komponent klienta TCP umożliwiający łączenie się z serwerami zwykłego TCP.

## Wprowadzenie

TsgcTCPClient to komponent klienta TCP umożliwiający nawiązywanie połączeń z prostymi serwerami TCP. Zapewnia obsługę połączeń TLS/SSL, serwerów proxy, monitorowania bicia serca, automatycznego ponownego łączenia (WatchDog) oraz funkcji TCP keep-alive.

1. Upuść komponent **TsgcTCPClient** na formularz.
2. Ustaw właściwości **Host** i **Port** (domyślnie 80), aby połączyć się z dostępnym serwerem TCP.
3. Ustaw **TLS** na True, jeśli wymagane jest bezpieczne połączenie (domyślnie False).
4. Opcjonalnie skonfiguruj **Proxy**, jeśli połączenie odbywa się przez serwer proxy.
5. Obsługuj zdarzenia: **OnConnect**, **OnDisconnect**, **OnMessage**, **OnBinary**, **OnError**, **OnException**.
6. Ustaw właściwość **Active** na True, aby otworzyć połączenie.

## Dokumentacja

- Właściwości — 23 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 12 obsługiwanych zdarzeń.

# TsgcTCPClient

TsgcTCPClient to komponent klienta TCP umożliwiający łączenie się z serwerami zwykłego TCP.

## Wprowadzenie

TsgcTCPClient to komponent klienta TCP umożliwiający nawiązywanie połączeń z prostymi serwerami TCP. Zapewnia obsługę połączeń TLS/SSL, serwerów proxy, monitorowania bicia serca, automatycznego ponownego łączenia (WatchDog) oraz funkcji TCP keep-alive.

1. Upuść komponent **TsgcTCPClient** na formularz.
2. Ustaw właściwości **Host** i **Port** (domyślnie 80), aby połączyć się z dostępnym serwerem TCP.
3. Ustaw **TLS** na True, jeśli wymagane jest bezpieczne połączenie (domyślnie False).
4. Opcjonalnie skonfiguruj **Proxy**, jeśli połączenie odbywa się przez serwer proxy.
5. Obsługuj zdarzenia: **OnConnect**, **OnDisconnect**, **OnMessage**, **OnBinary**, **OnError**, **OnException**.
6. Ustaw właściwość **Active** na True, aby otworzyć połączenie.

## Dokumentacja

- Właściwości — 23 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 12 obsługiwanych zdarzeń.

# TsgcWebSocketServer

TsgcWebSocketServer implementuje serwerowy komponent WebSocket i może obsługiwać wiele połączeń klientów w osobnych wątkach.

## Wprowadzenie

TsgcWebSocketServer implementuje komponent serwera WebSocket i może obsługiwać wiele wielowątkowych połączeń klienta. Aby skonfigurować ten komponent, należy wykonać poniższe kroki:

1. Upuść komponent TsgcWebSocketServer na formularz
2. Ustaw Port (domyślnie 80). Jeśli jesteś za zaporą sieciową, prawdopodobnie konieczne będzie jej skonfigurowanie.
3. Należy ustawić dozwolone specyfikacje; domyślnie wszystkie specyfikacje są dozwolone.

**RFC6455:** jest standardową i zalecaną specyfikacją WebSocket.

**Hixie76:** jest to wersja robocza i zaleca się nawiązywanie połączeń Hixie76 wyłącznie w celu zapewnienia obsługi starych przeglądarek, takich jak Safari 4.2

```
void __fastcall TForm1::OnTCPConnectEvent(TsgcWSConnection *aConnection, bool &Accept)
{
    aConnection->Transport = trpTCP;
    Accept = true;
}
```

5. Utwórz procedurę i ustaw właściwość Active = True.

## Dokumentacja

- Właściwości — 27 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 22 zdarzenia do obsłużenia.

# Przewodniki

- **Start**
  - [Uruchomienie serwera](#)
  - [Powiązania serwera](#)
  - [Uruchamianie i zatrzymywanie serwera](#)
  - [Utrzymywanie aktywności serwera](#)
- **Połączenia**
  - [Utrzymywanie połączeń serwera przy życiu](#)
  - [Serwer — zwykłe TCP](#)
  - [Serwer — zamykanie połączenia](#)
  - [Połączenia klientów](#)
- **Uwierzytelnianie**
  - [Uwierzytelnianie serwera](#)
- **Wysyłanie wiadomości**
  - [Serwer wysyła wiadomość tekstową](#)
  - [Serwer wysyła wiadomość binarną](#)
- **Odbieranie wiadomości**
  - [Odbieranie wiadomości tekstowych przez serwer](#)
  - [Odbieranie wiadomości binarnych przez serwer](#)
- **SSL**
  - [SSL serwera](#)
  - [SSL serwera SChannel](#)

# TsgcWebSocketServer | Uruchamianie serwera

Pierwszą czynnością, którą należy wykonać przed uruchomieniem serwera, jest ustawienie portu nasłuchiwania. Domyślnie jest to port 80, ale można go zmienić na dowolny inny port.

Po ustawieniu portu dostępne są 2 metody uruchamiania serwera.

## Właściwość Active

Jeśli właściwość Active zostanie ustawiona na true, serwer rozpocznie nasłuchiwanie wszystkich przychodzących połączeń na skonfigurowanym porcie.

```
oServer = new TsgcWebSocketServer();
oServer->Port = 80;
oServer->Active = true;
```

Jeśli właściwość Active zostanie ustawiona na false, serwer zatrzyma się i zamknie wszystkie aktywne połączenia.

```
oServer->Active = false;
```

## Metody Start / Stop

Ustawienie właściwości Active uruchamia lub zatrzymuje serwer w tym samym wątku, natomiast metody Start i Stop są wykonywane w wątku pomocniczym.

```
oServer = new TsgcWebSocketServer();
oServer->Port = 80;
oServer->Start();
```

Wywołanie metody Stop() spowoduje zatrzymanie serwera i zamknięcie wszystkich aktywnych połączeń.

```
oServer->Stop();
```

Metody **ReStart** można użyć do zatrzymania i uruchomienia serwera w wątku pomocniczym.

Jeśli po zamknięciu serwera zmieniony zostanie port, aby serwer nasłuchiwał na innym porcie, należy wywołać metodę **Bindings.Clear()** po zamknięciu serwera w celu usunięcia wszystkich poprzednich bindingów. W przeciwnym razie serwer spróbuje się połączyć z poprzednimi bindingami.



## TsgcWebSocketServer | Powiązania serwera

Domyślnie, jeśli wypełniona jest wyłącznie **właściwość Port**, serwer **wiąże port nasłuchiwanie na WSZYSTKICH adresach IP**; jeśli na przykład istnieją 3 adresy IP: 127.0.0.1, 80.54.11.22 i 12.55.41.17, serwer zwiąże ten port na wszystkich 3 adresach IP.

Zazwyczaj zaleca się powiązanie tylko z potrzebnymi adresami IP. Właściwość Bindings służy właśnie do tego.

Zamiast właściwości Port należy użyć właściwości Bindings i wypełnić wymagany adres IP i port.

**Przykład:** powiąż Port 5555 z adresem IP 127.0.0.1 i adresem IP 80.58.25.40

```
oServer = new TsgcWebSocketServer();
binding = oServer->Bindings->Add();
binding->IP = "127.0.0.1";
binding->Port = 5555;
binding = oServer->Bindings->Add();
binding->IP = "80.58.25.40";
binding->Port = 5555;
oServer->Active = true;
```

Jeśli zmienisz port po zamknięciu serwera, aby rozpocząć nasłuchiwanie na innym porcie, wywołaj metodę **Bindings.Clear()** po zamknięciu serwera, aby usunąć wszystkie poprzednie powiązania. W przeciwnym razie serwer będzie próbował powiązać się z poprzednimi powiązaniami.

# TsgcWebSocketServer | Uruchamianie i zatrzymywanie serwera

Po skonfigurowaniu wszystkich wymaganych ustawień serwera dostępne są 2 przydatne zdarzenia informujące o uruchomieniu i zatrzymaniu serwera.

## OnStartup

Zdarzenie jest wywoływane, gdy serwer uruchomił się i może przetwarzać nowe połączenia.

```
void OnStartup(TObject *Sender)
{
    WriteLn("#server started");
}
```

## OnShutdown

To zdarzenie jest wywoływane po zatrzymaniu serwera i zaprzestaniu akceptowania połączeń.

```
void OnShutdown(TObject *Sender)
{
    WriteLn("#server stopped");
}
```

# TsgcWebSocketServer | Utrzymywanie aktywności serwera

Po uruchomieniu serwera może on czasem zatrzymać się z dowolnego powodu. Jeśli chcesz ponownie uruchomić serwer po nieoczekiwanym wyłączeniu, możesz użyć właściwości WatchDog.

## WatchDog

Jeśli WatchDog jest włączony, to gdy serwer wykryje wyłączenie, WatchDog będzie próbował ponownie uruchomić serwer co X sekund do momentu, gdy serwer będzie znowu aktywny.

**Przykład:** ponowne uruchomienie co 10 sekund po nieoczekiwanym zatrzymaniu, bez ograniczenia liczby prób.

```
oServer = new TsgcWebSocketServer();  
oServer->WatchDog->Interval = 10;  
oServer->WatchDog->Attempts = 0;  
oServer->WatchDog->Enabled = true;  
oServer->Active = true;
```

## TsgcWebSocketServer | SSL serwera

Serwer można skonfigurować do używania **certyfikatów SSL**. Aby uruchomić serwer produkcyjny z certyfikatem serwera, należy **zakupić** certyfikat od **renomowanego dostawcy**: Namecheap, GoDaddy, Thawte itp. Do celów **testowych** można użyć **certyfikatu z podpisem własnym** (patrz przykład Demos/Chat, który używa certyfikatu z podpisem własnym).

Certyfikat musi być w formacie **PEM**. PEM (Privacy Enhanced Mail) jest zdefiniowany w RFC 1421–1424; jest to format kontenera, który może zawierać tylko certyfikat publiczny (jak w instalacjach Apache i plikach certyfikatów CA /etc/ssl/certs) lub może zawierać pełny łańcuch certyfikatów, w tym klucz publiczny, klucz prywatny i certyfikaty główne. Aby utworzyć pojedynczy certyfikat PEM, należy otworzyć plik klucza prywatnego, skopiować jego zawartość i wkleić do pliku certyfikatu.

### Przykład:

certificate.crt

```
-----BEGIN CERTIFICATE-----  
.....  
-----END CERTIFICATE-----
```

certificate.key

```
-----BEGIN PRIVATE KEY-----  
.....  
-----END PRIVATE KEY-----
```

certificate.pem

```
-----BEGIN PRIVATE KEY-----  
.....  
-----END PRIVATE KEY-----  
-----BEGIN CERTIFICATE-----
```

.....

-----END CERTIFICATE-----

Aby włączyć SSL, wystarczy **włączyć właściwość SSL** i skonfigurować ścieżki do plików **CertFile**, **KeyFile** oraz **RootFile**. Jeśli certyfikat zawiera pełny łańcuch certyfikatów (klucz publiczny, klucz prywatny itp.), należy ustawić wszystkie ścieżki na ten sam certyfikat.

Kolejną właściwością, którą należy ustawić, jest **SSLOptions.Port** — jest to port używany dla bezpiecznych połączeń.

## Prosta konfiguracja SSL

**Przykład:** konfiguracja SSL na adresie IP 127.0.0.1 i porcie 443

```
oServer = new TsgcWebSocketServer();
oServer->SSL = true;
oServer->SSLOptions->CertFile = "c:\\certificates\\mycert.pem";
oServer->SSLOptions->KeyFile = "c:\\certificates\\mycert.pem";
oServer->SSLOptions->RootCertFile = "c:\\certificates\\mycert.pem";
oServer->SSLOptions->Port = 443;
oServer->Port = 443;
oServer->Active = true;
```

## SSL i bez SSL

Serwer można skonfigurować do nasłuchiwanie na więcej niż jednym adresie IP i porcie; zapoznaj się z [artykułem dotyczącym powiązań](#), który wyjaśnia zasadę działania. Serwer można skonfigurować tak, aby jednocześnie akceptował połączenia SSL i bez SSL (nasłuchując oczywiście na różnych portach). Wystarczy powiązać się z dwoma różnymi portami i skonfigurować jeden port dla połączeń SSL, a drugi dla połączeń bez SSL.

**Przykład:** konfiguracja serwera na adresie IP 127.0.0.1, porcie 80 (bez szyfrowania) i 443 (SSL)

```
oServer = new TsgcWebSocketServer();
bind = oServer->Bindings->Add();
bind->IP = "127.0.0.1";
bind->Port = 80;
bind = oServer->Bindings->Add();
bind->IP = "127.0.0.1";
bind->Port = 443;
```

```
oServer->Port = 80;  
oServer->SSL = true;  
oServer->SSLOptions->CertFile = "c:\\certificates\\mycert.pem";  
oServer->SSLOptions->KeyFile = "c:\\certificates\\mycert.pem";  
oServer->SSLOptions->RootCertFile = "c:\\certificates\\mycert.pem";  
oServer->SSLOptions->Port = 443;  
oServer->Active = true;
```

# Serwer SSL | SChannel dla serwerów Indy

Komponenty serwerowe oparte na Indy (**TsgcWebSocketServer**, **TsgcWebSocketHTTPServer**) mogą używać **Windows SChannel** (Secure Channel) jako dostawcy TLS zamiast OpenSSL. SChannel jest natywną implementacją TLS systemu Windows, dlatego nie wymaga zewnętrznych bibliotek DLL.

## Jak to działa

Gdy komponent serwera ma włączony SSL i IOHandler jest ustawiony na **iohSChannel**, serwer tworzy instancję **TsgcIdServerIOHandlerSSLSChannel**, która obsługuje wszystkie operacje TLS za pomocą Windows SChannel API. Dla każdego przychodzącego połączenia klienta serwer wykonuje uzgadnianie TLS przez dostawcę SChannel, negocjując wersję protokołu, zestaw szyfrów i wiążąc skonfigurowany certyfikat.

SChannel odczytuje certyfikaty z **magazynu certyfikatów systemu Windows** lub z **pliku PFX** (.pfx / .p12). Pliki PEM nie są wymagane i nie trzeba wdrażać bibliotek OpenSSL.

## Konfiguracja

Aby włączyć SChannel na serwerze Indy, skonfiguruj następujące właściwości:

1. Ustaw właściwość **SSL** na **True**.
2. Ustaw **SSLOptions.IOHandler** na **iohSChannel**.
3. Ustaw właściwość **SSLOptions.Version** na żadaną wersję TLS (tls1\_2, tls1\_3, ...).
4. Ustaw wartość **SSLOptions.Port** na port używany do bezpiecznych połączeń.
5. Skonfiguruj certyfikat przy użyciu jednej z dwóch opisanych poniżej metod.

## Właściwości SChannel\_Options

Właściwość podrzędna **SSLOptions.SChannel\_Options** udostępnia ustawienia specyficzne dla SChannel:

|                             |                                                                                                                                                                                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CertHash</b>             | Odcisk palca (skrót w formacie szesnastkowym) certyfikatu zainstalowanego w magazynie certyfikatów systemu Windows.                                                                                           |
| <b>CertStoreName</b>        | Magazyn certyfikatów do przeszukania: <b>scsnMY</b> (Osobisty), <b>scsnRoot</b> , <b>scsnTrust</b> , <b>scsnCA</b> .                                                                                          |
| <b>CertStorePath</b>        | Lokalizacja magazynu: <b>scspStoreLocalMachine</b> lub <b>scspStoreCurrentUser</b> .                                                                                                                          |
| <b>CipherList</b>           | Opcjonalna lista algorytmów szyfrowania oddzielona dwukropkami (np. <b>CALG_AES_256:CALG_AES_128</b> ). Pusta wartość oznacza użycie domyślnych ustawień systemowych.                                         |
| <b>UseLegacyCredentials</b> | Gdy wartość wynosi True, zamiast <b>SCH_CREDENTIALS</b> używana jest starsza struktura <b>SCHANNEL_CRED</b> . Należy włączyć tę opcję w starszych wersjach systemu Windows, które nie obsługują nowszego API. |

Ponadto ogólne właściwości **SSLOptions CertFile** i **Password** są używane podczas ładowania certyfikatu z pliku PFX.

## Certyfikat z magazynu systemu Windows

Jeśli certyfikat jest już zainstalowany w magazynie certyfikatów Windows, należy podać **odcisk palca** certyfikatu i wskazać jego lokalizację.

Aby znaleźć odcisk palca, otwórz **PowerShell** i uruchom:

```
dir cert:\localmachine\my
```

Kolumna **Thumbprint** pokazuje szesnastkowy skrót, który jest potrzebny.

```
Directory: Microsoft.PowerShell.Security\Certificate::localmachine\my
Thumbprint                               Subject
-----
C12A8FC8AE668F866B48F23E753C93D357E9BE10  CN=*.mydomain.com
```

```
oServer = new TsgcWebSocketServer();
oServer->SSL = true;
oServer->SSLOptions->IOHandler = iohSChannel;
oServer->SSLOptions->Version = tls1_2;
oServer->SSLOptions->Port = 443;
oServer->Port = 443;
oServer->SSLOptions->SChannel_Options->CertHash = "C12A8FC8AE668F866B48F23E753C93D357E9BE10";
oServer->SSLOptions->SChannel_Options->CertStoreName = scsnMY;
oServer->SSLOptions->SChannel_Options->CertStorePath = scspStoreLocalMachine;
oServer->Active = true;
```

## Certyfikat z pliku PFX

Jeśli posiadasz plik certyfikatu PFX (.pfx lub .p12), ustaw właściwości **CertFile** i **Password** w **SSLOptions**. **SChannel** zaimportuje certyfikat podczas uruchamiania.

```
oServer = new TsgcWebSocketServer();
oServer->SSL = true;
oServer->SSLOptions->IOHandler = iohSChannel;
oServer->SSLOptions->Version = tls1_2;
oServer->SSLOptions->Port = 443;
oServer->Port = 443;
oServer->SSLOptions->CertFile = "c:\\certificates\\server.pfx";
oServer->SSLOptions->Password = "mypassword";
oServer->Active = true;
```

Jeśli posiadasz certyfikat PEM i klucz prywatny, należy je najpierw przekonwertować do formatu PFX przy użyciu OpenSSL:

```
openssl pkcs12 -inkey server.key -in server.crt -export -out server.pfx
```

## Wersja TLS

Użyj właściwości **SSLOptions.Version** do kontrolowania, które wersje TLS serwer akceptuje:

|        |                       |
|--------|-----------------------|
| tls1_0 | TLS 1.0 (niezalecane) |
|--------|-----------------------|



|              |                             |
|--------------|-----------------------------|
| tls1_1       | TLS 1.1                     |
| tls1_2       | TLS 1.2 (zalecane)          |
| tls1_3       | TLS 1.3                     |
| tlsUndefined | Akceptuj TLS 1.0, 1.1 i 1.2 |

## Lista szyfrów

Domyślnie SChannel używa systemowej konfiguracji szyfrowania. Można ograniczyć dozwolone szyfry, ustawiając **SChannel\_Options.CipherList** na listę nazw algorytmów oddzielonych dwukropkami, na przykład:

```
CALG_AES_256:CALG_AES_128
```

Pozostaw tę właściwość pustą, aby używać wartości domyślnych systemu Windows.

## Starsze poświadczenia

Windows Server 2019 i wcześniejsze wersje mogą nie obsługiwać nowszego interfejsu API **SCH\_CREDENTIALS**. Jeśli uruchomienie serwera na starszej wersji systemu Windows nie powiedzie się, należy ustawić **SChannel\_Options.UseLegacyCredentials** na **True**, aby użyć starszej struktury **SCHANNEL\_CRED**.

Komponent automatycznie wykrywa wersję systemu Windows w większości przypadków, ale w razie potrzeby można wymusić tryb legacy.

## Zalety w porównaniu z OpenSSL

- **Brak zewnętrznych bibliotek DLL:** SChannel jest wbudowany w system Windows, dlatego nie ma potrzeby wdrażania plików libeay32.dll / ssleay32.dll ani libcrypto / libssl.
- **Integracja z magazynem certyfikatów systemu Windows:** korzystanie z certyfikatów już zainstalowanych i zarządzanych przez system operacyjny.
- **Automatyczne aktualizacje:** Ulepszenia TLS i poprawki zabezpieczeń są stosowane za pośrednictwem usługi Windows Update.

## Uwagi

- SChannel jest dostępny wyłącznie w systemie **Windows**. W przypadku serwerów wieloplatformowych należy używać OpenSSL (iohOpenSSL).
- Serwer musi posiadać **klucz prywatny** powiązany z certyfikatem. W przypadku korzystania z metody magazynu Windows certyfikat musi zostać zaimportowany wraz z kluczem prywatnym.
- W przypadku serwerów produkcyjnych korzystających z metody magazynu certyfikatów zalecane jest użycie magazynu **Komputer lokalny** (scspStoreLocalMachine), dzięki czemu certyfikat jest dostępny niezależnie od konta użytkownika uruchamiającego usługę.
- Obsługiwane są wyłącznie pliki certyfikatów **PFX** (.pfx / .p12). Jeśli posiadane są pliki PEM, należy najpierw przekonwertować je do formatu PFX.

# TsgcWebSocketServer | Weryfikacja certyfikatu serwera

Domyślnie serwer nie weryfikuje certyfikatów peer. Aby skonfigurować serwer do weryfikacji certyfikatów klientów, należy wykonać następujące kroki:

1. Ustaw właściwość `SSLOptions.VerifyCertificate = true`

Obsłuż zdarzenie `OnSSLVerifyPeer` i zaimplementuj poniższy kod, aby otrzymywać powiadomienia za każdym razem, gdy klient łączy się z certyfikatem.

```
void OnSSLVerifyPeer(System::TObject* Sender, TIdX509* Certificate, bool &Accept)
{
    // ... validate the certificate
    if (Certificate_OK)
        Accept = true;
    else
        Accept = false;
}
```

Należy pamiętać, że zdarzenie **OnSSLVerifyPeer** jest wywoływane **tylko wtedy, gdy klient dostarcza certyfikat**; jeśli klient nie dostarcza certyfikatu, zdarzenie nie jest wyzwalane.

Można skonfigurować **serwer tak, aby akceptował wyłącznie połączenia SSL wymagające certyfikatu**. W tym celu należy ustawić następującą właściwość:

- `SSLOptions.VerifyCertificate_Options.FailIfNoCertificate = true`

Jeśli klient nie dostarczy certyfikatu, połączenie zostanie zamknięte podczas uzgodnienia SSL.

# TsgcWebSocketServer | Utrzymywanie połączeń aktywnych

Po nawiązaniu połączenia klienta z serwerem połączenie może zostać zerwane z powodu słabego sygnału, błędów połączenia itp. Należy użyć funkcji HeartBeat, aby utrzymać połączenie aktywne.

## HeartBeat

Właściwość **HeartBeat** umożliwia **wysyłanie Ping** co **X sekund** w celu **podtrzymania aktywnego połączenia**. Niektóre klienty zamykają połączenia TCP, gdy między węzłami nie jest wymieniana żadna komunikacja. HeartBeat rozwiązuje ten problem, wysyłając ping w określonym interwale. Zazwyczaj jest to wystarczające do utrzymania aktywnego połączenia, jednak można ustawić interwał TimeOut, aby zamknąć połączenie, gdy odpowiedź od klienta nie zostanie odebrana po X sekundach.

**Przykład:** wysyłanie polecenia ping do wszystkich połączonych klientów co 30 sekund

```
oServer = new TsgcWebSocketServer();
oServer->HeartBeat->Interval = 30;
oServer->HeartBeat->Timeout = 0;
oServer->HeartBeat->Enabled = true;
oServer->Active = true;
```

# TsgcWebSocketServer | Serwer Plain TCP

Serwer WebSocket akceptuje połączenia WebSocket, HTTP, SSE i inne protokoły, ale może również działać ze zwykłymi połączeniami TCP. Więcej informacji na stronie [Połączenia TCP](#).

Do obsługi połączeń TCP dostępne są 2 zdarzenia.

## OnTCPConnect

Zdarzenie jest wywoływane po połączeniu klienta z serwerem, a przed jakimkolwiek uzgadnianiem między klientem a serwerem. Zdarzenie OnConnect jest uruchamiane dopiero po wysłaniu przez klienta wiadomości (co pozwala serwerowi wykryć używany protokół).

To zdarzenie informuje o próbie połączenia nowego klienta z serwerem; serwer może zaakceptować lub odrzucić połączenie. Domyślnie serwer zawsze akceptuje połączenie.

## OnUnknownProtocol

To zdarzenie jest wywoływane, gdy serwer odbierze pierwszą wiadomość od klienta, ale nie może wykryć, czy jest to jeden ze znanych protokołów. W tym zdarzeniu serwer może zaakceptować lub odrzucić protokół.

## OnConnect

Zdarzenie to jest wywoływane po pomyślnym i kompletnym nawiązaniu połączenia. W przypadku połączenia TCP bez szyfrowania jest wywoływane po zaakceptowaniu protokołu w zdarzeniu OnUnknownProtocol.

# TsgcWebSocketServer | Zamknięcie połączenia przez serwer

Pojedyncze połączenie można zamknąć za pomocą metod Close lub Disconnect.

## Rozłącz

[TsgcWSConnection](#) posiada metodę Disconnect(), która umożliwia rozłączenie połączenia na poziomie gniazda. Wywołanie tej metody powoduje natychmiastowe rozłączenie gniazda bez oczekiwania na odpowiedź od klienta. Za pomocą tej metody można wysłać kod zamknięcia.

## Close

[TsgcWSConnection](#) posiada metodę Close(), która umożliwia wysłanie do klienta żądania zamknięcia połączenia. Po otrzymaniu tej wiadomości klient musi zamknąć połączenie, a serwer otrzyma powiadomienie o jego zamknięciu. Za pomocą tej metody można wysłać kod zamknięcia.

## DisconnectAll

Rozłącza wszystkie aktywne połączenia. Ta metoda jest wywoływana automatycznie przed zatrzymaniem nasłuchiwanie przez serwer, jednak można ją wywołać w dowolnym momencie.

## TsgcWebSocketServer | Client Connections

Aby uzyskać dostęp do aktywnych połączeń klientów, można użyć właściwości `Connections` w celu iterowania po liście i uzyskania dostępu do klasy połączenia klienta. Właściwość `Connections` umożliwia dostęp do listy z obsługą wątków, dlatego należy najpierw zablokować listę, a po zakończeniu pracy ją odblokować.

```
void DoClientIPAddresses() {
    TList *oList;
    TsgcWSConnectionServer *oConnection;
    oList = TsgcWebSocketHTTPServer1->LockList();
    try {
        for (int i = 0; i < oList->Count; ++i) {
            oConnection = dynamic_cast<TsgcWSConnectionServer*>(static_cast<TIdContext*>(oList->Items[i])->Data);
            ShowMessage(oConnection->IP + ":" + IntToStr(oConnection->Port));
        }
    } __finally {
        TsgcWebSocketHTTPServer1->UnlockList();
    }
}
```

# TsgcWebSocketServer | Uwierzytelnianie serwera

Serwer TsgcWebSocket obsługuje 3 rodzaje uwierzytelniania:

- **Basic:** odczytuje nagłówek HTTP podczas uzgadniania WebSocket zawierający użytkownika i hasło zakodowane jako autoryzacja podstawowa (Basic Authorization).
- **Session:** najpierw klient żąda sesji HTTP od serwera, a jeśli serwer zwróci sesję, jest ona przekazywana w nagłówku GET HTTP uzgadniania WebSocket. (\* własna metoda autoryzacji biblioteki sgcWebSockets).
- **URL:** odczytuje autoryzację żądania przy użyciu nagłówka GET HTTP uzgadniania WebSocket. (\* własna metoda autoryzacji biblioteki sgcWebSockets).

Można ustawić listę uwierzytelnionych użytkowników za pomocą właściwości **AuthUsers** — wystarczy podać użytkowników w następującym formacie: użytkownik=hasło

## OnAuthentication

Za każdym razem, gdy serwer odbierze żądanie uwierzytelnienia od klienta, wywoływane jest to zdarzenie wskazujące, czy użytkownik jest uwierzytelniony, czy nie.

Należy użyć parametru **Authenticated**, aby zaakceptować lub odrzucić połączenie.

```
void OnAuthentication(TsgcWSConnection *Connection, string aUser, string aPassword,
    ref bool Authenticated)
{
    if ((aUser == "user") && (aPassword == "secret"))
    {
        Authenticated = true;
    }
    else
    {
        Authenticated = false;
    }
}
```

## OnUnknownAuthentication

Jeśli typ uwierzytelniania nie jest domyślnie obsługiwany, np. JWT, nadal można użyć tego zdarzenia, aby zaakceptować lub odrzucić połączenie. Wystarczy odczytać parametry i podjąć decyzję o akceptacji połączenia.

```
void OnUnknownAuthentication(TsgcWSConnection *Connection, string AuthType, string AuthData,
    ref string aUser, ref string aPassword, ref bool Authenticated)
{
    if (AuthType == "Bearer")
    {

```

```
    if (AuthData == "jwt_token")
    {
        Authenticated = true;
    }
    else
    {
        Authenticated = false;
    }
}
else
{
    Authenticated = false;
}
}
```



# TsgcWebSocketServer | Serwer wysyła wiadomość tekstową

Po nawiązaniu przez klienta połączenia z serwerem, serwer może wysyłać wiadomości tekstowe. Aby wysłać wiadomość tekstową, należy wywołać metodę `WriteData()` w celu wysłania wiadomości do pojedynczego klienta lub użyć metody `Broadcast`, aby wysłać wiadomość do wszystkich klientów.

## Wysyłanie wiadomości tekstowej

Wywołaj metodę **`WriteData()`**, aby wysłać wiadomość tekstową.

```
TsgcWebSocketServer1->WriteData("guid", "My First sgcWebSockets Message!.");
```

Jeśli **`QueueOptions.Text`** ma wartość inną niż **`qmNone`**, zamiast być przetwarzana w tym samym wątku, który ją wywołuje, zostanie przetworzona w wątku pomocniczym. Domyślnie ta opcja jest wyłączona.

**QueueOptions nie działa, gdy właściwość `IOHandlerOptions.IOHandlerType = iohIOCP` (ze względu na architekturę IOCP funkcja ta nie jest obsługiwana).**

Metodę `WriteData()` można również wywołać z obiektu **`TsgcWSConnection`**, przykład: wysłanie wiadomości do klienta po połączeniu z serwerem.

```
void OnConnect(TsgcWSConnection *Connection)
{
    Connection->WriteData("Hello From Server");
}
```

## Wysyłanie wiadomości do WSZYSTKICH połączonych klientów

Należy wywołać metodę **`Broadcast()`**, aby wysłać wiadomość tekstową do wszystkich podłączonych klientów.

```
TsgcWebSocketServer1->Broadcast("Hello From Server");
```

# TsgcWebSocketServer | Wysyłanie binarnej wiadomości przez serwer

Po połączeniu klienta z serwerem, serwer może wysyłać wiadomości binarne. Aby wysłać wiadomość binarną, należy wywołać metodę `WriteData()`, aby wysłać wiadomość do pojedynczego klienta, lub użyć `Broadcast`, aby wysłać wiadomość do wszystkich klientów.

## Wyślij wiadomość binarną

Wywołać metodę **`WriteData()`**, aby wysłać wiadomość binarną.

```
TsgcWebSocketServer1->WriteData("guid", new TMemoryStream());
```

Jeśli właściwość **`QueueOptions.Binary`** ma wartość inną niż **`qmNone`**, zamiast być przetwarzana w tym samym wątku, z którego została wywołana, zostanie przetworzona w wątku pomocniczym. Domyślnie ta opcja jest wyłączona.

**QueueOptions nie działa, gdy właściwość `IOHandlerOptions.IOHandlerType` = `iohIOCP` (ze względu na architekturę IOCP ta funkcja nie jest obsługiwana).**

Metodę `WriteData()` można również wywołać z obiektu **`TsgcWSCConnection`**, przykład: wysłanie wiadomości do klienta po połączeniu z serwerem.

```
void OnConnect(TsgcWSCConnection *Connection)
{
    Connection->WriteData(new TMemoryStream());
}
```

## Wysyłanie wiadomości do WSZYSTKICH połączonych klientów

Wywołaj metodę **`Broadcast()`**, aby wysłać wiadomość binarną do wszystkich połączonych klientów.

```
TsgcWebSocketServer1->Broadcast(new TMemoryStream());
```



## TsgcWebSocketServer | Odbiór wiadomości tekstowej przez serwer

Gdy serwer otrzyma wiadomość tekstową, wywoływane jest zdarzenie **OnMessage**. Należy odczytać parametr Text, aby pobrać ciąg znaków otrzymanej wiadomości.

```
void OnMessage(TsgcWSConnection *Connection, const string Text)
{
    ShowMessage("Message Received from Client: " + Text);
}
```

Domyślnie serwer używa metody **neAsynchronous** do wywoływania zdarzenia OnMessage, co oznacza, że **zdarzenie to jest wykonywane w kontekście głównego wątku**, dzięki czemu aktualizowanie dowolnej kontrolki formularza jest bezpieczne wątkowo.

Jeśli serwer odbiera dużą liczbę wiadomości lub konieczne jest kontrolowanie synchronizacji z innymi wątkami, należy ustawić właściwość NotifyEvents na **neNoSync**. Oznacza to, że zdarzenie OnMessage będzie **wykonywane w kontekście wątku połączenia**; jeśli więc wymagana jest aktualizacja kontrolek formularza lub dostęp do współdzielonych obiektów, należy zaimplementować własne metody synchronizacji.

## TsgcWebSocketServer | Serwer: odbiór wiadomości binarnej

Gdy serwer odbiera wiadomość binarną, wywoływane jest zdarzenie **OnBinary**. Należy odczytać parametr *Data*, aby uzyskać odebraną wiadomość binarną.

```
void OnBinary(TsgcWSConnection *Connection, const TMemoryStream *Data)
{
    oBitmap = new TBitmap();
    oBitmap->LoadFromStream(Data);
    Image1->Picture->Assign(oBitmap);
    Log(
        "#image uncompressed size: " + IntToStr(Data->Size) +
        ". Total received: " + IntToStr(Connection->RecBytes));
    oBitmap.Free();
}
```

Domyślnie serwer używa metody **neAsynchronous** do wywoływania zdarzenia *OnMessage*, co oznacza, że **zdarzenie to jest wykonywane w kontekście głównego wątku**, dzięki czemu aktualizowanie dowolnej kontrolki formularza jest bezpieczne wątkowo.

Jeśli serwer odbiera dużą liczbę wiadomości lub konieczne jest kontrolowanie synchronizacji z innymi wątkami, należy ustawić właściwość *NotifyEvents* na **neNoSync**. Oznacza to, że zdarzenie *OnMessage* będzie **wykonywane w kontekście wątku połączenia**; jeśli więc wymagana jest aktualizacja kontrolki formularza lub dostęp do współdzielonych obiektów, należy zaimplementować własne metody synchronizacji.

## TsgcWebSocketServer | Odczytywanie nagłówków z klienta przez serwer

Gdy **klient łączy się** z serwerem WebSocket, wysyła listę **nagłówków** z informacjami o połączeniu. Aby odczytać te nagłówki klienta, można użyć zdarzenia **OnHandshake** komponentu serwera, które jest wywoływane po otrzymaniu nagłówków od klienta i przed wysłaniem odpowiedzi.

Nagłówki klienta są przechowywane we właściwości **HeadersRequest** obiektu **TsgcWSConnectionServer**.

```
void OnServerHandshake(TsgcWSConnection *Connection, TStringList *Headers)
{
    ShowMessage(dynamic_cast<TsgcWSConnectionServer*>(Connection)->HeadersRequest->Text);
}
```

# TsgcWebSocketHTTPServer

AResponseInfo.

## Wprowadzenie

TsgcWebSocketHTTPServer implementuje komponent WebSocket po stronie serwera i obsługuje wiele połączeń klientów w wątkach, podobnie jak [TsgcWebSocketServer](#), a ponadto umożliwia serwowanie stron HTML za pośrednictwem wbudowanego serwera HTTP, współdzielącego ten sam port dla połączeń WebSocket i żądań HTTP.

Należy wykonać poniższe kroki w celu skonfigurowania tego komponentu:

1. Upuść komponent TsgcWebSocketHTTPServer na formularzu
2. Ustaw Port (domyślnie 80). Jeśli jesteś za zaporą sieciową, prawdopodobnie konieczne będzie jej skonfigurowanie.
3. Należy ustawić dozwolone specyfikacje; domyślnie wszystkie specyfikacje są dozwolone.

**RFC6455:** jest standardową i zalecaną specyfikacją WebSocket.

**Hixie76:** jest to wersja robocza i zaleca się nawiązywanie połączeń Hixie76 wyłącznie w celu zapewnienia obsługi starych przeglądarek, takich jak Safari 4.2

```
AResponseInfo.ContentText := '<HTML><HEADER>TEST</HEAD><BODY>Hello!</BODY></HTML>';
```

**OnHTTPUploadBeforeCreatePostStream:** to zdarzenie jest wywoływane po odczytaniu nagłówków i przed utworzeniem strumienia POST.

\* W niektórych przypadkach może wystąpić wysokie zużycie procesora spowodowane niepożądanymi połączeniami; w takich przypadkach należy zwrócić błąd 500 dla żądań HTTP lub zamknąć połączenie dla żądań nieznanego protokołu.

5. Utwórz procedurę i ustaw właściwość `Active = true`.

## Dokumentacja

- Właściwości — 33 właściwości dostępne na komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 36 zdarzeń do obsłużenia.

## Przewodniki

- **HTTP**
  - [Żądania serwera HTTP](#)
  - [Wysyłanie plików HTTP](#)
  - [Serwer HTTP/2](#)
  - [Server Push HTTP/2](#)
  - [Usługa alternatywna HTTP/2](#)
  - [Wątki serwera HTTP/2](#)
  - [Wysyłanie dużych plików HTTP](#)
  - [Błąd HTTP 404 bez treści odpowiedzi](#)
- **SSL**
  - [Server SSL SChannel](#)
- **Inne**
  - [HTTP Server Sessions](#)



# TsgcWebSocketHTTPServer | HTTP Server Requests

Należy użyć zdarzenia OnCommandGet do obsługi żądań HTTP od klientów. Dostępne są następujące parametry:

- **RequestInfo:** zawiera informacje o żądaniu HTTP.
- **ResponseInfo:** jest odpowiedzią HTTP na żądanie HTTP.
  - **ContentText:** jest odpowiedzią w formacie tekstowym.
  - **ContentType:** jest typem Content-Type.
  - **ResponseNo:** numer odpowiedzi HTTP, przykład: 200.

```
void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo,
    TidHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/")
    {
        AResponseInfo->ContentText = "<html><head><title>Test Page</title></head><body></body></html>";
        AResponseInfo->ContentType = "text/html";
        AResponseInfo->ResponseNo = 200;
    }
}
```

## OnBeforeCommand

To zdarzenie umożliwia dostosowanie odpowiedzi HTTP. Na przykład, jeśli niektóre punkty końcowe mają używać schematu autoryzacji, a inne mają być dostępne bez autoryzacji, należy użyć parametru options, aby to włączyć lub wyłączyć. Poniżej przedstawiono przykład, w którym włączona jest autoryzacja Basic, lecz gdy użytkownik żąda zasobu /public, autoryzacja nie jest wymagana.

```
void __fastcall TForm1::OnBeforeCommand(TsgcWSConnection* aConnection, TidHTTPRequestInfo* ARequestInfo, TidHTTPResponseInfo* AResponseInfo)
{
    if (ARequestInfo->Document == "/public")
        aOptions << hcoAuthorizedBasic;
}
```

## TsgcWebSocketHTTPServer | Dystrybucja plików HTTP

Gdy klient żąda pliku, wywoływane jest zdarzenie **OnCommandGet**, jednak do automatycznego dostarczania plików można użyć właściwości **DocumentRoot**.

**Przykład:** jeśli ustawisz **DocumentRoot** na **c:/www/files**. Za każdym razem, gdy zostanie zażądany nowy plik, serwer wyszuka go w tym folderze. Jeśli plik istnieje, zostanie automatycznie wysłany.

## TsgcWebSocketHTTPServer | Serwer HTTP/2

Serwer HTTP sgcWebSockets umożliwia obsługę żądań HTTP/1.1 i HTTP/2.0; protokół HTTP/2 można włączyć, korzystając z HTTP2Options serwera.

Należy ustawić **HTTP2Options.Enabled = true**, aby umożliwić serwerowi przyjmowanie żądań protokołu HTTP/2. Żądania mogą być przetwarzane przez użytkownika dokładnie tak samo jak w protokole HTTP/1.1, [czytaj więcej](#).

Gdy protokół HTTP/2 jest włączony, serwer nadal obsługuje żądania HTTP/1.1.

Domyślnie zdarzenia OnConnect i OnDisconnect nie są wywoływane przy nowym połączeniu HTTP/2, ale można to zmienić poprzez właściwości HTTP2Options.Events, gdzie można dostosować, czy ma być wywoływane powiadomienie za każdym razem, gdy pojawi się nowe połączenie lub rozłączenie HTTP/2.

## TsgcWebSocketHTTPServer | Serwer Push HTTP/2

HTTP zazwyczaj działa w oparciu o wzorzec żądanie/odpowiedź, gdzie klient wysyła ŻĄDANIE zasobu do SERWERA, a SERWER odsyła ODPOWIEDŹ z żądanym zasobem lub komunikatem błędu. Zazwyczaj klient, np. przeglądarka, wysyła serię żądań o zasoby dostarczane przez serwer.



Głównym problemem tego podejścia jest to, że klient musi najpierw wysłać żądanie, aby pobrać zasób (na przykład index.html), poczekać na odpowiedź serwera, odczytać zawartość, a następnie wysłać wszystkie pozostałe żądania (na przykład styles.css).

HTTP/2 server push stara się rozwiązać ten problem. Gdy klient żąda pliku, a serwer stwierdzi, że plik ten wymaga dodatkowych zasobów, pliki te zostaną automatycznie przesłane (PUSH) do klienta.



Na powyższym rzucie ekranu klient najpierw żąda pliku index.html. Serwer odczytuje to żądanie i wysyła 2 pliki jako odpowiedź: index.html i styles.css, unikając w ten sposób drugiego żądania o styles.css.

### Konfiguracja Server Push

Zgodnie z powyższymi zrzutami ekranu, można skonfigurować serwer tak, aby przy każdym żądaniu pliku /index.html serwer wysyłał index.html i styles.css.

Należy użyć metody **PushPromiseAddPreLoadLinks**, aby skojarzyć każde żądanie z listą push promise.

```
TsgcWebSocketHTTPServer *server = new TsgcWebSocketHTTPServer(this);
TStringList *oLinks = new TStringList();
try
{
    oLinks->Add("/styles.css");
    server->PushPromiseAddPreLoadLinks("/index.html", oLinks);
}
__finally
{
    oLinks->Free();
}

void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo, TidHTTPResponseInfo *AResponseInfo)
```

```
{
  if (ARequestInfo->Document == "/index.html")
  {
    AResponseInfo->ContentText = "";
    AResponseInfo->ContentType = "text/html";
    AResponseInfo->ResponseNo = 200;
  }
  else if (ARequestInfo->Document == "/styles.css")
  {
    AResponseInfo->ContentText = "";
    AResponseInfo->ContentType = "text/css";
    AResponseInfo->ResponseNo = 200;
  }
}
```

Korzystając z narzędzi deweloperskich Chrome, można sprawdzić, w jaki sposób plik styles.css jest przesyłany do klienta.



## TsgcWebSocketHTTPServer | Usługa alternatywna HTTP/2

Nagłówek HTTP **Alt-Svc** służy do **informowania** klientów, że **ten sam zasób jest dostępny za pośrednictwem innej usługi lub protokołu**. Jest to przydatne na przykład do informowania klientów HTTP, że serwer obsługuje HTTP/2.

**Przykład:** jeśli serwer działa pod lokalnym adresem IP 127.0.0.1 i nasłuchuje na 2 portach: 80 (nieszyfrowany) i 443 (szyfrowany). Można poinformować klientów, że HTTP/2 jest obsługiwane na porcie 443, używając następującego nagłówka HTTP

```
Alt-Svc: h2=":443"
```

Gdy protokół HTTP/2 jest włączony, ten nagłówek jest automatycznie dodawany, jeśli połączenie nie korzysta z protokołu HTTP/2.

Tę funkcję można włączyć lub wyłączyć za pomocą właściwości **HTTP2Options.AltSvc**.

# TsgcWebSocketHTTPServer | Wątki serwera HTTP/2

Poniżej przedstawiono różnice między HTTP 1.1 a HTTP 2.0:

## HTTP 1.1

W tradycyjnym zachowaniu HTTP przy wysyłaniu wielu żądań przez to samo połączenie klient musi czekać na odpowiedź każdego żądania przed wysłaniem następnego. To sekwencyjne podejście znacząco zwiększa czas ładowania zasobów strony. Aby rozwiązać ten problem, HTTP/1.1 wprowadził funkcję zwaną potokowaniem (pipelining), pozwalającą klientowi wysyłać wiele żądań bez oczekiwania na odpowiedź serwera. Serwer natomiast odpowiada klientowi w tej samej kolejności, w jakiej otrzymał żądania.

Chociaż pipelining wydawał się rozwiązaniem, napotykał na trudności:

- **Ignorowanie żądań potokowych lub uszkodzanie odpowiedzi przez serwer:** niektóre serwery ignorowały żądania potokowe lub uszkadzały odpowiedzi, co prowadziło do zawodnej komunikacji.
- **Blokowanie nagłówka kolejki (Head-of-Line Blocking):** Pierwsze żądanie w potoku może blokować kolejne żądania, powodując opóźnienie w ich przetwarzaniu. To zjawisko, znane jako blokowanie nagłówka kolejki, skutkowało dłuższym czasem ładowania stron.

W celu optymalizacji ładowania stron z serwerów obsługujących HTTP/1.1 przeglądarki internetowe zastosowały obejście. Otwierają od sześciu do ośmiu równoległych połączeń z serwerem, umożliwiając jednocześnie wysyłanie wielu żądań. Ten równoległy mechanizm ma na celu złagodzenie problemów związanych z potokowaniem i poprawę ogólnego czasu ładowania strony.

Wybór sześciu do ośmiu równoległych połączeń przez przeglądarki internetowe wynika z optymalizacji. Konkretnie powody wyboru tej liczby mogą wynikać z kompromisu między wykorzystaniem zasobów, efektywnością sieci i unikaniem potencjalnych wąskich gardeł.

## HTTP 2.0

W odpowiedzi na ograniczenia napotkane w pipelingu, HTTP/2 wprowadził funkcję zwaną multipleksowaniem. **Multipleksowanie** umożliwia **efektywniejszą komunikację** między klientem a serwerem poprzez umożliwienie **jednoczesnego przesyłania wielu żądań i odpowiedzi przez pojedyncze połączenie**.

HTTP/2 wykorzystuje binarny mechanizm ramkowania, co oznacza, że wiadomości HTTP są dzielone na mniejsze, niezależne jednostki zwane ramkami. Ramki mogą być przeplatane i przesyłane przez połączenie niezależnie od siebie. Po stronie odbiorczej ramki są ponownie składane w celu odtworzenia oryginalnej wiadomości HTTP.

Ten binarny mechanizm ramkowania jest fundamentem multipleksowania w HTTP/2. Umożliwia przeglądanie wysyłanie wielu żądań przez to samo połączenie bez problemów z blokowaniem. W rezultacie przeglądarki takie jak Chrome używają tego samego identyfikatora połączenia dla żądań HTTP/2, co pozwala na efektywną i nieprzerwaną komunikację między klientem a serwerem.

Dzięki funkcji multipleksowania HTTP/2, realizowanej przez mechanizm ramkowania binarnego, efektywność i szybkość wymiany danych między klientami a serwerami jest zwiększona poprzez umożliwienie jednoczesnego przesyłania wielu żądań i odpowiedzi przez pojedyncze połączenie.

## TsgcWebSocketHTTPServer

Aby poprawić wydajność protokołu HTTP/2, żądania są domyślnie rozdzielane w puli wątków (domyślnie 32) za każdym razem, gdy serwer odbiera nowe żądanie HTTP/2. Pozwala to uniknąć oczekiwania, gdy pojedyncze połączenie wysyła wiele współbieżnych żądań wymagających sekwencyjnego przetwarzania w kontekście wątku połączenia bez tej puli.

Zachowanie puli wątków można skonfigurować przy użyciu następujących właściwości.

- **HTTP2Options.PoolOfThreads.Enabled:** (domyślnie false) włącz, aby przekazywać żądania HTTP/2 do puli wątków zamiast wątku połączenia.
- **HTTP2Options.Threads:** (domyślnie 32) liczba wątków używanych do obsługi żądań HTTP/2. Należy ustawić tę wartość odpowiednio do liczby procesorów na serwerze.

Aby **precyzyjnie dostroić żądania**, wybierając, które żądania muszą być przetwarzane w puli wątków (ponieważ są czasochłonne), a które mogą być przetwarzane w wątku połączenia, można użyć zdarzenia **OnHttp2BeforeAsyncRequest**. Zdarzenie to jest wywoływane przed umieszczeniem żądania w kolejce w puli wątków. Należy użyć parametru **Async**, aby określić, czy żądanie jest wielowątkowe.

```
void __fastcall TForm1::OnHTTP2BeforeAsyncRequest(TObject *Sender, TsgcWSConnection *Connection, const TIdHTTPRequestInfo &
{
    if (ARequestInfo.Document == "/fast-request")
        Async = false;
}
```





## TsgcWebSocketHTTPServer | Błąd 404 bez treści odpowiedzi

Domyślnie biblioteka Indy dodaje treść do odpowiedzi HTTP, jeśli nie przypisano `ContentText` ani `ContentStream`. Aby zwrócić pustą treść odpowiedzi (dla błędu 404 lub podobnego), można zastosować następujące podejście.

Należy utworzyć nowy `TStringStream` bez zawartości i przypisać go do właściwości `ContentStream` odpowiedzi HTTP. W ten sposób odpowiedź HTTP zostanie wysłana bez domyślnych znaczników HTML.

### Przykład

```
private void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo,
    TidHTTPResponseInfo *AResponseInfo)
{
    AResponseInfo->ContentStream = new TStringStream("");
    AResponseInfo->ContentType = "text/html";
    AResponseInfo->ResponseNo = 404;
}
```

# TsgcWebSocketHTTPServer | Sesje

HTTP jest protokołem bezstanowym (przynajmniej do wersji HTTP 1.1), więc klient żąda pliku, serwer wysyła odpowiedź i połączenie jest zamykane (można włączyć keep-alive, aby połączenie nie było zamykane natychmiast, ale wykracza to poza zakres tego artykułu). Sesje umożliwiają przechowywanie informacji o kliencie, które mogą być używane na przykład podczas logowania klienta. Można użyć dowolnego unikalnego identyfikatora sesji, przeszukać listę sesji w celu sprawdzenia, czy już istnieje, a jeśli nie, utworzyć nową sesję. Sesja może zostać zniszczona po okresie bezczynności lub ręcznie po wylogowaniu klienta.

## Konfiguracja

W komponencie TsgcWebSocketHTTPServer istnieje kilka właściwości włączających/wyłączających sesje. Najważniejsze z nich to:

|                  |                                                                                                                                                                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SessionState     | To pierwsza właściwość, która musi zostać włączona, aby korzystać z sesji. Bez włączenia tej właściwości będą działać                                                                                                                                                         |
| SessionTimeout   | Należy ustawić wartość większą od zera (w milisekundach) określającą maksymalny czas aktywności sesji                                                                                                                                                                         |
| AutoStartSession | Sesje mogą być tworzone automatycznie (AutoStartSession = true) lub ręcznie (AutoStartSession = false). Jeśli AutoStartSession = true, sesje są tworzone automatycznie, serwer użyje RemoteIP jako unikalnego identyfikatora do sprawdzenia, czy istnieje już sesja zapisana. |

```
TsgcWebSocketHTTPServer1->SessionState = true;  
TsgcWebSocketHTTPServer1->SessionTimeout = 600000;  
AutoStartSession = false;
```

## Utwórz sesję

Aby utworzyć nową sesję, należy wygenerować **unikalny identyfikator sesji**. Można użyć dowolnej wartości. **Przykład:** jeśli klient się uwierzytelnia, można jako identyfikator sesji użyć kombinacji nazwy użytkownika, hasła i zdalnego adresu IP.

Następnie przeszukaj listę sesji, aby sprawdzić, czy sesja już istnieje. Jeśli nie istnieje, utwórz nową.

Gdy tworzona jest nowa sesja, wywoływane jest zdarzenie **OnSessionStart**, a gdy sesja jest zamykana, wywoływane jest zdarzenie **OnSessionEnd**.

```

void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo,
TidHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/")
    {
        AResponseInfo->ServeFile(AContext, "yourpathhere\\index.html");
    }
    else
    {
        // check if user is valid
        if (((ARequestInfo->AuthUsername == "user") && (ARequestInfo->AuthPassword == "pass")) == false)
        {
            AResponseInfo->AuthRealm = "Authenticate";
        }
        else
        {
            // create a new session id with authentication data
            string vID = ARequestInfo->AuthUsername + "_" + ARequestInfo->AuthPassword + "_" + ARequestInfo->RemoteIP;

            // search session
            TidHTTPSession oSession = TsgcWebSocketHTTPServer1->SessionList->GetSession(vID, ARequestInfo->RemoteIP);

            // create new session if not exists
            if (oSession == NULL)
            {
                oSession = TsgcWebSocketHTTPServer1->SessionList->CreateSession(ARequestInfo->RemoteIP, vID);
            }

            AResponseInfo->ContentText = "<html><head></head><body>Authenticated</body></html>";
            AResponseInfo->ResponseNo = 200;
        }
    }
}

```

# TsgcWebSocketHTTPServer | Strumieniowanie wideo

Jeśli chcesz strumieniować plik wideo za pomocą serwera, możesz użyć funkcji `IndyStreamFileVideo` do strumieniowania pliku przy użyciu kodowania przesyłania fragmentarycznego.

Funkcja przyjmuje następujące parametry:

- **AContext:** pochodzi ze zdarzenia `OnCommandGet`.
- **AResponseInfo:** pochodzi ze zdarzenia `OnCommandGet`.
- **aFileName:** pełna nazwa pliku wideo do strumieniowania.
- **ContentType:** domyślnie „video/mpeg”.
- **aBufferSize:** domyślnie wynosi 1024 bajty.

**Przykład:** strumieniowanie wideo, gdy użytkownik przejdzie do dokumentu `/video.mp4`, a film ten znajduje się w folderze `c:\videos\video.mp4`

```
void OnServerCommandGet(TIdContext *AContext, TIdHTTPRequestInfo *ARequestInfo, TIdHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/video.mp4") {
        IndyStreamFileVideo(AContext, AResponseInfo, "C:\\videos\\video.mp4");
    } else {
        AResponseInfo->ResponseNo = 404;
    }
}
```

# Serwer SSL | SChannel dla serwerów Indy

Komponenty serwerowe oparte na Indy (**TsgcWebSocketServer**, **TsgcWebSocketHTTPServer**) mogą używać **Windows SChannel** (Secure Channel) jako dostawcy TLS zamiast OpenSSL. SChannel jest natywną implementacją TLS systemu Windows, dlatego nie wymaga zewnętrznych bibliotek DLL.

## Jak to działa

Gdy komponent serwera ma włączony SSL i IOHandler jest ustawiony na **iohSChannel**, serwer tworzy instancję **TsgcIdServerIOHandlerSSLSChannel**, która obsługuje wszystkie operacje TLS za pomocą Windows SChannel API. Dla każdego przychodzącego połączenia klienta serwer wykonuje uzgadnianie TLS przez dostawcę SChannel, negocjując wersję protokołu, zestaw szyfrów i wiążąc skonfigurowany certyfikat.

SChannel odczytuje certyfikaty z **magazynu certyfikatów systemu Windows** lub z **pliku PFX** (.pfx / .p12). Pliki PEM nie są wymagane i nie trzeba wdrażać bibliotek OpenSSL.

## Konfiguracja

Aby włączyć SChannel na serwerze Indy, skonfiguruj następujące właściwości:

1. Ustaw właściwość **SSL** na **True**.
2. Ustaw **SSLOptions.IOHandler** na **iohSChannel**.
3. Ustaw właściwość **SSLOptions.Version** na żadaną wersję TLS (tls1\_2, tls1\_3, ...).
4. Ustaw wartość **SSLOptions.Port** na port używany do bezpiecznych połączeń.
5. Skonfiguruj certyfikat przy użyciu jednej z dwóch opisanych poniżej metod.

## Właściwości SChannel\_Options

Właściwość podrzędna **SSLOptions.SChannel\_Options** udostępnia ustawienia specyficzne dla SChannel:

|                             |                                                                                                                                                                                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CertHash</b>             | Odcisk palca (skrót w formacie szesnastkowym) certyfikatu zainstalowanego w magazynie certyfikatów systemu Windows.                                                                                           |
| <b>CertStoreName</b>        | Magazyn certyfikatów do przeszukania: <b>scsnMY</b> (Osobisty), <b>scsnRoot</b> , <b>scsnTrust</b> , <b>scsnCA</b> .                                                                                          |
| <b>CertStorePath</b>        | Lokalizacja magazynu: <b>scspStoreLocalMachine</b> lub <b>scspStoreCurrentUser</b> .                                                                                                                          |
| <b>CipherList</b>           | Opcjonalna lista algorytmów szyfrowania oddzielona dwukropkami (np. <b>CALG_AES_256:CALG_AES_128</b> ). Pusta wartość oznacza użycie domyślnych ustawień systemowych.                                         |
| <b>UseLegacyCredentials</b> | Gdy wartość wynosi True, zamiast <b>SCH_CREDENTIALS</b> używana jest starsza struktura <b>SCHANNEL_CRED</b> . Należy włączyć tę opcję w starszych wersjach systemu Windows, które nie obsługują nowszego API. |

Ponadto ogólne właściwości **SSLOptions CertFile** i **Password** są używane podczas ładowania certyfikatu z pliku PFX.

## Certyfikat z magazynu systemu Windows

Jeśli certyfikat jest już zainstalowany w magazynie certyfikatów Windows, należy podać **odcisk palca** certyfikatu i wskazać jego lokalizację.

Aby znaleźć odcisk palca, otwórz **PowerShell** i uruchom:

```
dir cert:\localmachine\my
```

Kolumna **Thumbprint** pokazuje szesnastkowy skrót, który jest potrzebny.

```
Directory: Microsoft.PowerShell.Security\Certificate::localmachine\my
Thumbprint                               Subject
-----
C12A8FC8AE668F866B48F23E753C93D357E9BE10  CN=*.mydomain.com
```

```
oServer = new TsgcWebSocketHTTPServer();
oServer->SSL = true;
oServer->SSLOptions->IOHandler = iohSChannel;
oServer->SSLOptions->Version = tls1_2;
oServer->SSLOptions->Port = 443;
oServer->Port = 443;
oServer->SSLOptions->SChannel_Options->CertHash = "C12A8FC8AE668F866B48F23E753C93D357E9BE10";
oServer->SSLOptions->SChannel_Options->CertStoreName = scsnMY;
oServer->SSLOptions->SChannel_Options->CertStorePath = scspStoreLocalMachine;
oServer->Active = true;
```

## Certyfikat z pliku PFX

Jeśli posiadasz plik certyfikatu PFX (.pfx lub .p12), ustaw właściwości **CertFile** i **Password** w **SSLOptions**. **SChannel** zaimportuje certyfikat podczas uruchamiania.

```
oServer = new TsgcWebSocketHTTPServer();
oServer->SSL = true;
oServer->SSLOptions->IOHandler = iohSChannel;
oServer->SSLOptions->Version = tls1_2;
oServer->SSLOptions->Port = 443;
oServer->Port = 443;
oServer->SSLOptions->CertFile = "c:\\certificates\\server.pfx";
oServer->SSLOptions->Password = "mypassword";
oServer->Active = true;
```

Jeśli posiadasz certyfikat PEM i klucz prywatny, należy je najpierw przekonwertować do formatu PFX przy użyciu OpenSSL:

```
openssl pkcs12 -inkey server.key -in server.crt -export -out server.pfx
```

## Wersja TLS

Użyj właściwości **SSLOptions.Version** do kontrolowania, które wersje TLS serwer akceptuje:

|        |                       |
|--------|-----------------------|
| tls1_0 | TLS 1.0 (niezalecane) |
|--------|-----------------------|

|              |                             |
|--------------|-----------------------------|
| tls1_1       | TLS 1.1                     |
| tls1_2       | TLS 1.2 (zalecane)          |
| tls1_3       | TLS 1.3                     |
| tlsUndefined | Akceptuj TLS 1.0, 1.1 i 1.2 |

## Lista szyfrów

Domyślnie SChannel używa systemowej konfiguracji szyfrowania. Można ograniczyć dozwolone szyfry, ustawiając **SChannel\_Options.CipherList** na listę nazw algorytmów oddzielonych dwukropkami, na przykład:

```
CALG_AES_256:CALG_AES_128
```

Pozostaw tę właściwość pustą, aby używać wartości domyślnych systemu Windows.

## Starsze poświadczenia

Windows Server 2019 i wcześniejsze wersje mogą nie obsługiwać nowszego interfejsu API **SCH\_CREDENTIALS**. Jeśli uruchomienie serwera na starszej wersji systemu Windows nie powiedzie się, należy ustawić **SChannel\_Options.UseLegacyCredentials** na **True**, aby użyć starszej struktury **SCHANNEL\_CRED**.

Komponent automatycznie wykrywa wersję systemu Windows w większości przypadków, ale w razie potrzeby można wymusić tryb legacy.

## Zalety w porównaniu z OpenSSL

- **Brak zewnętrznych bibliotek DLL:** SChannel jest wbudowany w system Windows, dlatego nie ma potrzeby wdrażania plików libeay32.dll / ssleay32.dll ani libcrypto / libssl.
- **Integracja z magazynem certyfikatów systemu Windows:** korzystanie z certyfikatów już zainstalowanych i zarządzanych przez system operacyjny.
- **Automatyczne aktualizacje:** Ulepszenia TLS i poprawki zabezpieczeń są stosowane za pośrednictwem usługi Windows Update.

## Uwagi

- SChannel jest dostępny wyłącznie w systemie **Windows**. W przypadku serwerów wieloplatformowych należy używać OpenSSL (iohOpenSSL).
- Serwer musi posiadać **klucz prywatny** powiązany z certyfikatem. W przypadku korzystania z metody magazynu Windows certyfikat musi zostać zaimportowany wraz z kluczem prywatnym.
- W przypadku serwerów produkcyjnych korzystających z metody magazynu certyfikatów zalecane jest użycie magazynu **Komputer lokalny** (scspStoreLocalMachine), dzięki czemu certyfikat jest dostępny niezależnie od konta użytkownika uruchamiającego usługę.
- Obsługiwane są wyłącznie pliki certyfikatów **PFX** (.pfx / .p12). Jeśli posiadane są pliki PEM, należy najpierw przekonwertować je do formatu PFX.



# TsgcWebSocketServer\_HTTPAPI

HTTP Server API umożliwia aplikacjom komunikację przez HTTP bez korzystania z Microsoft Internet Information Server (IIS).

## Wprowadzenie

HTTP Server API umożliwia aplikacjom komunikację przez HTTP bez korzystania z Microsoft Internet Information Server (IIS). Aplikacje mogą rejestrować się w celu odbierania żądań HTTP dla określonych URL, odbierać żądania WebSocket i wysyłać odpowiedzi WebSocket. HTTP Server API obsługuje SSL, dzięki czemu aplikacje mogą wymieniać dane przez bezpieczne połączenia HTTP bez IIS. Jest również zaprojektowany do współpracy z portami uzupełniania operacji wejścia/wyjścia (I/O completion ports).

Serwer obsługuje następujące protokoły:

- WebSockets (wymaga systemu Windows 8 lub nowszego)
- HTTP 1.1
- HTTP/2 (wymaga Windows 2016+ lub Windows 10+).

**Domyślnie ten komponent wymaga uruchomienia aplikacji w trybie administratora w celu rejestracji adresu URL. Jeśli adres URL został już zarejestrowany za pomocą zewnętrznego narzędzia, takiego jak netsh, aplikacja może działać bez uprawnień administratora. Należy wówczas wyłączyć właściwość `BindingOptions.ConfigureSSLCertificate`, co umożliwi uruchomienie aplikacji bez praw administratora.**

**Ustaw FastMM4/FastMM5 jako pierwszą jednostkę projektu.**

Należy wykonać poniższe kroki w celu skonfigurowania tego komponentu:

1. Upuść komponent TsgcWebSocketServer\_HTTPAPI na formularzu
2. Zdefiniuj adres nasłuchiwania i port:

```
Server->Host = "127.0.0.1";  
Server->Port = 80;
```

3. Należy ustawić dozwolone specyfikacje; domyślnie wszystkie specyfikacje są dozwolone.

**RFC6455:** jest standardową i zalecaną specyfikacją WebSocket.

**Hixie76:** jest to wersja robocza i zaleca się nawiązywanie połączeń Hixie76 wyłącznie w celu zapewnienia obsługi starych przeglądarek, takich jak Safari 4.2

5. Utwórz procedurę i ustaw właściwość `Active = true`

## URL Reservation

Serwer HTTP.SYS używa rezerwacji adresów URL do przypisywania punktów końcowych URL wykonywanych przez serwer HTTP.SYS.

### Podstawowa rezerwacja URL

Jest to najprostszy tryb konfiguracji serwera: wystarczy ustawić `Host` i `Port`, które będzie obsługiwać serwer HTTP.SYS.

Przykład: jeśli serwer działa pod adresem IP 127.0.0.1 i portem 80, wystarczy ustawić następujące właściwości

```
Server->Host = "127.0.0.1";  
Server->Port = 80;
```

Jeśli serwer działa na wielu adresach IP i wymagane jest powiązanie z kilkoma z nich, należy użyć metody **NewBinding**. Należy najpierw wyczyścić właściwości `Host` i `Bindings`, a następnie użyć metody `NewBinding`, aby zdefiniować wszystkie powiązania serwera.

```
Server->Host = "";  
Server->Bindings->Clear;  
Server->Bindings->NewBinding("127.0.0.1", 80, "");  
Server->Bindings->NewBinding("80.50.55.11", 80, "");
```

Jeśli serwer wymaga połączeń SSL, należy wykonać następujące czynności, aby zdefiniować `host` i `port` obsługujące połączenia SSL.

```
Server->Host = "127.0.0.1";  
Server->Port = 443;  
Server->SSL = true;  
Server->SSLOptions->Hash = "CERTIFICATE_HASH";
```

Jeśli serwer wymaga połączeń SSL z wieloma adresami IP, należy najpierw wyczyścić właściwości `Host` i `Bindings`, a następnie zarejestrować nowe powiązania.

```
Server->Host = '';  
Server->Bindings->Clear;  
Server->Bindings->NewBinding("127.0.0.1", 443, "", true, "CERTIFICATE_HASH1");  
Server->Bindings->NewBinding("80.50.55.11", 443, "", true, "CERTIFICATE_HASH2");
```

## Dokumentacja

- Właściwości — 27 dostępnych właściwości komponentu.
- Metody — 9 metod publicznych.
- Zdarzenia — 23 zdarzenia do obsłużenia.

## Przewodniki

- **Konfiguracja**
  - [Rezerwacja URL](#)
- **Połączenie**
  - [OnDisconnect nie jest wywoływane](#)
- **SSL**
  - [HTTPAPI Server SSL](#)
  - [Certyfikaty z podpisem własnym](#)
- **HTTP**
  - [Niestandardowe nagłówki](#)
  - [Wyślij odpowiedź tekstową](#)
  - [Wyślij odpowiedź plikową](#)
  - [Publikuj duże pliki](#)
- **HTTP/2**
  - [Wyłącz HTTP/2](#)

# HTTPAPI | Rezerwacja URL

Rezerwacja adresów URL HTTP.SYS to funkcja systemu operacyjnego Windows umożliwiająca użytkownikowi zarezerwowanie określonego adresu URL (Uniform Resource Locator) dla swojej aplikacji lub usługi. Gdy adres URL jest zarezerwowany za pomocą HTTP.SYS, system operacyjny przechwytuje wszystkie przychodzące żądania HTTP dla tego adresu URL i kieruje je do wskazanej aplikacji lub usługi.

Aby zarezerwować adres URL przy użyciu HTTP.SYS, aplikacja lub usługa musi najpierw zarejestrować adres URL w sterowniku HTTP.SYS poprzez wywołanie HTTP API. Aplikacja lub usługa podaje adres URL, metodę HTTP (np. GET, POST) oraz wszelkie dodatkowe ustawienia, takie jak wymagania dotyczące uwierzytelniania.

Po zarejestrowaniu adresu URL, HTTP.SYS przechwytuje wszystkie przychodzące żądania HTTP dla tego adresu URL i wyszukuje zarejestrowaną aplikację lub usługę na podstawie adresu URL i metody. Jeśli zostanie znaleziona pasująca aplikacja lub usługa, sterownik HTTP.SYS przekazuje żądanie do tej aplikacji lub usługi w celu przetworzenia.

## Polecenia NETSH

### Zarejestruj adres URL

W tym przykładzie adres URL `http://example.com:80/` jest rejestrowany dla użytkownika `DOMAIN\user`. Można zastąpić go żądanym adresem URL i użytkownikiem.

```
netsh http add urlacl url=http://example.com:80/ user=DOMAIN\user
```

### Usuń adres URL

W tym przykładzie usuwany jest URL `http://example.com:80/`. Można zastąpić go dowolnym adresem URL, który ma zostać usunięty.

```
netsh http delete urlacl url=http://example.com:80/
```

### Pokaż wszystkie URL-e

To polecenie wyświetla listę wszystkich zarejestrowanych rezerwacji URL w systemie.

```
netsh http show urlacl
```

## TsgcWebSocketServer\_HTTPAPI

Serwer HTTP.SYS automatycznie rejestruje adresy URL podczas uruchamiania. Odbywa się to przy użyciu następujących parametrów i metod.

- **Host i Port:** jeśli Host nie jest pusty, a Port różni się od zera, serwer spróbuje zarejestrować adres URL. Przykład: adres URL `https://127.0.0.1:5000` zostanie zarejestrowany przy użyciu następujących właściwości
  - Host = '127.0.0.1';
  - Port = 5000
  - SSL = True
- **NewBinding:** użyj tej metody do rejestracji jednego lub wielu adresów URL.
  - Rejestracja adresu URL `https://127.0.0.1:5000` --> `NewBinding('127.0.0.1', 5000, '/', True)`
  - Rejestracja adresu URL `http://+:5000/ws/` --> `NewBinding('+', 5000, '/ws/')`

Rejestracja URL wymaga uprawnień administratora w następujących przypadkach:

- Numer portu jest poniżej 1024
- Host jest znakiem wieloznacznym "+", a nie adresem IP.

Jeśli chcesz zarejestrować port 443 dla wszystkich adresów IP serwera i nasłuchiwać tylko na punkcie końcowym `/ws/`, bez uruchamiania serwera z uprawnieniami administratora, wykonaj następujące kroki:

- Rejestracja URL za pomocą netsh
  - `netsh http add urlacl url=https://+:443/ws/ user=DOMAIN\user`
- Skonfiguruj serwer z następującym powiązaniem
  - `NewBinding('+', 443, '/ws/', True);`
- Wyłącz właściwość `ConfigureSSLCertificate`
  - `TsgcWebSocketServer_HTTPAPI.BindingOptions.ConfigureSSLCertificate = false;`
- Skonfiguruj certyfikat SSL
  - [HTTPAPI Server SSL](#)

# HTTPAPI | FineTune

**TsgcWSServer\_HTTPAPI** udostępnia opublikowaną właściwość **FineTune** typu **TsgcServerHTTPAPI\_FineTune**. Grupuje niskopoziomowe przełączniki trybu jądra wpływające na sposób, w jaki Windows HTTP Server API (**http.sys**) kolejkuje, wysyła i uzupełnia żądania.

## QueueLength

Opakowuje ustawienie jądra **HttpServerQueueLengthProperty**. Kontroluje maksymalną liczbę oczekujących żądań, które **http.sys** przechowuje w kolejce trybu jądra, zanim serwer je odczyta. Gdy kolejka jest pełna, jądro odpowiada na nowe połączenia kodem **503 Service Unavailable** bezpośrednio, bez sięgania do trybu użytkownika.

**Typ:** ULONG.

**Domyślnie:** 1000 (wartość domyślna jądra Windows).

**Zakres:** do 65535 żądań.

**Co to poprawia:** zwiększenie tej wartości zapobiega odrzucaniu przez jądro systemu legalnego ruchu przy skokami obciążeniach — dotyczy to przykładowo flot IoT nawiązujących ponowne połączenia po zaniku sieci, klientów rozgłaszania danych rynkowych po resecie dostawcy lub harmonogramowanych pracowników zadań, którzy wszyscy łączą się przy wspólnym ticku **crona**.

## SkipIOCPOnSuccess

Gdy wartość jest ustawiona na **True**, serwer wywołuje **SetFileCompletionNotificationModes** na uchwycie kolejki żądań z flagami **FILE\_SKIP\_COMPLETION\_PORT\_ON\_SUCCESS** i **FILE\_SKIP\_SET\_EVENT\_ON\_HANDLE**. Nakładkowe operacje **we/wy** na kolejce żądań, które kończą się synchronicznie, nie publikują już dodatkowego pakietu uzupełnienia do portu uzupełnienia **we/wy**.

**Typ:** Boolean.

**Domyślnie:** **False** (żadna flaga nie jest ustawiona, co zachowuje bieżące zachowanie).

**Co poprawia:** eliminuje przełączenie z trybu jądra do trybu użytkownika na krytycznej ścieżce żądania, gdy wywołanie synchronicznie zwraca **NO\_ERROR** — **worker** przetwarza zakończenie bezpośrednio na wątku wywołującym, zamiast oczekiwać na pakiet **IOCP**. Jest to wzorzec stosowany w referencyjnej próbkce **Microsoft HTTP Server High Performance**. Flaga jest przeznaczona do użycia łącznie z **OperatingMode = ompHighPerf**.

API jest rozwiązywane w czasie wykonywania za pomocą **GetProcAddress** z **kernel32**. W systemie Windows XP i wcześniejszych (cel wykonawczy Delphi 7) punkt wejścia nie istnieje — opakowanie zwraca **False**, a serwer działa bez ustawionej flagi.

## OperatingMode

Wybiera jedną z dwóch architektur akceptacji i wysyłki.

**Type:** `TsgcHTTPAPIOperatingMode = (ompClassic, ompHighPerf)`.

**Domyślnie:** `ompClassic`.

- **ompClassic** — pojedynczy wątek akceptujący wywołuje `HttpReceiveHttpRequest` synchronicznie i ręcznie przekazuje każde żądanie do puli wątków roboczych IOCP za pomocą `PostQueuedCompletionStatus`. Jest to historyczne zachowanie.
- **ompHighPerf** — implementuje wzorec wysokiej wydajności serwera HTTP firmy Microsoft. Wątek akceptujący jest usunięty. Podczas uruchamiania serwer z wyprzedzeniem wystawia `ThreadPoolSize × HighPerfAcceptsPerWorker` asynchronicznych odbiorów w kolejce IOCP (domyślnie  $32 \times 4 = 128$  jednoczesnych oczekujących odbiorów). Każdy wątek roboczy obsługuje ukończenie operacji i wystawia kolejny asynchroniczny odbiór, utrzymując przesuwne okno do momentu zamknięcia serwera.

**Co poprawia:** Tryb `ompHighPerf` opłaca się, gdy serwer obsługuje głębokie potoki dla pojedynczego strumienia (duże przesyłanie/pobieranie ramek) lub wielu równoczesnych klientów (setki do tysięcy). Wstępnie zarejestrowane okno odbierania pochłania skoki ruchu bez alokacji na połączenie, a wbudowane rozsyłanie eliminuje wąskie gardło przekazywania przez akceptor. Dla lekkich API i środowisk programistycznych należy pozostawić domyślny tryb `ompClassic`: przy małym obciążeniu koszty utrzymania 128 wstępnie zarejestrowanych kontekstów przewyższają korzyści. Tryb można zmienić tylko przed aktywacją serwera.

## HighPerfAcceptsPerWorker

Określa, ile asynchronicznych odbiorów każdy wątek roboczy IOCP wstępnie rejestruje, gdy `OperatingMode = ompHighPerf`. Wartość jest ignorowana w trybie `ompClassic`. Łączna liczba jednoczesnych oczekujących odbiorów utrzymywana przez serwer jest równa `ThreadPoolSize × HighPerfAcceptsPerWorker`.

**Type:** Integer.

**Domyślnie:** 4.

**Co to poprawia:** głębsze okno per-worker pozwala serwerowi absorbować większe skoki przychodzących żądań bez alokowania nowych kontekstów na gorącej ścieżce. Należy je zwiększyć w przypadku wdrożeń o wysokiej współbieżności (floty IoT, dystrybucja danych rynkowych, brokery fan-out); kompromisem jest pamięć — każde wstępnie wystawione odbieranie przechowuje bufor żądań (~16 KB) zarezerwowany do momentu jego zakończenia. Domyślna wartość 4 to konserwatywny punkt środkowy zwalidowany względem przykładu MSDN HP.

## Przykład

Poniższy fragment konfiguruje serwer HTTP.sys dla backendu IoT o wysokiej współbieżności: duża kolejka jądra do absorbowania burz ponownych połączeń, tryb dyspozycji HighPerf z rozszerzonym wstępnie wysłanym oknem odbioru oraz włączone zakończenie inline-completion.

```
#include "sgcWebSocket_Server_HTTPAPI.hpp"
#include "sgcWebSocket_HTTPAPI_Server.hpp" // TsgcHTTPAPIOperatingMode
TsgcWSServer_HTTPAPI *oServer = new TsgcWSServer_HTTPAPI(NULL);
oServer->Host = "0.0.0.0";
oServer->Port = 8080;
// absorb 10,000-device reconnect bursts before kernel-level 503
oServer->FineTune->QueueLength = 10000;
// switch from single-acceptor to pre-posted IOCP workers
oServer->FineTune->OperatingMode = ompHighPerf;
// widen the per-worker pre-posted receive window (32 threads * 8 = 256)
oServer->FineTune->HighPerfAcceptsPerWorker = 8;
// dispatch inline on sync-success completions; skip the IOCP round-trip
oServer->FineTune->SkipIOCPOnSuccess = true;
oServer->Active = true;
```

W przypadku typowego wewnętrznego API lub API o małym ruchu należy pozostawić wszystkie właściwości FineTune przy wartościach domyślnych:

```
TsgcWSServer_HTTPAPI *oServer = new TsgcWSServer_HTTPAPI(NULL);
oServer->Host = "localhost";
oServer->Port = 8080;
// FineTune defaults: QueueLength=1000, SkipIOCPOnSuccess=False,
//                    OperatingMode=ompClassic, HighPerfAcceptsPerWorker=4
oServer->Active = true;
```



# TsgcWebSocketServer\_HTTPAPI | Serwer HTTPAPI SSL

Serwer można skonfigurować do używania **certyfikatów SSL**. Aby uzyskać serwer produkcyjny z certyfikatem serwera, należy **zakupić** certyfikat od **renomowanego dostawcy**: Namecheap, GoDaddy, Thawte itp. Do **celów testowych** można użyć **certyfikatu z własnym podpisem** (patrz przykład Demos/Chat, który używa certyfikatu z własnym podpisem). Przeczytaj następujący artykuł: [Jak utworzyć certyfikat z własnym podpisem](#).

Po uzyskaniu certyfikatu należy skonfigurować serwer, aby określić, który certyfikat ma być używany do szyfrowania połączeń.

## Skrót certyfikatu

Najpierw należy poznać skrót certyfikatu. Znalezienie skrótu certyfikatu jest tak proste w **powershell**, jak uruchomienie polecenia **dir** na kontenerze certyfikatów.

```
dir cert:\localmachine\my
```

Skrót jest szesnastkową wartością **Thumbprint**.

```
Directory: Microsoft.PowerShell.Security\Certificate::localmachine\my
Thumbprint                               Subject
-----
C12A8FC8AE668F866B48F23E753C93D357E9BE10  CN=*.mydomain.com
```

Po uzyskaniu wartości Thumbprint należy ustawić ją we właściwości **TsgcWebSocketServer\_HTTPAPI.TLSOptions.Hash**.

Po ustawieniu skrótu wystarczy ustawić **TsgcWebSocketServer\_HTTPAPI.SSL = true**, a serwer jest gotowy do uruchomienia.

Aby ręcznie zarejestrować certyfikat za pomocą netsh, należy użyć następującego polecenia:

```
netsh http add sslcert ipport=<IP>:<PORT> certhash=<THUMBPRINT>appid="{<GUID>}"
```

<IP>: Określa lokalny adres IP powiązania. Nie należy używać powiązania z symbolem wieloznacznym. Należy użyć prawidłowego adresu IP.

<PORT>: Określa port powiązania.

<THUMBPRINT>: Odcisk palca certyfikatu X.509.

<GUID>: Wygenerowany przez programistę identyfikator GUID reprezentujący aplikację w celach informacyjnych.

# TsgcWebSocketServer\_HTTPAPI | Certyfikaty z podpisem własnym

Jeśli certyfikat jest potrzebny do własnych testów, można utworzyć certyfikat z podpisem własnym na komputerze testowym. Należy wykonać następujące kroki:

1. Uruchom **Powershell** jako **Administrator**.
2. Uruchom następujące polecenie, aby utworzyć certyfikat:

```
New-SelfSignedCertificate -DnsName localhost -CertStoreLocation "cert:\LocalMachine\My"
```

W przypadku powodzenia zostanie wyświetlone potwierdzenie dotyczące nowego certyfikatu. Wystarczy skopiować wartość Thumbprint i wkleić ją do właściwości

**TsgcWebSocketServer\_HTTPAPI.TLSOptions.Hash.**



3. **Opcjonalnie** można dodać certyfikat z własnym podpisem jako **zaufany urząd certyfikacji**

```
Run MMC -32 as administrator
```

- 3.1. Wybierz **File / Add or Remove Snap-in**

- 3.2. Wybierz Certyfikat, a następnie kliknij Dodaj

- 3.3. Wybierz konto komputera i naciśnij Dalej.

- 3.4. Wybierz Komputer lokalny i kliknij przycisk OK. Teraz zostaną wyświetlone certyfikaty.

- 4.5. Wybierz certyfikat z sekcji **Personal / Certificates** i wklej go do sekcji **Trusted Root Certificates Authorities / Certificates**.

# TsgcWebSocketServer\_HTTPAPI | Disable HTTP/2

Protokół HTTP/2 jest domyślnie włączony w systemach **Server 2016+** i **Windows 10+**. W niektórych starszych przeglądarkach lub klientach HTTP może wystąpić błąd, ponieważ protokół nie jest w pełni obsługiwany. Błędowi tym można zapobiec, wyłączając protokół HTTP/2.

## Jak wyłączyć HTTP/2

- Otwórz Edytor rejestru systemu Windows
- Przejdź do następującego klucza rejestru:

HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\HTTP\Parameters

- Dodaj następujące wartości DWORD i ustaw obie wartości na zero.
  - EnableHttp2Tls
  - EnableHttp2Cleartext
- Uruchom ponownie komputer.

# TsgcWebSocketServer\_HTTPAPI | Niestandardowe nagłówki

Odpowiedź serwera HTTP.SYS można dostosować przy użyciu właściwości **CustomHeaders** obiektu odpowiedzi.

Ustaw wartość CustomHeaders, podając nazwę nagłówka i wartość nagłówka oddzielone znakami nowej linii.

**Przykład:** jeśli chcesz dodać poniższe nagłówki, zapoznaj się z przykładowym kodem

```
Access-Control-Allow-Origin: *
```

```
Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, PATCH, DELETE
```

```
private void OnHttpRequest(TsgcWSConnection_HTTPAPI *aConnection, const THttpRequest *aRequestInfo,
    ref THttpResponse *aResponseInfo)
{
    aResponseInfo->ResponseNo = 200;
    aResponseInfo->CustomHeaders = "Access-Control-Allow-Origin: *\r\n" + "Access-Control-Allow-Methods: " +
        "GET, POST, OPTIONS, PUT, PATCH, DELETE";
}
```

# TsgcWebSocketServer\_HTTPAPI | Wysyłanie odpowiedzi tekstowej

Użyj zdarzenia **OnHttpRequest** do obsługi żądań HTTP.

Klasa **THttpRequest** zawiera dane żądania HTTP.

- **Document:** dokument, do którego próbuje uzyskać dostęp peer.
- **Method:** metoda HTTP ('GET', 'POST' itp.)
- **Headers:** nagłówki żądania HTTP.
- **AcceptEncoding:** zmienna akceptowanego kodowania, przykład: "gzip, deflate, br".
- **ContentType:** przykład: „text/html”
- **Content:** treść żądania, jeśli istnieje.
- **QueryParams:** parametry zapytania.
- **Cookies:** ciasteczka, jeśli istnieją.
- **ContentLength:** rozmiar treści.
- **AuthExists, AuthUsername, AuthPassword:** dane żądania uwierzytelnienia.
- **Stream:** jeśli żądanie http ma treść, jest to strumień tej treści.

Klasa **THttpResponse** zawiera dane odpowiedzi HTTP.

- **ContentText:** odpowiedź w postaci tekstu.
- **ContentType:** przykład: "text/html". Aby zakodować ContentText w UTF8, należy ustawić charset='utf-8'. Przykład: text/html; charset=utf-8
- **CustomHeaders:** jeśli wymagane jest wysyłanie własnych nagłówków, należy użyć tej zmiennej
- **AuthRealm:** jeśli serwer wymaga uwierzytelniania, należy ustawić tę zmienną.
- **ResponseNo:** numer odpowiedzi HTTP. Przykład: 200 oznacza pomyślną odpowiedź.
- **ContentStream:** jeśli odpowiedź zawiera strumień, należy go tutaj ustawić (nie należy zwalniać strumienia — zostanie zwolniony automatycznie).
- **FileName:** jeśli odpowiedź jest nazwą pliku, należy tutaj podać pełną ścieżkę do pliku.
- **Date, Expires, LastModified:** zmienne datetime odpowiedzi.
- **CacheControl:** umożliwia dostosowanie zachowania pamięci podręcznej.

**Przykład:** jeśli serwer odbierze żądanie GET dla dokumentu "/test.html", należy wysłać odpowiedź OK. W przeciwnym razie należy wysłać odpowiedź 404 dla żądania GET dotyczącego innego dokumentu lub błąd 500 dla żądania z inną metodą.

```
void __fastcall OnHttpRequest(TsgcWSConnection_HTTPAPI *aConnection,
    const THttpRequest *aRequestInfo,
    THttpResponse *aResponseInfo)
{
    if (aRequestInfo->Method == "GET")
```

```
{
  if (aRequestInfo->Document == "/test.html")
  {
    aResponseInfo->ResponseNo = 200;
    aResponseInfo->ContentText = "OK";
    aResponseInfo->ContentType = "text/html; charset=UTF-8";
  }
  else
  {
    aResponseInfo->ResponseNo = 404;
  }
}
else
{
  aResponseInfo->ResponseNo = 500;
}
}
```

# TsgcWebSocketServer\_HTTPAPI | Wysyłanie odpowiedzi z plikiem

Należy użyć właściwości `FileName` obiektu **THttpRequestResponse**, jeśli chcemy wysłać plik jako odpowiedź na żądanie HTTP.

```
void __fastcall OnHTTPRequest(TsgcWSConnection_HTTPAPI *aConnection,
    const THttpRequest *aRequestInfo,
    THttpRequestResponse *aResponseInfo)
{
    if (aRequestInfo->Method == "GET")
    {
        if (aRequestInfo->Document == "/test.zip")
        {
            aResponseInfo->ResponseNo = 200;
            aResponseInfo->FileName = "c:\\download\\test.zip";
            aResponseInfo->ContentType = "application/zip";
        }
        else
        {
            aResponseInfo->ResponseNo = 404;
        }
    }
    else
    {
        aResponseInfo->ResponseNo = 500;
    }
}
```

## Wznawiane pobieranie plików

Odpowiedź HTTP 206 Partial Content jest używana, gdy serwer realizuje żądanie dotyczące określonego fragmentu (zakresu) zasobu, zamiast przesyłać cały plik. Jest to powszechnie stosowane przy wznowianych pobieraniach, strumieniowaniu multimedialnych oraz przesyłaniu dużych plików.

Jak to działa:

**Klient żąda częściowego zasobu:** Klient (przeglądarka, program pobierający lub odtwarzacz multimedialny) wysyła nagłówki `Range` określający żądany zakres bajtów. Przykładowe żądanie:

```
GET /video.mp4 HTTP/1.1
Host: example.com
Range: bytes=1000-5000
```

To żądanie pobiera bajty od 1000 do 5000 pliku `video.mp4`.



**Serwer odpowiada kodem HTTP 206:** Jeśli serwer obsługuje żądania zakresowe, odpowiada kodem 206 Partial Content i dołącza nagłówek Content-Range. Przykładowa odpowiedź:

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 1000-5000/1000000
Content-Length: 4001
Content-Type: video/mp4
```

Nagłówek Content-Range pokazuje:

Obsługiwany zakres (1000–5000)

Całkowity rozmiar pliku (1 000 000 bajtów).

Nagłówek Content-Length zawiera rozmiar zwróconej części (4001 bajtów).

**Klient może żądać kolejnych fragmentów:**

Klient może wysyłać wiele żądań dotyczących różnych części.

Umożliwia to wznawialne pobieranie i wydajne strumieniowanie.

```
void __fastcall OnHTTPRequest(TidContext* AContext, TidHttpRequestInfo* ARequestInfo, TidHttpResponseInfo* AResponseInfo)
{
    std::unique_ptr<TFileStream> oStream(new TFileStream("test.pdf", fmOpenRead | fmShareDenyWrite));
    std::unique_ptr<TIdEntityRangeStrings> oRanges(new TIdEntityRangeStrings(nullptr));
    try {
        oRanges->Text = ARequestInfo->RawHeaders->Values["Range"];
        AResponseInfo->ContentType = "application/pdf";
        if (oRanges->Count > 0) {
            AResponseInfo->ResponseNo = 206; // Partial Content
            AResponseInfo->AcceptRanges = "bytes";
            AResponseInfo->ContentRangeStart = oRanges->Items[0]->StartPos;
            AResponseInfo->ContentRangeEnd = oRanges->Items[0]->EndPos;
            AResponseInfo->ContentRangeInstanceLength = oStream->Size;

            AResponseInfo->ContentStream = new TidHTTPRangeStream(oStream.release(),
                AResponseInfo->ContentRangeStart, AResponseInfo->ContentRangeEnd);
        } else {
            AResponseInfo->ResponseNo = 200; // OK
            AResponseInfo->ContentStream = oStream.release();
        }
    } catch (...) {
        // Handle exceptions
    }
}
```

## TsgcWebSocketServer\_HTTPAPI | Zdarzenie OnDisconnect nie jest wywoływane

Podczas pierwszej pracy z serwerem HTTPAPI bardzo często można zaobserwować, że zdarzenie OnDisconnect nie jest wywoływane natychmiast po zamknięciu połączenia przez klienta. Wynika to z faktu, że serwer HTTPAPI działa nieco inaczej niż inne serwery, takie jak Indy. W **serwerze Indy dla każdego połączenia przydzielony jest osobny wątek**, który co określoną liczbę milisekund sprawdza, czy **połączenie jest aktywne**. Serwer HTTPAPI używa **puli wątków** obsługującej wszystkie połączenia i **nie sprawdza** aktywności każdego połączenia osobno.

Aby otrzymywać powiadomienia o zamknięciu połączenia przez klienta, należy wykonać następującą konfigurację:

1. Jeśli używany jest komponent [TsgcWebSocketClient](#), należy ustawić opcję **Options.CleanDisconnect := True**. Oznacza to, że przed zamknięciem połączenia klient spróbuje wysłać do serwera powiadomienie o zamknięciu połączenia. Jeśli serwer odbierze tę wiadomość, zostanie wywołane zdarzenie OnDisconnect.
2. W przypadku pozostałych rozłączeń jedynym rozwiązaniem jest zapisanie danych do gniazda; w przypadku niepowodzenia połączenie zostanie rozłączone. Należy włączyć funkcję **HeartBeat** na serwerze HTTPAPI, ustawić interwał na przykład 60 sekund i limit czasu na 0. Ta konfiguracja oznacza, że co 60 sekund wszystkie połączenia będą pingowane. Jeśli którekolwiek z nich zostanie rozłączone, zostanie wywołane zdarzenie **OnDisconnect**. Można ustawić mniejszą wartość parametru HeartBeat.Interval, jednak nie należy ustawiać jej zbyt małej (na przykład 1 sekunda to za mało), ponieważ wpłynie to negatywnie na wydajność serwera.

# TsgcWebSocketClient\_WinHTTP

TsgcWebSocketClient\_WinHTTP implementuje kliencki komponent VCL WebSocket, który może łączyć się z serwerem WebSocket.

## Wprowadzenie

TsgcWebSocketClient\_WinHTTP implementuje komponent VCL WebSocket po stronie klienta, który może łączyć się z serwerem WebSocket. Jest oparty na interfejsie API WinHTTP i wymaga systemu Windows 8 lub nowszego. Aby skonfigurować ten komponent, należy wykonać poniższe kroki:

1. Upuść komponent TsgcWebSocketClient\_WinHTTP na formularz
2. Ustaw Host i Port (domyślnie 80) w celu połączenia z dostępnym serwerem WebSocket. Można ustawić właściwość URL, a Host, Port, Parameters... zostaną zaktualizowane na podstawie URL. Przykład: wss://127.0.0.1:8080/ws/ da w rezultacie:  
  

```
oClient = new TsgcWebSocketClient_WinHTTP();  
oClient->Host = "127.0.0.1";  
oClient->Port = 80;  
oClient->TLS = true;  
oClient->Options->Parameters = "/ws/";
```
3. Można wybrać, czy ma być używany TLS (bezpieczne połączenie), czy nie; domyślnie jest wyłączony.
4. Następujące zdarzenia można wykorzystać do dostosowania przepływu klienta websocket:
5. Utwórz procedurę i ustaw właściwość Active na True.

## Dokumentacja

- Właściwości — 16 właściwości dostępnych w komponencie.
- Metody — 8 metod publicznych.
- Zdarzenia — 9 zdarzeń do obsługi.

# TsgcWebSocketFirewall

TsgcWebSocketFirewall implementuje kompleksowy komponent zapory sieciowej chroniący serwery WebSocket przed szerokim zakresem zagrożeń bezpieczeństwa.

## Wprowadzenie

TsgcWebSocketFirewall implementuje kompleksowy komponent firewall, który chroni serwery WebSocket przed szerokim zakresem zagrożeń bezpieczeństwa. Zapewnia szesnaście modułów ochrony, w tym filtrowanie czarnej/białej listy IP, ochronę przed brute force z automatycznym banowaniem IP, wykrywanie wstrzykiwania SQL, wykrywanie XSS (Cross-Site Scripting), ograniczanie szybkości połączeń, ochronę przed floodem wiadomości, filtrowanie według kraju oparte na GeoIP, wykrywanie botów na podstawie adresu IP, dynamiczną ocenę zagrożeń, wykrywanie path traversal, wykrywanie wstrzykiwania poleceń, ograniczanie rozmiaru ładunku, zabezpieczenia specyficzne dla WebSocket (walidacja origin, limity rozmiaru ramki, filtrowanie podprotokołów), progresywną eskalację banów, silnik reguł niestandardowych oraz statystyki w czasie rzeczywistym.

Zapora sieciowa integruje się automatycznie z komponentami serwera za pomocą właściwości **Firewall** dostępnej w TsgcWebSocketServer, TsgcWebSocketHTTPServer i TsgcWebSocketServer\_HTTPAPI. Po przypisaniu i skonfigurowaniu przechwytuje połączenia i wiadomości bez konieczności ręcznego podłączania zdarzeń.

## Szybki start

1. Upuść komponent **TsgcWebSocketFirewall** na formularzu.

2. Skonfiguruj wybrane moduły ochrony:

**Blacklist:** Należy ustawić Enabled na True i dodać adresy IP lub zakresy CIDR do listy IPs (np. "10.0.0.0/8", "192.168.1.100").

**Lista dozwolonych adresów:** Ustaw właściwość Enabled na True i dodaj zaufane adresy IP. Adresy IP na liście dozwolonych pomijają wszystkie pozostałe kontrole.

**BruteForce:** Ustaw Enabled na True. Skonfiguruj MaxAttempts (domyślnie 5), TimeWindowSec (domyślnie 60) i BanDurationSec (domyślnie 300).

**SQLInjection:** ustaw Enabled na True. Wykrywa typowe wzorce iniekcji SQL w wiadomościach.

**XSS:** Ustaw Enabled na True. Wykrywa wzorce cross-site scripting w wiadomościach.

**RateLimit:** Należy ustawić Enabled na True. Ogranicza liczbę równoczesnych połączeń na adres IP (domyślnie 10 na 60 sekund).

**FloodProtection:** Ustaw Enabled na True. Ogranicza liczbę wiadomości na sekundę na IP (domyślnie 100).

**PayloadLimit:** Ustaw Enabled na True. Blokuje wiadomości przekraczające MaxSizeBytes (domyślnie 1 MB).

**PathTraversal:** Ustaw Enabled na True. Wykrywa wzorce przemierzania katalogów.

**CommandInjection:** należy ustawić Enabled na True. Wykrywa wzorce wstrzyknięcia poleceń powłoki.

**ThreatScore:** Ustaw Enabled na True. Przypisuje dynamiczne wyniki adresom IP na podstawie naruszeń.

**BanEscalation:** Ustaw Enabled na True. Eskaluje czas trwania blokad dla wielokrotnych sprawców.

**GeoIP:** Ustaw właściwość Enabled na True. Blokuje lub zezwala na połączenia według kodu kraju.

**BotDetection:** Ustaw Enabled na True. Klasyfikuje łączące się adresy IP jako zweryfikowane roboty indeksujące, hosty centrów danych, podejrzane boty, znajdujące się na liście blokującej lub jako ludzi.

**WebSocketProtection:** należy ustawić właściwość Enabled na True. Weryfikuje źródła, rozmiary ramek i subprotokołu.

**CustomRules:** należy ustawić Enabled na True. Konfigurowalny silnik reguł do złożonego filtrowania.

### 3. Przypisz zaporę do komponentu serwera:

```
sgcWebSocketHTTPServer1.Firewall := sgcWebSocketFirewall1;
```

### 4. Uruchom serwer. Zapora automatycznie chroni wszystkie połączenia i wiadomości.

## Właściwości

| Właściwość          | Opis                                                                                                                                                                                                                                                 |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączone            | Włącza lub wyłącza zaporę sieciową. Domyślnie: True.                                                                                                                                                                                                 |
| Lista zablokowanych | Konfiguracja czarnej listy IP. Zawiera właściwości Enabled (Boolean) oraz IPs (TStringList). Obsługuje dokładne adresy IP oraz notację CIDR (np. „192.168.0.0/16”).                                                                                  |
| Whitelist           | Konfiguracja białej listy IP. Adresy IP na białej liście omijają wszystkie kontrole bezpieczeństwa. Ta sama struktura co Blacklist.                                                                                                                  |
| BruteForce          | Ochrona przed atakami brute force. Właściwości: Enabled, MaxAttempts (domyślnie 5), TimeWindowSec (domyślnie 60), BanDurationSec (domyślnie 300). Automatycznie blokuje adresy IP, które przekroczą limit prób.                                      |
| SQLInjection        | Wykrywanie wstrzyknięć SQL. Właściwości: Enabled, Action (faDeny/faAllow/faLog), CustomPatterns (TStringList dla dodatkowych wzorców).                                                                                                               |
| XSS                 | Wykrywanie ataków XSS. Właściwości: Enabled, Action (faDeny/faAllow/faLog).                                                                                                                                                                          |
| RateLimit           | Ograniczenie szybkości połączeń. Właściwości: Enabled, MaxConnectionsPerIP (domyślnie 10), TimeWindowSec (domyślnie 60).                                                                                                                             |
| FloodProtection     | Ochrona przed zalewem wiadomości. Właściwości: Enabled, MaxMessagesPerSec (domyślnie 100), Action (faDeny/faAllow/faLog).                                                                                                                            |
| PayloadLimit        | Ograniczanie rozmiaru wiadomości. Właściwości: Enabled, MaxSizeBytes (domyślnie 1048576), Action (faDeny/faAllow/faLog).                                                                                                                             |
| PathTraversal       | Wykrywanie przechodzenia przez katalogi. Właściwości: Enabled, Action (faDeny/faAllow/faLog).                                                                                                                                                        |
| CommandInjection    | Wykrywanie wstrzykiwania poleceń. Właściwości: Enabled, Action (faDeny/faAllow/faLog), CustomPatterns (TStringList dla dodatkowych wzorców).                                                                                                         |
| ThreatScore         | Dynamiczne punktowanie zagrożeń. Właściwości: Enabled, AutoBanThreshold (domyślnie 80), DecayPerHour (domyślnie 5) oraz wagi dla poszczególnych typów naruszeń do precyzyjnego dostosowania obliczeń punktacji.                                      |
| BanEscalation       | Progresywna eskalacja blokad. Właściwości: Enabled, Levels (TStringList czasów trwania w sekundach), ResetAfterSec (domyślnie 86400). Każda kolejna blokada stosuje czas trwania z następnego poziomu. Wartość 0 na poziomie oznacza trwałą blokadę. |

|                     |                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GeoIP               | Filtrowanie geograficzne adresów IP. Właściwości: Enabled, Mode (gmBlockList/gmAllowList), Countries (TStringList kodów krajów ISO 3166-1 alfa-2), DatabaseFile (ścieżka do bazy danych GeoIP w formacie CSV).                                                                                                                                                                                   |
| BotDetection        | Wykrywanie botów na podstawie adresu IP. Właściwości: Enabled, KnownBotsFile, CacheDurationSec (domyślnie 3600), DNSTimeoutMS (domyślnie 2000) oraz podobiekty ReverseDNS, Datacenter i DNSBL. Wyłącznie klasyfikujące: klasyfikuje adresy IP jako zweryfikowane roboty indeksujące, hosty centrów danych, podejrzane boty, znajdujące się na liście blokującej lub jako ludzie, bez blokowania. |
| WebSocketProtection | Zabezpieczenia specyficzne dla WebSocket. Właściwości: Enabled, AllowedOrigins (TStringList), MaxFrameSize (Integer), AllowedSubprotocols (TStringList).                                                                                                                                                                                                                                         |
| CustomRules         | Niestandardowy silnik reguł. Właściwości: Enabled, Rules (TCollection elementów TsgcFirewallRuleItem). Każda reguła definiuje warunki i akcje dla złożonej logiki filtrowania.                                                                                                                                                                                                                   |

## Zdarzenia

**OnFiltered:** Wywoływane po zablokowaniu połączenia lub wiadomości. Parametr *Allow* (var Boolean) umożliwia nadpisanie decyzji zapory sieciowej.

**OnViolation:** Wywoływane po wykryciu naruszenia bezpieczeństwa. Dostarcza adres IP, typ naruszenia i szczegóły do celów rejestrowania.

**OnResolveCountry:** Wywoływane, gdy GeoIP musi rozwiązać adres IP na kod kraju. Przypisz procedurę obsługi, aby zapewnić niestandardowe rozwiązywanie krajów, lub załaduj plik bazy danych GeoIP w celu automatycznych wyszukiwań.

**OnBotDetected:** Wywoływane, gdy wykrywanie botów sklasyfikuje adres IP. Dostarcza adres IP, werdykt TsgcBotClassification oraz dopasowaną nazwę bota do celów logowania lub analityki.

**OnResolveBot:** Wywoływane, gdy wykrywanie botów musi sklasyfikować adres IP. Przypisz handler, aby dostarczyć własny werdykt zamiast wbudowanego resolvera.

**OnThreatScoreChanged:** Wywoływane, gdy zmienia się wynik zagrożenia dla adresu IP. Udostępnia adres IP wraz ze starą i nową wartością wyniku.

## Metody publiczne

| Metoda                            | Opis                                                                                                                                                                                                        |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IsConnectionAllowed(IP)           | Sprawdza, czy połączenie z podanego adresu IP jest dozwolone. Wywoływane automatycznie po zintegrowaniu z serwerem.                                                                                         |
| IsMessageAllowed(IP, Message)     | Sprawdza, czy wiadomość od podanego adresu IP przechodzi przez wszystkie filtry wiadomości. Wywoływana automatycznie po integracji z serwerem.                                                              |
| RegisterConnection(IP)            | Rejestruje nowe połączenie do śledzenia limitów szybkości. Wywoływane automatycznie.                                                                                                                        |
| UnregisterConnection(IP)          | Usuwa połączenie ze śledzenia. Wywoływane automatycznie po rozłączeniu.                                                                                                                                     |
| RegisterFailedAttempt(IP)         | Rejestruje nieudaną próbę uwierzytelnienia na potrzeby wykrywania ataków brute force. Należy wywoływać tę metodę z procedury obsługi zdarzenia OnAuthentication w przypadku niepowodzenia uwierzytelnienia. |
| BanIP(IP, DurationSec)            | Ręcznie blokuje adres IP. Wartość DurationSec = 0 oznacza trwały ban.                                                                                                                                       |
| UnbanIP(IP)                       | Usuwa zakaz dla podanego adresu IP.                                                                                                                                                                         |
| IsBanned(IP)                      | Zwraca True, jeśli podany adres IP jest aktualnie zablokowany.                                                                                                                                              |
| ClearBans                         | Usuwa wszystkie aktywne blokady.                                                                                                                                                                            |
| ClearTracking                     | Resetuje wszystkie wewnętrzne dane śledzenia (liczby połączeń, dzienniki prób, liczby wiadomości).                                                                                                          |
| SaveBansToFile(FileName)          | Zapisuje wszystkie aktywne zakazy do pliku w celu zachowania trwałości między restartami.                                                                                                                   |
| LoadBansFromFile(FileName)        | Przywraca blokady z wcześniej zapisanego pliku.                                                                                                                                                             |
| LoadGeoIPDatabase(FileName)       | Ładuje bazę danych GeoIP w formacie CSV. Oczekiwany format: start_ip,end_ip,country_code.                                                                                                                   |
| LookupCountry(IP)                 | Zwraca 2-literowy kod kraju ISO dla adresu IP.                                                                                                                                                              |
| GetBotClassification(IP)          | Zwraca TsgcBotClassification dla adresu IP (najpierw z pamięci podręcznej; brak trafienia może zwrócić bcUnknown i wyzwolić asynchroniczne wyszukiwanie).                                                   |
| GetThreatScore(IP)                | Zwraca bieżący wynik zagrożenia (0–100) dla danego adresu IP.                                                                                                                                               |
| ResetThreatScore(IP)              | Resetuje wynik zagrożenia dla adresu IP do 0.                                                                                                                                                               |
| GetTopThreats(Count)              | Zwraca listę adresów IP z najwyższymi wynikami zagrożenia.                                                                                                                                                  |
| IsOriginAllowed(Origin)           | Sprawdza, czy nagłówek Origin WebSocket jest dozwolony przez konfigurację WebSocketProtection.                                                                                                              |
| IsFrameSizeAllowed(Size)          | Sprawdza, czy rozmiar wiadomości mieści się w limitach zdefiniowanych przez WebSocketProtection.MaxFrameSize.                                                                                               |
| IsSubprotocolAllowed(Subprotocol) | Sprawdza, czy subprotokół WebSocket jest dozwolony przez konfigurację WebSocketProtection.                                                                                                                  |



|            |                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Statystyki | Publiczny obiekt TsgcFirewallStats tylko do odczytu z licznikami czasu rzeczywistego: zablokowane połączenia, odfiltrowane wiadomości, wydane zakazy, wyniki zagrożeń i liczniki trafień dla poszczególnych modułów. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Automatyczna integracja

Po przypisaniu do właściwości Firewall serwera, komponent automatycznie:

- **Przy nowym połączeniu:** sprawdza czarną listę, białą listę, zakazy, limity szybkości, kraj GeoIP i ocenia niestandardowe reguły. Odrzuca połączenie przed zdarzeniem OnTCPConnect, jeśli jest zablokowane.
- **Po odebraniu wiadomości:** Sprawdzane są wzorce SQL injection, XSS, rozmiar ładunku, path traversal, wstrzykiwanie poleceń, limity flood oraz niestandardowe reguły. Aktualizowane są wyniki zagrożeń. Klient zostaje rozłączony w przypadku wykrycia naruszenia.
- **Po rozłączeniu:** Wyrejestrowuje połączenie z wewnętrznego systemu śledzenia.

## Obsługa CIDR

Lista zablokowanych i lista dozwolonych obsługują notację CIDR dla zakresów IP:

- **192.168.1.0/24** — obejmuje adresy od 192.168.1.0 do 192.168.1.255
- **10.0.0.0/8** — pasuje do wszystkich adresów 10.x.x.x
- **172.16.0.0/16** — pasuje do adresów 172.16.x.x

## Przykład

```
// Configure firewall
sgcWebSocketFirewall1->Blacklist->Enabled = true;
sgcWebSocketFirewall1->Blacklist->IPs->Add("10.0.0.0/8");
sgcWebSocketFirewall1->Whitelist->Enabled = true;
sgcWebSocketFirewall1->Whitelist->IPs->Add("192.168.1.1");
sgcWebSocketFirewall1->BruteForce->Enabled = true;
sgcWebSocketFirewall1->BruteForce->MaxAttempts = 3;
sgcWebSocketFirewall1->BruteForce->BanDurationSec = 600;
sgcWebSocketFirewall1->SQLInjection->Enabled = true;
sgcWebSocketFirewall1->XSS->Enabled = true;
sgcWebSocketFirewall1->RateLimit->Enabled = true;
sgcWebSocketFirewall1->RateLimit->MaxConnectionsPerIP = 5;
sgcWebSocketFirewall1->FloodProtection->Enabled = true;

// GeoIP: block connections from specific countries
sgcWebSocketFirewall1->GeoIP->Enabled = true;
```

```

sgcWebSocketFirewall1->GeoIP->Mode = gmBlockList;
sgcWebSocketFirewall1->GeoIP->Countries->Add("CN");
sgcWebSocketFirewall1->GeoIP->Countries->Add("RU");
sgcWebSocketFirewall1->LoadGeoIPDatabase("geoip.csv");

// Threat scoring with auto-ban
sgcWebSocketFirewall1->ThreatScore->Enabled = true;
sgcWebSocketFirewall1->ThreatScore->AutoBanThreshold = 80;

// Progressive ban escalation (5min, 30min, 2hr, 24hr, permanent)
sgcWebSocketFirewall1->BanEscalation->Enabled = true;
sgcWebSocketFirewall1->BanEscalation->Levels->Add("300");
sgcWebSocketFirewall1->BanEscalation->Levels->Add("1800");
sgcWebSocketFirewall1->BanEscalation->Levels->Add("7200");
sgcWebSocketFirewall1->BanEscalation->Levels->Add("86400");
sgcWebSocketFirewall1->BanEscalation->Levels->Add("0");

// Additional content protection
sgcWebSocketFirewall1->PayloadLimit->Enabled = true;
sgcWebSocketFirewall1->PayloadLimit->MaxSizeBytes = 65536;
sgcWebSocketFirewall1->PathTraversal->Enabled = true;
sgcWebSocketFirewall1->CommandInjection->Enabled = true;

// Persistent bans
sgcWebSocketFirewall1->SaveBansToFile("bans.dat");
sgcWebSocketFirewall1->LoadBansFromFile("bans.dat");

// Assign to server
sgcWebSocketHTTPServer1->Firewall = sgcWebSocketFirewall1;

// Optional: Track failed login attempts
void __fastcall TForm1::ServerAuthentication(TsgcWSConnection *Connection,
    String aUser, String aPassword, bool &Authenticated)
{
    Authenticated = (aUser == "admin") && (aPassword == "secret");
    if (!Authenticated)
        sgcWebSocketFirewall1->RegisterFailedAttempt(Connection->IP);
}

// Optional: Handle firewall events for logging
void __fastcall TForm1::FirewallViolation(TObject *Sender, const String aIP,
    const TsgcFirewallViolationType aViolationType, const String aDetails)
{
    Log("Firewall violation from " + aIP + ": " + aDetails);
}

```

## Wykryto wzorce SQL Injection

Wbudowany detektor wstrzyknięć SQL sprawdza następujące wzorce (bez rozróżniania wielkości liter):

- ' OR ', ' AND ' — wstrzyknięcie logiki Boolean
- UNION SELECT — wstrzyknięcie oparte na UNION
- '; DROP, '; DELETE, '; INSERT, '; UPDATE — wstrzyknięcie instrukcji
- -- (komentarze SQL)
- EXEC(, EXECUTE( — Wykonanie polecenia
- xp\_cmdshell — rozszerzone procedury składowane
- CAST(, CONVERT( — nadużycie konwersji typów
- 1=1, 1='1 — wstrzyknięcie tautologiczne

Dodatkowe wzorce niestandardowe można dodawać za pomocą właściwości `SQLInjection.CustomPatterns`.

## Wykryto wzorce XSS

- `<script` — wstrzyknięcie skryptu
- `javascript:` — XSS oparty na protokole
- `onerror=`, `onload=`, `onclick=`, `onmouseover=` — Wstrzykiwanie procedur obsługi zdarzeń
- `eval(` — ewaluacja JavaScript
- `document.cookie` — kradzież ciasteczek
- `<iframe`, `<object`, `<embed` — Wstrzyknięcie elementu
- `<svg onload` — XSS oparty na SVG
- `expression(` — wstrzykiwanie wyrażeń CSS

## Wykryto wzorce path traversal

Wbudowany detektor path traversal sprawdza następujące wzorce:

- `../` i `..\` — Względne przemierzanie katalogów
- `%2e%2e%2f`, `%2e%2e/`, `..%2f` — warianty zakodowane w URL
- `%00` — wstrzyknięcie bajtu null
- `/etc/passwd` — dostęp do poufnego pliku Unix
- `c:\windows` — dostęp do katalogu systemowego Windows

## Wykryto wzorce wstrzyknięcia poleceń

Wbudowany detektor wstrzykiwania poleceń sprawdza następujące wzorce:

- `;` `|` `&&` `||` — Operatory powłoki (łańcuchowanie poleceń)
- ``command`` — Wykonanie z użyciem grawisów
- `$(polecenie)` — Wykonanie podpowłoki
- `rm -rf` — Destrukcyjne operacje na plikach
- `wget`, `curl` — pobieranie plików zdalnych
- `/bin/sh`, `/bin/bash` — wywołanie powłoki Unix
- `cmd.exe` — interpreter poleceń systemu Windows
- `powershell` — wywołanie PowerShell

Dodatkowe wzorce niestandardowe można dodawać za pomocą właściwości `CommandInjection.CustomPatterns`.

## Bezpieczeństwo wątkowe

Komponent zapory sieciowej jest w pełni bezpieczny wątkowo. Wszystkie metody publiczne używają wewnętrznych sekcji krytycznych w celu ochrony współbieżnego dostępu do danych śledzenia. Komponent był testowany obciążeniowo z 20 równoczesnymi wątkami wykonującymi 100 000 operacji bez żadnych błędów i wycieków pamięci.

Komponent może być bezpiecznie współdzielony przez wiele instancji serwera i dostępny z dowolnego wątku (procedury obsługi zdarzeń serwera, wątki timerów itp.) bez zewnętrznej synchronizacji.

## Dokumentacja

- Właściwości — 16 właściwości dostępnych w komponencie.
- Metody — 21 metod publicznych.
- Zdarzenia — 6 zdarzeń, które możesz obsługiwać.

# Zapora: czarna lista i lista dozwolonych

TsgcWebSocketFirewall zapewnia kontrolę dostępu opartą na adresach IP za pomocą dwóch uzupełniających się mechanizmów: czarnej listy blokującej określone adresy IP oraz białej listy udzielającej bezwarunkowego dostępu zaufanym adresom IP.

## Lista zablokowanych

Lista blokowanych adresów IP uniemożliwia nawiązywanie połączeń ze wskazanych adresów lub zakresów adresów IP. Gdy jest włączona, każde przychodzące połączenie z zablokowanego adresu IP jest odrzucane przed dotarciem do zdarzeń połączenia serwera.

| Właściwość        | Opis                                                            |
|-------------------|-----------------------------------------------------------------|
| Blacklist.Enabled | Włącza lub wyłącza sprawdzanie czarnej listy. Domyślnie: False. |
| Blacklist.IPs     | TStringList zawierający zablokowane adresy IP lub zakresy CIDR. |

## Dodawanie adresów IP w czasie projektowania

Kliknij właściwość IPs w Inspektorze obiektów, aby otworzyć edytor listy ciągów znaków. Dodaj jeden adres IP lub zakres CIDR w każdym wierszu:

```
192.168.1.100
10.0.0.0/8
172.16.0.0/12
```

## Dodawanie adresów IP w czasie działania

```
sgcWebSocketFirewall1->Blacklist->Enabled = true;
sgcWebSocketFirewall1->Blacklist->IPs->Add("192.168.1.100");
sgcWebSocketFirewall1->Blacklist->IPs->Add("10.0.0.0/8");
```

## Whitelist

Lista dozwolonych adresów przyznaje bezwarunkowy dostęp określonym adresom IP. **Adresy IP z listy dozwolonych omijają wszystkie pozostałe sprawdzenia zapory**, w tym czarną listę, blokady wynikające z ataków brute force, limity szybkości i filtrowanie wiadomości.

| Właściwość        | Opis                                                                  |
|-------------------|-----------------------------------------------------------------------|
| Whitelist.Enabled | Włącza lub wyłącza sprawdzanie białej listy. Wartość domyślna: False. |
| Whitelist.IPs     | TStringList zawierający zaufane adresy IP lub zakresy CIDR.           |

## Przykład

```
// Allow internal network unconditionally
sgcWebSocketFirewall1->Whitelist->Enabled = true;
sgcWebSocketFirewall1->Whitelist->IPs->Add("192.168.1.0/24");
sgcWebSocketFirewall1->Whitelist->IPs->Add("127.0.0.1");
```

## Kolejność priorytetów

Gdy obie listy (czarna i biała) są włączone, zapora ocenia je w następującej kolejności:

1. Jeśli adres IP znajduje się na białej liście, połączenie jest **natychmiast dopuszczane**. Dalsze sprawdzenia nie są wykonywane.
2. Jeśli adres IP znajduje się na czarnej liście, połączenie zostaje **odrzucone**.
3. Jeśli adres IP nie znajduje się na żadnej z list, połączenie przechodzi do innych sprawdzeń (brute force, ograniczanie liczby żądań itd.).

Oznacza to, że wpis na białej liście ma zawsze pierwszeństwo przed wpisem na czarnej liście dla tego samego adresu IP.

## Notacja CIDR

Zarówno czarna lista, jak i biała lista obsługują notację CIDR (Classless Inter-Domain Routing) do określania zakresów adresów IP:

| CIDR           | Range                         | Addresses  |
|----------------|-------------------------------|------------|
| 192.168.1.0/24 | 192.168.1.0 - 192.168.1.255   | 256        |
| 192.168.0.0/16 | 192.168.0.0 - 192.168.255.255 | 65,536     |
| 10.0.0.0/8     | 10.0.0.0 - 10.255.255.255     | 16,777,216 |
| 172.16.0.0/12  | 172.16.0.0 - 172.31.255.255   | 1,048,576  |

Na tej samej liście można mieszać dokładne adresy IP i zakresy CIDR:

```
sgcWebSocketFirewall1->Blacklist->IPs->Add("203.0.113.50");    // single IP
sgcWebSocketFirewall1->Blacklist->IPs->Add("198.51.100.0/24");    // entire subnet
```

## Łączenie czarnej listy i białej listy

Powszechnym wzorcem jest blokowanie szerokiego zakresu adresów przy jednoczesnym zezwoleniu na określone adresy IP w tym zakresie:

```
// Block the entire 10.x.x.x range
sgcWebSocketFirewall1->Blacklist->Enabled = true;
sgcWebSocketFirewall1->Blacklist->IPs->Add("10.0.0.0/8");

// But allow the monitoring server
sgcWebSocketFirewall1->Whitelist->Enabled = true;
sgcWebSocketFirewall1->Whitelist->IPs->Add("10.1.1.50");
```

W tym przykładzie wszystkie adresy IP z zakresu 10.x.x.x są zablokowane z wyjątkiem 10.1.1.50, który jest na białej liście i omija wszystkie sprawdzenia.

# Zapora sieciowa: Ochrona przed atakami brute force

Moduł ochrony przed atakami brute force wykrywa powtarzające się nieudane próby uwierzytelnienia z tego samego adresu IP i automatycznie blokuje ten adres na konfigurowalny czas.

## Jak to działa

Zapora sieciowa śledzi nieudane próby per adres IP w przesuwającym się oknie czasowym. Gdy liczba błędów przekroczy skonfigurowany próg, adres IP jest automatycznie blokowany. Zablokowane adresy IP są odrzucane na poziomie połączenia, przed wywołaniem jakichkolwiek zdarzeń serwera.

1. Klient łączy się i próbuje uwierzytelnienia.
2. Jeśli uwierzytelnianie nie powiedzie się, kod wywołuje metodę **RegisterFailedAttempt(IP)** w celu powiadomienia zapory sieciowej.
3. Zapora sieciowa zwiększa licznik błędów dla danego adresu IP w bieżącym oknie czasowym.
4. Jeśli licznik osiągnie **MaxAttempts**, adres IP zostaje zablokowany na **BanDurationSec** sekund.
5. Każde nowe połączenie z zablokowanego adresu IP jest natychmiast odrzucane.
6. Po upływie czasu blokady adres IP jest automatycznie odblokowywany, a licznik błędów jest resetowany.

## Właściwości

| Właściwość                | Domyślny | Opis                                                                                                                                       |
|---------------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| BruteForce.Enabled        | False    | Włącza lub wyłącza ochronę przed atakami brute force.                                                                                      |
| BruteForce.MaxAttempts    | 5        | Maksymalna dozwolona liczba nieudanych prób w oknie czasowym przed zablokowaniem adresu IP.                                                |
| BruteForce.TimeWindowSec  | 60       | Czas trwania przesuwającego się okna czasowego w sekundach. Nieudane próby starsze niż ten limit są odrzucane.                             |
| BruteForce.BanDurationSec | 300      | Czas trwania automatycznej blokady w sekundach. Ustawienie wartości 0 oznacza blokadę trwałą (do ręcznego usunięcia lub restartu serwera). |

## RegisterFailedAttempt

Zapora sieciowa nie wie, które połączenia reprezentują nieudane próby logowania. Należy wywołać metodę **RegisterFailedAttempt** z procedury obsługi zdarzenia uwierzytelniania za każdym razem, gdy próba logowania się nie powiedzie:

```
void __fastcall TForm1::ServerAuthentication(TsgcWSConnection *Connection,
    String aUser, String aPassword, bool &Authenticated)
{
    Authenticated = CheckCredentials(aUser, aPassword);
    if (!Authenticated)
```

```
sgcWebSocketFirewall1->RegisterFailedAttempt(Connection->IP);  
}
```

## Manual Ban Management

Oprócz automatycznego blokowania można ręcznie zarządzać blokadami:

| Metoda                 | Opis                                                                        |
|------------------------|-----------------------------------------------------------------------------|
| BanIP(IP, DurationSec) | Ręcznie blokuje adres IP. Użyj DurationSec = 0, aby ustawić trwałą blokadę. |
| UnbanIP(IP)            | Usuwa blokadę z określonego adresu IP.                                      |
| IsBanned(IP)           | Zwraca True, jeżeli adres IP jest aktualnie zablokowany.                    |
| ClearBans              | Usuwa wszystkie aktywne blokady (zarówno automatyczne, jak i ręczne).       |

### Przykład: ręczne zablokowanie

```
// Ban an IP for 1 hour  
sgcWebSocketFirewall1->BanIP("203.0.113.50", 3600);  
  
// Permanent ban  
sgcWebSocketFirewall1->BanIP("198.51.100.25", 0);  
  
// Check and unban  
if (sgcWebSocketFirewall1->IsBanned("203.0.113.50"))  
    sgcWebSocketFirewall1->UnbanIP("203.0.113.50");
```

### Przykład konfiguracji

```
// Strict configuration: 3 attempts in 30 seconds, ban for 10 minutes  
sgcWebSocketFirewall1->BruteForce->Enabled = true;  
sgcWebSocketFirewall1->BruteForce->MaxAttempts = 3;  
sgcWebSocketFirewall1->BruteForce->TimeWindowSec = 30;  
sgcWebSocketFirewall1->BruteForce->BanDurationSec = 600;  
  
// Lenient configuration: 10 attempts in 5 minutes, ban for 1 minute  
sgcWebSocketFirewall1->BruteForce->Enabled = true;  
sgcWebSocketFirewall1->BruteForce->MaxAttempts = 10;  
sgcWebSocketFirewall1->BruteForce->TimeWindowSec = 300;  
sgcWebSocketFirewall1->BruteForce->BanDurationSec = 60;
```

### Uwagi

- Adresy IP na liście dozwolonych nigdy nie są blokowane, nawet jeśli dla nich wywoływana jest metoda RegisterFailedAttempt.
- Blokady są domyślnie przechowywane w pamięci. Użyj **SaveBansToFile** i **LoadBansFromFile**, aby utrzymywać blokady między restartami serwera.
- Okno czasowe jest przesuwne, a nie stałe. Każda nieudana próba jest śledzona indywidualnie i wygasa po TimeWindowSec sekundach.



# Zapora sieciowa: wykrywanie SQL Injection i XSS

TsgcWebSocketFirewall zawiera wbudowane wykrywanie oparte na wzorcach dla ataków SQL injection i Cross-Site Scripting (XSS) w wiadomościach WebSocket. Moduły te skanują treść przychodzących wiadomości i podejmują działanie po wykryciu wzorca zagrożenia.

## Wykrywanie ataku SQL Injection

Moduł SQL injection skanuje wszystkie przychodzące wiadomości WebSocket pod kątem typowych wzorców SQL injection. Wykrywanie jest niezależne od wielkości liter.

### Właściwości

| Właściwość                  | Opis                                                                                                                                                                                                                        |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLInjection.Enabled        | Włącza lub wyłącza wykrywanie wstrzykiwania SQL. Wartość domyślna: False.                                                                                                                                                   |
| SQLInjection.Action         | Akcja podejmowana po wykryciu wzorca: <b>faDeny</b> (blokuje wiadomość i rozłącza, domyślnie), <b>faLog</b> (wywołuje zdarzenie OnViolation, ale przepuszcza wiadomość), <b>faAllow</b> (brak akcji, wykrywanie wyłączone). |
| SQLInjection.CustomPatterns | TStringList z dodatkowymi wzorcami wyrażeń regularnych do sprawdzenia. Są one oceniane jako uzupełnienie wbudowanych wzorców.                                                                                               |

### Wbudowane wzorce

Domyślnie sprawdzane są następujące wzorce (bez rozróżniania wielkości liter):

| Pattern                                  | Typ ataku                                      |
|------------------------------------------|------------------------------------------------|
| ' OR ', ' AND '                          | Wstrzykiwanie oparte na wartościach logicznych |
| UNION SELECT                             | Wstrzykiwanie oparte na UNION                  |
| '; DROP, '; DELETE, '; INSERT, '; UPDATE | Wstrzyknięcie instrukcji / manipulacja danymi  |
| -- (podwójny myślnik)                    | Wstrzykiwanie komentarzy SQL                   |
| EXEC(, EXECUTE(                          | Wykonywanie poleceń                            |
| xp_cmdshell                              | Nadużycie rozszerzonych procedur składowanych  |
| CAST(, CONVERT(                          | Nadużycie konwersji typów                      |
| 1=1, 1='1                                | Wstrzykiwanie tautologiczne                    |

### Niestandardowe wzorce

Wzorce specyficzne dla aplikacji można dodawać za pomocą właściwości CustomPatterns:

```
sgcWebSocketFirewall1->SQLInjection->Enabled = true;  
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("WAITFOR DELAY");  
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("BENCHMARK\\(");  
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("SLEEP\\(");
```

# Wykrywanie XSS

Moduł XSS wykrywa wzorce Cross-Site Scripting w wiadomościach WebSocket, zapobiegając wstrzykiwaniu złośliwych skryptów, które mogłyby być renderowane przez innych połączonych klientów.

## Właściwości

| Właściwość  | Opis                                                                                                                                                                                                |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XSS.Enabled | Włącza lub wyłącza wykrywanie XSS. Domyślnie: False.                                                                                                                                                |
| XSS.Action  | Akcja podejmowana po wykryciu wzorca: <b>faDeny</b> (blokowanie i rozłączenie, domyślnie), <b>faLog</b> (wywołanie zdarzenia OnViolation z zezwoleniem na przesłanie), <b>faAllow</b> (brak akcji). |

## Wbudowane wzorce

Wykrywane są następujące wzorce XSS (bez rozróżniania wielkości liter):

| Pattern                                   | Typ ataku                       |
|-------------------------------------------|---------------------------------|
| <script                                   | Wstrzykiwanie znacznika skryptu |
| javascript:                               | XSS oparty na protokole         |
| onerror=, onload=, onclick=, onmouseover= | Wstrzykiwanie obsługi zdarzeń   |
| eval(                                     | Ocena kodu JavaScript           |
| document.cookie                           | Kradzież plików cookie          |
| <iframe, <object, <embed                  | Wstrzykiwanie elementów         |
| <svg onload                               | XSS oparty na SVG               |
| expression(                               | Wstrzyknięcie wyrażeń CSS       |

## Właściwość Action

Moduły SQLInjection i XSS obsługują właściwość Action typu TsgcFirewallAction:

| Wartość | Zachowanie                                                                                                              |
|---------|-------------------------------------------------------------------------------------------------------------------------|
| faDeny  | Wiadomość jest blokowana, a klient jest rozłączany. Wywoływane jest zdarzenie OnViolation. Jest to ustawienie domyślne. |
| faLog   | Wiadomość jest przepuszczana, jednak zdarzenie OnViolation jest wywoływane w celach rejestrowania.                      |
| faAllow | Nie jest podejmowane żadne działanie. Skutecznie wyłącza wykrywanie, pozostawiając flagę Enabled ustawioną na True.     |

## Zdarzenie OnViolation

Gdy wykryty zostanie wzorzec SQL injection lub XSS (przy parametrze Action ustawionym na faDeny lub faLog), zdarzenie OnViolation jest wywoływane ze szczegółami dotyczącymi naruszenia:

```

void __fastcall TForm1::FirewallViolation(TObject *Sender, const String aIP,
const TsgcFirewallViolationType aViolationType, const String aDetails)
{
    switch (aViolationType)
    {
        case fvSQLInjection:
            LogWarning("SQL injection attempt from " + aIP + ": " + aDetails);
            break;
        case fvXSS:
            LogWarning("XSS attempt from " + aIP + ": " + aDetails);
            break;
    }
}

```

## Przykład konfiguracji

```

// Enable SQL injection detection with deny action
sgcWebSocketFirewall1->SQLInjection->Enabled = true;
sgcWebSocketFirewall1->SQLInjection->Action = faDeny;

// Add custom patterns for time-based SQL injection
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("WAITFOR DELAY");
sgcWebSocketFirewall1->SQLInjection->CustomPatterns->Add("SLEEP\\(");

// Enable XSS detection in log-only mode (monitor without blocking)
sgcWebSocketFirewall1->XSS->Enabled = true;
sgcWebSocketFirewall1->XSS->Action = faLog;

// Assign violation handler
sgcWebSocketFirewall1->OnViolation = FirewallViolation;

```

## Uwagi

- Wykrywanie wzorców jest stosowane do wszystkich przychodzących wiadomości tekstowych. Wiadomości binarne nie są skanowane.
- Adresy IP z białej listy całkowicie omijają filtrowanie wiadomości.
- Wbudowane wzorce obejmują najczęstsze wektory ataku. W przypadku zagrożeń specyficznych dla aplikacji należy dodać niestandardowe wzorce za pomocą `SQLInjection.CustomPatterns`.
- Podczas początkowego wdrożenia należy stosować tryb `faLog` w celu monitorowania wykryć bez wpływu na klientów, a następnie przełączyć się na `faDeny` po zweryfikowaniu wzorców dla danej aplikacji.

# Firewall: Ograniczanie szybkości i ochrona przed atakami flood

TsgcWebSocketFirewall zapewnia dwa uzupełniające się mechanizmy kontroli ruchu: ograniczanie liczby połączeń (kontrolowanie liczby połączeń, które adres IP może otworzyć) oraz ochronę przed zalewaniem wiadomościami (kontrolowanie liczby wiadomości, które połączony klient może wysłać na sekundę).

## Ograniczanie szybkości połączeń

Ograniczanie szybkości połączeń ogranicza liczbę połączeń, które może nawiązać jeden adres IP w danym oknie czasowym. Zapobiega to wyczerpaniu przez jednego klienta wszystkich dostępnych połączeń serwera.

### Właściwości

| Właściwość                    | Domyślny | Opis                                                                                                             |
|-------------------------------|----------|------------------------------------------------------------------------------------------------------------------|
| RateLimit.Enabled             | False    | Włącza lub wyłącza ograniczanie liczby połączeń.                                                                 |
| RateLimit.MaxConnectionsPerIP | 10       | Maksymalna liczba jednoczesnych połączeń dozwolonych z jednego adresu IP.                                        |
| RateLimit.TimeWindowSec       | 60       | Okno czasowe w sekundach służące do śledzenia połączeń. Połączenia starsze niż ta wartość są usuwane z licznika. |

### Jak to działa

1. Gdy nadchodzi nowe połączenie, zapora zlicza aktywne połączenia z danego adresu IP.
2. Jeśli licznik osiągnął wartość MaxConnectionsPerIP lub ją przekroczył, nowe połączenie jest odrzucone.
3. Po rozłączeniu połączenia jest ono usuwane z licznika śledzenia.
4. Połączenia starsze niż TimeWindowSec są automatycznie usuwane ze śledzenia.

### Przykład

```
// Allow maximum 5 connections per IP
sgcWebSocketFirewall1->RateLimit->Enabled = true;
sgcWebSocketFirewall1->RateLimit->MaxConnectionsPerIP = 5;
sgcWebSocketFirewall1->RateLimit->TimeWindowSec = 60;
```

## Ochrona przed zalewaniem wiadomościami

Ochrona przed zalewem ogranicza liczbę wiadomości, jaką jeden adres IP może wysłać na sekundę. Zapobiega to przeciążaniu serwera przez nadużywające klientów gwałtownymi seriami wiadomości.

## Właściwości

| Właściwość                        | Domyślny | Opis                                                                                                                                            |
|-----------------------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| FloodProtection.Enabled           | False    | Włącza lub wyłącza ochronę przed zalewaniem wiadomościami.                                                                                      |
| FloodProtection.MaxMessagesPerSec | 100      | Maksymalna liczba wiadomości dozwolona na sekundę z jednego adresu IP.                                                                          |
| FloodProtection.Action            | faDeny   | Akcja po przekroczeniu limitu: <b>faDeny</b> (rozłącza klienta), <b>faLog</b> (wywołuje OnViolation, ale zezwala), <b>faAllow</b> (brak akcji). |

## Jak to działa

1. Zapora sieciowa utrzymuje licznik wiadomości dla każdego adresu IP, który resetuje się co sekundę.
2. Każda przychodząca wiadomość zwiększa licznik dla adresu IP nadawcy.
3. Gdy licznik przekroczy MaxMessagesPerSec, wykonywana jest skonfigurowana akcja.
4. Przy akcji faDeny klient jest natychmiast rozłączany. Przy akcji faLog zdarzenie OnViolation jest wywoływane, ale wiadomość jest przetwarzana normalnie.

## Przykład

```
// Limit to 50 messages per second per IP
sgcWebSocketFirewall1->FloodProtection->Enabled = true;
sgcWebSocketFirewall1->FloodProtection->MaxMessagesPerSec = 50;
sgcWebSocketFirewall1->FloodProtection->Action = faDeny;
```

## Łączenie obydwu mechanizmów ochrony

Ograniczanie szybkości i ochrona przed zalewaniem działają razem, zapewniając kompleksową kontrolę ruchu:

```
// Connection limiting: max 5 connections per IP
sgcWebSocketFirewall1->RateLimit->Enabled = true;
sgcWebSocketFirewall1->RateLimit->MaxConnectionsPerIP = 5;

// Message limiting: max 50 messages/sec per IP
sgcWebSocketFirewall1->FloodProtection->Enabled = true;
sgcWebSocketFirewall1->FloodProtection->MaxMessagesPerSec = 50;

// Assign to server
sgcWebSocketHTTPServer1->Firewall = sgcWebSocketFirewall1;
```

W tej konfiguracji każdy adres IP może otworzyć co najwyżej 5 jednoczesnych połączeń, a przez wszystkie te połączenia może wysyłać łącznie maksymalnie 50 wiadomości na sekundę.

## Wskazówki dotyczące dostrajania

| Scenariusz | MaxConnectionsPerIP | MaxMessagesPerSec |
|------------|---------------------|-------------------|
|------------|---------------------|-------------------|

|                                       |       |          |
|---------------------------------------|-------|----------|
| Aplikacja czatu                       | 3-5   | 10-30    |
| Strumień danych w czasie rzeczywistym | 2-3   | 100-500  |
| Serwer gier                           | 1-2   | 50-100   |
| Brama API                             | 10-20 | 200-1000 |

Należy zacząć od konserwatywnych limitów i zwiększać je na podstawie monitorowania. Na początku warto używać trybu faLog, aby obserwować wzorce ruchu sieciowego przed włączeniem trybu faDeny.

## Uwagi

- Adresy IP z białej listy omijają zarówno ograniczanie szybkości, jak i ochronę przed zalewaniem.
- Dane śledzenia limitów zapytań są automatycznie czyszczone po zamknięciu połączeń. Metoda ClearTracking służy do ręcznego resetowania wszystkich liczników.
- Za odwrotnym serwerem proxy wszystkie połączenia mogą wyglądać tak, jakby pochodziły z adresu IP serwera proxy. Należy skonfigurować serwer proxy, aby przekazywał rzeczywisty adres IP klienta (np. nagłówek X-Forwarded-For) i upewnić się, że serwer rozwiązuje go poprawnie.

# Zapora: Wykrywanie botów

TsgcWebSocketFirewall potrafi klasyfikować łączące się adresy IP jako boty, roboty indeksujące, hosty centrów danych lub odwiedzających będących ludźmi. Wykrywanie botów łączy bazę danych znanych botów, weryfikację odwrotnego DNS potwierdzonego w przód, dopasowanie zakresów centrów danych/ASN oraz wyszukiwania w listach blokujących DNSBL, aby przypisać każdemu adresowi IP klasyfikację. Jest to moduł **wyłącznie klasyfikujący**: nigdy samodzielnie nie blokuje połączenia. Zgłasza werdykt, dzięki czemu Twoja aplikacja może zdecydować, co zrobić.

## Jak to działa

1. Adres IP łączy się z serwerem.
2. Firewall wyszukuje adres IP w pamięci podręcznej wyników. Jeśli istnieje klasyfikacja w pamięci podręcznej (i nie wygasła), jest używana natychmiast.
3. Przy braku trafienia w pamięci podręcznej adres IP jest porównywany z plikiem znanych botów (zweryfikowane roboty indeksujące) oraz plikiem ASN centrów danych (zakresy chmurowe/hostingowe).
4. Jeśli włączona jest weryfikacja ReverseDNS lub DNSBL, wyszukiwanie jest planowane w wątku w tle, aby nigdy nie blokowało wątku akceptacji. Do czasu zakończenia wyszukiwania adres IP może być sklasyfikowany jako **bcUnknown**.
5. Po rozwiązaniu klasyfikacji jest ona zapisywana w pamięci podręcznej (na CacheDurationSec sekund) i wywoływane jest zdarzenie **OnBotDetected**.
6. Klasyfikacja aktualizuje Stats firewalla (licznik naruszeń **fvBot**) i jest dostępna dla reguł niestandardowych poprzez właściwość reguły **BotType**.

## Właściwości

| Właściwość                      | Opis                                                                                                                                                    |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| BotDetection.Enabled            | Włącza lub wyłącza wykrywanie botów. Domyślnie: False.                                                                                                  |
| BotDetection.KnownBotsFile      | Ścieżka do pliku CSV zweryfikowanych robotów indeksujących. Każdy wiersz ma postać <b>CIDR,botname</b> (np. zakres IP i robot, do którego należy).      |
| BotDetection.CacheDurationSec   | Liczba sekund, przez które rozwiązana klasyfikacja jest przechowywana w pamięci podręcznej wyników, zanim zostanie ponownie wyszukana. Domyślnie: 3600. |
| BotDetection.DNSTimeoutMS       | Limit czasu w milisekundach dla każdego wyszukiwania DNS (odwrotny DNS i DNSBL). Domyślnie: 2000.                                                       |
| BotDetection.ReverseDNS.Enabled | Włącza weryfikację odwrotnego DNS potwierdzonego w przód (FCrDNS) deklarowanych robotów indeksujących. Domyślnie: False.                                |
| BotDetection.Datacenter.Enabled | Włącza wykrywanie zakresów centrów danych / hostingu w chmurze. Domyślnie: False.                                                                       |

|                                 |                                                                                                                                                         |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| BotDetection.Datacenter.ASNFile | Ścieżka do pliku CSV zawierającego zakresy ASN centrów danych. Każdy wiersz ma postać <b>CIDR,asname</b> (np. zakres dostawcy chmury i jego nazwa ASN). |
| BotDetection.DNSBL.Enabled      | Włącza wyszukiwania DNSBL (DNS blocklist). Domyślnie: False.                                                                                            |
| BotDetection.DNSBL.Zones        | TStringList stref DNSBL do odpytania (na przykład <b>zen.spamhaus.org</b> ).                                                                            |

## Wartości klasyfikacji

Wyliczenie **TsgcBotClassification** opisuje werdykt przypisany do adresu IP:

| Wartość            | Opis                                                                                                                                               |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| bcUnknown          | Adres IP nie został jeszcze sklasyfikowany. Zwracane przy braku trafienia w pamięci podręcznej, gdy trwa asynchroniczne wyszukiwanie.              |
| bcVerified-Crawler | Znany, legalny robot indeksujący (na przykład bot wyszukiwarki) potwierdzony przez plik znanych botów lub przez odwrotny DNS potwierdzony w przód. |
| bcDatacenter       | Adres IP należy do zakresu dostawcy chmury lub hostingu wymienionego w pliku ASN. Często jest to ruch automatyczny, a nie użytkownik domowy.       |
| bcSuspected-Bot    | Adres IP wykazuje cechy bota, ale nie udało się go dopasować do zweryfikowanego robota indeksującego.                                              |
| bcBlocklisted      | Adres IP znajduje się na jednej ze skonfigurowanych stref DNSBL.                                                                                   |
| bcHuman            | Adres IP wygląda jak zwykły odwiedzający będący człowiekiem (brak dopasowania do bota, centrum danych lub listy blokującej).                       |

## Formaty plików CSV

Plik znanych botów (**KnownBotsFile**) zawiera listę zakresów zweryfikowanych robotów indeksujących. Każdy wiersz to zakres CIDR, po którym następuje nazwa bota:

```
66.249.64.0/19,Googlebot
157.55.0.0/16,Bingbot
17.58.0.0/16,Applebot
```

Plik ASN centrów danych (**Datacenter.ASNFile**) zawiera listę zakresów chmurowych/hostingowych. Każdy wiersz to zakres CIDR, po którym następuje nazwa ASN:

```
3.0.0.0/15,AMAZON-AES
13.64.0.0/11,MICROSOFT-AZURE
34.64.0.0/10,GOOGLE-CLOUD
```

## Włączanie wykrywania botów

Włącz moduł, wczytaj pliki znanych botów i ASN oraz włącz potrzebne podmoduły weryfikacji. Wyszukiwania DNS działają w wątku w tle, więc włączenie ReverseDNS lub DNSBL nie spowalnia wątku akceptacji.



```
// Enable bot detection
sgcWebSocketFirewall1->BotDetection->Enabled = true;
sgcWebSocketFirewall1->BotDetection->KnownBotsFile = "C:\\data\\known_bots.csv";
sgcWebSocketFirewall1->BotDetection->CacheDurationSec = 3600;
sgcWebSocketFirewall1->BotDetection->DNSTimeoutMS = 2000;

// Forward-confirmed reverse DNS verification
sgcWebSocketFirewall1->BotDetection->ReverseDNS->Enabled = true;

// Datacenter / cloud-hosting detection
sgcWebSocketFirewall1->BotDetection->Datacenter->Enabled = true;
sgcWebSocketFirewall1->BotDetection->Datacenter->ASNFile = "C:\\data\\datacenter_asn.csv";

// DNSBL blocklist lookups
sgcWebSocketFirewall1->BotDetection->DNSBL->Enabled = true;
sgcWebSocketFirewall1->BotDetection->DNSBL->Zones->Add("zen.spamhaus.org");
```

## Zachowanie wyłącznie klasyfikujące i OnBotDetected

Wykrywanie botów nigdy samo z siebie nie odrzuca połączenia. Gdy adres IP zostaje sklasyfikowany, wyzwalane jest zdarzenie **OnBotDetected**, aktualizowane są statystyki zapory (licznik **fvBot**) i wynik jest zapisywany w pamięci podręcznej. Użyj tego zdarzenia, aby rejestrować ruch, budować analitykę lub zasilać własną logikę:

```
void __fastcall TForm1::FirewallBotDetected(TObject *Sender, const String aIP,
    TsgcBotClassification aClassification, const String aBotName)
{
    switch (aClassification)
    {
        case bcVerifiedCrawler: Log("Verified crawler " + aBotName + " from " + aIP); break;
        case bcDatacenter:      Log("Datacenter host from " + aIP); break;
        case bcBlocklisted:     Log("Blocklisted IP " + aIP); break;
    }
}
```

## Reagowanie na klasyfikację za pomocą reguły niestandardowej

Ponieważ wykrywanie botów tylko klasyfikuje, blokowanie lub banowanie odbywa się za pośrednictwem silnika reguł niestandardowych. Każdy **TsgcFirewallRuleItem** udostępnia właściwość **BotType**: gdy jest ustawiona, reguła dopasowuje połączenia, których klasyfikacja jest równa tej wartości, a Ty możesz je wtedy odrzucić, zbanować lub zarejestrować.

```
sgcWebSocketFirewall1->CustomRules->Enabled = true;

// Deny connections coming from blocklisted IPs
TsgcFirewallRuleItem *vRule =
    (TsgcFirewallRuleItem *)sgcWebSocketFirewall1->CustomRules->Rules->Add();
vRule->BotType = bcBlocklisted;
vRule->ActionType = raDeny;
```

## Werdykty niestandardowe z OnResolveBot

Jeśli dysponujesz własną inteligencją wykrywania botów (zewnętrzna usługa reputacji, wewnętrzna lista dozwolonych/zablokowanych itp.), przypisz zdarzenie **OnResolveBot**. Po przypisaniu Twój handler dostarcza werdykt zamiast wbudowanego resolvera. Ustaw parametry *var aClassification* i *aBotName*, aby zwrócić swój wynik:

```
void __fastcall TForm1::FirewallResolveBot(TObject *Sender, const String aIP,
    TsgcBotClassification &aClassification, String &aBotName)
{
    // Custom lookup - query your own bot intelligence service
    if (MyBotService->IsCrawler(aIP))
    {
        aClassification = bcVerifiedCrawler;
        aBotName = MyBotService->GetName(aIP);
    }
    else
        aClassification = bcHuman;
}
```

**Uwaga:** Gdy OnResolveBot jest przypisany, wbudowany resolver (plik znanych botów, odwrotny DNS, sprawdzenia centrów danych i DNSBL) jest pomijany dla tego adresu IP. Werdykt zwrócony przez Twój handler jest zapisywany w pamięci podręcznej i przekazywany do OnBotDetected oraz reguł niestandardowych dokładnie tak, jak wynik wbudowany.

# TsgcWSRateLimiter

TsgcWSRateLimiter implementuje w pełni funkcjonalny komponent ograniczania szybkości, chroniący punkty końcowe serwera przed nadmiernym ruchem, nadużyciami i scrapingiem.

## Wprowadzenie

TsgcWSRateLimiter implementuje w pełni funkcjonalny komponent ograniczania szybkości, który chroni punkty końcowe serwera przed nadmiernym ruchem, nadużyciami i scrapingiem. Obsługuje trzy algorytmy (Token Bucket, Sliding Window, Fixed Window), zasady o określonym zakresie (per IP, per klucz API, per użytkownik, per punkt końcowy), długoterminowe limity (godzina / dzień / miesiąc), ochronę przed ruchem szczytowym (burst protection), konfigurowalny kształt odpowiedzi HTTP 429 oraz statystyki w czasie rzeczywistym. Stan wewnętrzny można utrwalić w pliku lub strumieniu, dzięki czemu liczniki zachowują wartości po ponownym uruchomieniu.

Ogranicznik przepustowości integruje się automatycznie z komponentami serwerowymi przez właściwość **RateLimiter** dostępną w TsgcWebSocketHTTPServer i TsgcWSServer\_HTTPAPI. Po przypisaniu ocenia każde połączenie i każdą wiadomość bez konieczności ręcznego okablowania zdarzeń.

## Szybki start

1. Upuść komponent **TsgcWSRateLimiter** na formularz.
2. Włącz jedną lub więcej strategii i zakresów (patrz dokumentacja poszczególnych sekcji poniżej).
3. Przypisz ogranicznik liczby żądań (rate limiter) do komponentu serwera:

```
sgcWebSocketHTTPServer1.RateLimiter := sgcWSRateLimiter1;
```

4. Uruchom serwer. Mechanizm ograniczania liczby żądań automatycznie ocenia każde połączenie i wiadomość, odrzuca nadmierny ruch z kodem HTTP 429 i aktualizuje liczniki statystyk.

## Właściwości najwyższego poziomu

**Działanie:** Główny przełącznik komponentu oraz opcjonalny plik magazynu używany przez metody trwałości stanu. Gdy właściwość Enabled ma wartość False, każda metoda publiczna zwraca wynik „dzwolone” i żadne liczniki nie są aktualizowane.

**Kiedy używać:** Wyłącz ogranicznik przepustowości w czasie wykonywania podczas kontrolowanego testu obciążeniowego lub w oknie konserwacyjnym bez jego odłączania od serwera. Ustaw StorageFile, aby liczniki przetrwały ponowne uruchomienie aplikacji — na przykład ciągły Token Bucket kontynuuje napełnianie od miejsca, w którym się zatrzymał, zamiast zaczynać od nowa i akceptować duży skok ruchu.

| Właściwość   | Typ     | Domyślny | Opis                                                                                                                                                                          |
|--------------|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączony     | Boolean | True     | Przełącznik główny. Gdy wartość wynosi False, metody IsAllowed / Consume / IsConnectionAllowed / IsMessageAllowed zawsze zwracają True i żadne liczniki nie są aktualizowane. |
| Storage-File | string  | ""       | Opcjonalna ścieżka pliku używana przez SaveStateToFile / LoadStateFromFile do utrwalania liczników między restartami.                                                         |

```
sgcWSRateLimiter1->Enabled = true;  
sgcWSRateLimiter1->StorageFile = "ratelimit.dat";  
sgcWSRateLimiter1->LoadStateFromFile(sgcWSRateLimiter1->StorageFile);
```

## Strategia TokenBucket

**Działanie:** Utrzymuje zasobnik zawierający do *Capacity* tokenów na klucz. Zasobnik jest uzupełniany z prędkością *RefillRate* tokenów co *RefillIntervalMs* milisekund. Każde żądanie zużywa jeden token (lub wartość argumentu *Cost* przekazaną do *Consume*). Gdy zasobnik jest pusty, żądanie jest odrzucane, a wskazówka *RetryAfterSec* jest obliczana na podstawie harmonogramu uzupełniania.

**Kiedy stosować:** API deklaruje średnie tempo 10 żądań/s, ale chcemy pozwolić poprawnie zachowującym się klientom na serię do 50 żądań bez ograniczania przepustowości. Token Bucket z *Capacity=50* i *RefillRate=10* na 1000 ms pozwala klientowi wysłać 50 żądań natychmiastowo, a następnie wymusza ograniczenie do 10 żądań/s do czasu uzupełnienia zasobnika.

| Właściwość | Typ     | Domyślny | Opis                                                           |
|------------|---------|----------|----------------------------------------------------------------|
| Włączone   | Boolean | False    | Włącza algorytm Token Bucket jako globalną strategię domyślną. |

|                  |         |      |                                                                                             |
|------------------|---------|------|---------------------------------------------------------------------------------------------|
| Capacity         | Integer | 100  | Maksymalna liczba tokenów, jaką może pomieścić zasobnik. Określa maksymalną wielkość serii. |
| RefillRate       | Integer | 10   | Liczba tokenów dodawanych do zasobnika co RefillIntervalMs.                                 |
| RefillIntervalMs | Integer | 1000 | Okres uzupełniania w milisekundach. Razem z RefillRate określa trwałą szybkość żądań.       |

```
// Average 10 req/sec, bursts up to 50
sgcWSRateLimiter1->TokenBucket->Enabled = true;
sgcWSRateLimiter1->TokenBucket->Capacity = 50;
sgcWSRateLimiter1->TokenBucket->RefillRate = 10;
sgcWSRateLimiter1->TokenBucket->RefillIntervalMs = 1000;
// First 50 requests succeed immediately; afterwards throttled to 10/sec until refill
```

## Strategia SlidingWindow

**Działanie:** Przechowuje znacznik czasu dla każdego żądania i zlicza znaczniki czasu mieszczące się w przesuwym oknie o długości *WindowSec* sekund. Odrzuca żądanie, gdy liczba przekroczyłaby *MaxRequests*. Okno przesuwa się w sposób ciągły, więc nie ma momentu twardego resetowania, podczas którego mogłoby przejść nagłe obciążenie.

**Kiedy stosować:** Publikujesz ściśle SLA gwarantujące „nie więcej niż 100 żądań w dowolnym 60-sekundowym oknie kroczącym”. Algorytm Fixed Window pozwoliłby klientowi wysłać 100 żądań o 12:00:59 i kolejne 100 o 12:01:00 (200 w ciągu jednej sekundy). Sliding Window zapobiega temu i ściśle egzekwuje SLA.

| Właściwość  | Typ     | Domyślny | Opis                                                                                 |
|-------------|---------|----------|--------------------------------------------------------------------------------------|
| Włączone    | Boolean | False    | Włącza algorytm Sliding Window jako globalną domyślną strategię.                     |
| WindowSec   | Integer | 60       | Rozmiar okna kroczącego w sekundach.                                                 |
| MaxRequests | Integer | 1000     | Maksymalna liczba żądań dozwolona w dowolnym kroczącym oknie o szerokości WindowSec. |

```
// Strict SLA: max 100 requests in any rolling 60 seconds
sgcWSRateLimiter1->SlidingWindow->Enabled = true;
sgcWSRateLimiter1->SlidingWindow->WindowSec = 60;
sgcWSRateLimiter1->SlidingWindow->MaxRequests = 100;
// Client that sends a burst of 100 requests near t=59s will be throttled at t=60s
```

# Strategia FixedWindow

**Działanie:** Zlicza żądania w stałym oknie czasowym wynoszącym *WindowSec* sekund i resetuje licznik do zera na granicy okna. Jest to najprostszy algorytm i najtańszy pod względem pamięci; jednak może zezwolić na do  $2 \times \text{MaxRequests}$  w krótkim okresie wokół granicy okna.

**Kiedy stosować:** Limity w stylu rozliczeniowym, np. „1000 żądań na godzinę, licznik resetuje się o pełnej godzinie”. Klienci oczekują wyraźnie widocznego czasu resetowania, a drobne przekroczenia graniczne są akceptowalne, ponieważ jednostką rozliczeniową jest godzina, a nie krocząca minuta.

| Właściwość  | Typ      | Domyślny | Opis                                                                            |
|-------------|----------|----------|---------------------------------------------------------------------------------|
| Włączone    | Boole-an | False    | Włącza algorytm Fixed Window jako domyślną strategię globalną.                  |
| WindowSec   | Integer  | 60       | Rozmiar stałego okna w sekundach. Licznik jest resetowany do 0 po upływie okna. |
| MaxRequests | Integer  | 1000     | Maksymalna liczba żądań dozwolona w jednym stałym oknie czasowym.               |

```
// Billing-style: 1000 requests per hour, resets on the hour
sgcWSRateLimiter1->FixedWindow->Enabled = true;
sgcWSRateLimiter1->FixedWindow->WindowSec = 3600;
sgcWSRateLimiter1->FixedWindow->MaxRequests = 1000;
```

## PerIP

**Działanie:** Definiuje limit przepustowości przypisany do źródłowego adresu IP. Gdy klucz żądania nie pasuje do reguł PerAPIKey / PerUser / PerEndpoint, reguła PerIP staje się aktywna i zastępuje globalne wartości domyślne Token/Sliding/Fixed. Każdy adres IP ma własny niezależny licznik.

**Kiedy stosować:** Publiczne API z wieloma klientami. Bez PerIP jeden hałaśliwy klient może wyczerpać globalny budżet i zagłodzić wszystkich pozostałych. Przyznanie każdemu adresowi IP własnego zasobnika (na przykład 100 żądań na minutę) zapobiega wpływaniu jednego klienta na pozostałych.

| Właściwość | Typ | Domyślny | Opis |
|------------|-----|----------|------|
|------------|-----|----------|------|

|             |                       |                |                                                                                                                                                      |
|-------------|-----------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączone    | Boolean               | False          | Włącza ograniczanie liczby żądań na adres IP.                                                                                                        |
| MaxRequests | Integer               | 100            | Maksymalna liczba żądań na okno (lub pojemność zasobnika, gdy strategia to rlsTokenBucket).                                                          |
| WindowSec   | Integer               | 60             | Rozmiar okna w sekundach. W przypadku Token Bucket służy również do wyznaczania szybkości uzupełniania (MaxRequests / WindowSec tokenów na sekundę). |
| Strategia   | TsgcRateLimitStrategy | rlsTokenBucket | Algorytm używany dla tego zakresu: rlsTokenBucket, rlsSliding lub rlsFixed.                                                                          |

```
// Every client IP gets its own 100 req/min bucket
sgcWSRateLimiter1->PerIP->Enabled = true;
sgcWSRateLimiter1->PerIP->Strategy = rlsTokenBucket;
sgcWSRateLimiter1->PerIP->MaxRequests = 100;
sgcWSRateLimiter1->PerIP->WindowSec = 60;
```

## PerAPIKey

**Działanie:** Ta sama semantyka co PerIP, ale w zakresie identyfikatora klucza API. Komponent rozpoznaje klucze zaczynające się od prefiksu `apikey:` jako objęte zakresem klucza API. Zaprojektowany z myślą o integracji z `TsgcWSAPIKeyManager`, dzięki czemu każdy klient otrzymuje niezależny budżet limitów, niezależnie od źródłowego adresu IP.

**Kiedy stosować:** SaaS z wielopoziomowym cennikiem. Każdy klient ma własny klucz API i własny limit niezależny od miejsca połączenia. Należy użyć `PerAPIKey` jako domyślnego poziomu i różnicować poziomy klientów (Free / Pro / Enterprise) za pomocą reguły `PerEndpoint` z kluczem prefiksu ścieżki lub poprzez nadpisanie decyzji w procedurze obsługi zdarzenia `OnThrottled`.

| Właściwość  | Typ                   | Domyślny       | Opis                                                                              |
|-------------|-----------------------|----------------|-----------------------------------------------------------------------------------|
| Włączone    | Boolean               | False          | Włącza ograniczanie szybkości per klucz API.                                      |
| MaxRequests | Integer               | 100            | Maksymalna liczba żądań na okno (lub pojemność kubelka w przypadku Token Bucket). |
| WindowSec   | Integer               | 60             | Rozmiar okna w sekundach.                                                         |
| Strategia   | TsgcRateLimitStrategy | rlsTokenBucket | Algorytm używany dla tego zakresu.                                                |

```
// Default API-key tier: 1000 req/min
sgcWSRateLimiter1->PerAPIKey->Enabled = true;
sgcWSRateLimiter1->PerAPIKey->Strategy = rlsSliding;
sgcWSRateLimiter1->PerAPIKey->MaxRequests = 1000;
sgcWSRateLimiter1->PerAPIKey->WindowSec = 60;
// Tier-differentiation can be done via PerEndpoint rules or OnThrottled
```

## PerUser

**Co robi:** Ta sama struktura co PerIP / PerAPIKey, ale z kluczem będącym identyfikatorem użytkownika zdefiniowanym przez aplikację. Komponent rozpoznaje klucze zaczynające się od prefiksu user : jako należące do zakresu użytkownika. Przydatne, gdy jeden użytkownik może łączyć się z wielu urządzeń lub adresów IP, a ma być stosowany jeden wspólny budżet dla tego użytkownika.

**Kiedy stosować:** Uwierzytelniona aplikacja webowa lub mobilna. Użytkownik `alice@example.com` ma prawo do 1000 żądań dziennie niezależnie od tego, czy korzysta z telefonu, laptopa czy komputera służbowego. Agregacja według adresu IP prowadziłaby do podwójnego liczenia; agregacja według użytkownika daje budżet, który faktycznie ma być rozliczany.

| Właściwość  | Typ                    | Domyślny        | Opis                                                                              |
|-------------|------------------------|-----------------|-----------------------------------------------------------------------------------|
| Włączone    | Boolean                | False           | Włącza ograniczenie częstotliwości na poziomie użytkownika.                       |
| MaxRequests | Integer                | 100             | Maksymalna liczba żądań na okno (lub pojemność kubelka w przypadku Token Bucket). |
| WindowSec   | Integer                | 60              | Rozmiar okna w sekundach.                                                         |
| Strategia   | TsgcRateLimit-Strategy | rlsToken-Bucket | Algorytm używany dla tego zakresu.                                                |

```
// 1000 requests/day per authenticated user across all their devices
sgcWSRateLimiter1->PerUser->Enabled = true;
sgcWSRateLimiter1->PerUser->Strategy = rlsFixed;
sgcWSRateLimiter1->PerUser->MaxRequests = 1000;
sgcWSRateLimiter1->PerUser->WindowSec = 86400;
// Pass keys like "user:alice@example.com" to Consume / IsAllowed
```

## PerEndpoint



**Opis:** Kolekcja reguł opartych na wzorcach. Każdy element ma własny *Pattern* (obsługiwane symbole wieloznaczne \* i ?), *Strategy*, *MaxRequests* i *WindowSec*. Wygrywa pierwszy włączony element, którego wzorec pasuje do klucza — żadne kolejne elementy nie są oceniane. Ma pierwszeństwo przed PerAPI-Key, PerUser i PerIP.

**Kiedy należy użyć:** Istnieje kosztowny punkt końcowy `/api/v1/expensive-report`, który musi być ograniczony do 10 żądań/min na klienta, podczas gdy tani punkt końcowy statusu `/api/v1/status` zezwala na 1000 żądań/min. Różne punkty końcowe mają różne budżety — należy skonfigurować jeden element `PerEndpoint` na wzorec ścieżki.

| Właściwość | Typ                      | Domyślny | Opis                                                                                                                   |
|------------|--------------------------|----------|------------------------------------------------------------------------------------------------------------------------|
| Włączone   | Boolean                  | False    | Włącza zestaw reguł <code>PerEndpoint</code> . Gdy wartość wynosi <code>False</code> , żadne elementy nie są oceniane. |
| Reguły     | <code>TCollection</code> | puste    | Kolekcja elementów <code>TsgcRateLimitRuleItem</code> . Kolejność ma znaczenie — wygrywa pierwsze dopasowanie.         |

Każdy element `TsgcRateLimitRuleItem` udostępnia:

| Właściwość  | Typ                                | Domyślny                    | Opis                                                                                                    |
|-------------|------------------------------------|-----------------------------|---------------------------------------------------------------------------------------------------------|
| Nazwa       | string                             | "                           | Przyjazna nazwa wyświetlana w inspektorze obiektów.                                                     |
| Włączone    | Boolean                            | True                        | Wyłącza ten element bez jego usuwania.                                                                  |
| Wzorec      | string                             | "                           | Wzorec wieloznaczny dopasowywany do klucza (bez rozróżniania wielkości liter). Obsługuje symbole * i ?. |
| MaxRequests | Integer                            | 100                         | Maksymalna liczba żądań w oknie / pojemność zasobnika.                                                  |
| WindowSec   | Integer                            | 60                          | Rozmiar okna w sekundach.                                                                               |
| Strategia   | <code>TsgcRateLimitStrategy</code> | <code>rlsTokenBucket</code> | Algorytm używany dla dopasowanych żądań.                                                                |

```
sgcWSRateLimiter1->PerEndpoint->Enabled = true;

TsgcRateLimitRuleItem *ruleReport = (TsgcRateLimitRuleItem*)
    sgcWSRateLimiter1->PerEndpoint->Rules->Add();
ruleReport->Name = "expensive-report";
ruleReport->Pattern = "*/api/v1/expensive-report*";
ruleReport->Strategy = rlsSliding;
ruleReport->MaxRequests = 10;
ruleReport->WindowSec = 60;
```

```
TsgcRateLimitRuleItem *ruleStatus = (TsgcRateLimitRuleItem*)
    sgcWSRateLimiter1->PerEndpoint->Rules->Add();
ruleStatus->Name = "status";
ruleStatus->Pattern = "*/api/v1/status*";
ruleStatus->Strategy = rlsTokenBucket;
ruleStatus->MaxRequests = 1000;
ruleStatus->WindowSec = 60;
```

## Limity

**Działanie:** Długoterminowe limity, które obowiązują poza kroczącym / uzupełniającym oknem głównej strategii. Każdy element ogranicza liczbę żądań, które klucz (lub cały system) może wykonać w danym okresie kalendarzowym (*qpHour*, *qpDay* lub *qpMonth*). Po osiągnięciu limitu *Limit* żądanie jest odrzucone i wywoływane jest zdarzenie *OnQuotaExceeded*.

**Kiedy stosować:** Plan bezpłatny z twardym limitem 10 000 żądań miesięcznie. Toczący się zasobnik tokenów nadal wygładza krótkoterminowy ruch, jednak po wyczerpaniu miesięcznego limitu każde kolejne wywołanie kończy się niepowodzeniem aż do początku następnego miesiąca kalendarzowego.

| Właściwość | Typ         | Domyślny | Opis                                                                                                |
|------------|-------------|----------|-----------------------------------------------------------------------------------------------------|
| Włączone   | Boolean     | False    | Włącza zestaw reguł przydziału. Gdy False, żadne elementy nie są oceniane.                          |
| Limity     | TCollection | puste    | Kolekcja obiektów <i>TsgcRateLimitQuotaItem</i> . Wszystkie włączone elementy są oceniane po kolei. |

Każdy obiekt **TsgcRateLimitQuotaItem** udostępnia:

| Właściwość | Typ                      | Domyślny   | Opis                                                                                                                |
|------------|--------------------------|------------|---------------------------------------------------------------------------------------------------------------------|
| Nazwa      | string                   | "          | Przyjazna nazwa zgłaszana w <i>OnQuotaExceeded</i> i w <i>Result.Reason</i> („quota:Name”).                         |
| Włączone   | Boolean                  | True       | Wyłącza ten przydział bez jego usuwania.                                                                            |
| Zakres     | TsgcRateLimitQuotaScope  | qscPer-Key | <i>qscPerKey</i> zlicza niezależnie dla każdego klucza; <i>qscGlobal</i> zlicza łącznie dla wszystkich kluczy.      |
| Okres      | TsgcRateLimitQuotaPeriod | qpHour     | <i>qpHour</i> , <i>qpDay</i> lub <i>qpMonth</i> . Licznik jest automatycznie resetowany na początku każdego okresu. |

|       |       |       |                                                      |
|-------|-------|-------|------------------------------------------------------|
| Limit | Int64 | 10000 | Maksymalna liczba żądań dozwolonych w danym okresie. |
|-------|-------|-------|------------------------------------------------------|

```
sgcWSRateLimiter1->Quotas->Enabled = true;

TsgcRateLimitQuotaItem *quota = (TsgcRateLimitQuotaItem*)
    sgcWSRateLimiter1->Quotas->Quotas->Add();
quota->Name = "free-tier-monthly";
quota->Scope = qscPerKey;
quota->Period = qpMonth;
quota->Limit = 10000;
```

## BurstProtection

**Działanie:** Krótkoterminowy detektor skoków działający równoległe z główną strategią. Jeśli więcej niż *BurstThreshold* żądań nadejdzie z tego samego klucza w ciągu *BurstWindowMs* milisekund, klucz jest umieszczany w trybie oczekiwania na *CooldownSec* sekund, podczas których każde żądanie jest odrzuca-  
ne, nawet jeśli główna strategia by je dopuściła.

**Kiedy stosować:** Wykrywanie klientów-scraperów i skanerów, które zalewają serwer ponad 100 żądania-  
mi w ciągu 100 ms, lecz pozostają poniżej limitu na minutę. Ochrona przed atakami burst wykrywa ten  
maszynowy wzorzec, który nigdy nie jest generowany przez człowieka.

| Właściwość         | Typ          | Domyślny | Opis                                                                                                 |
|--------------------|--------------|----------|------------------------------------------------------------------------------------------------------|
| Włączone           | Boole-<br>an | False    | Włącza detektor skoków.                                                                              |
| BurstThreshold     | Integer      | 50       | Liczba żądań w oknie BurstWindowMs, która wyzwala okres spoczynku.                                   |
| BurstWin-<br>dowMs | Integer      | 500      | Okno obserwacji serii w milisekundach.                                                               |
| CooldownSec        | Integer      | 30       | Czas trwania okresu schładzania po wykryciu serii. Podczas schładzania każde żądanie jest odrzucane. |

```
// More than 50 requests in 500ms triggers a 30-second cooldown
sgcWSRateLimiter1->BurstProtection->Enabled = true;
sgcWSRateLimiter1->BurstProtection->BurstThreshold = 50;
sgcWSRateLimiter1->BurstProtection->BurstWindowMs = 500;
sgcWSRateLimiter1->BurstProtection->CooldownSec = 30;
```

# Odpowiedź

**Działanie:** Konfiguruje kształt odpowiedzi HTTP zwracanej do klientów, gdy żądanie jest ograniczane. Kontroluje kod stanu, treść komunikatu oraz to, czy do odpowiedzi są dodawane nagłówki `Retry-After` i `X-RateLimit-*`.

**Kiedy stosować:** Gdy chcemy, aby dobrze zachowujące się klienci prawidłowo się wycofywały. Należy zwrócić standardowy status 429 z nagłówkiem `Retry-After` wskazującym czas oczekiwania oraz dołączyć `X-RateLimit-Limit` / `X-RateLimit-Remaining` / `X-RateLimit-Reset`, aby klienci mogli proaktywnie dostosowywać tempo zamiast reagować na odrzucenia.

| Właściwość              | Typ     | Domyślny            | Opis                                                                                                                                                                    |
|-------------------------|---------|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StatusCode              | Integer | 429                 | Kod stanu HTTP zwracany dla ograniczonych żądań. Należy używać 429 (standardowy) lub 503 dla odrzucenia w trybie konserwacji.                                           |
| Wiadomość               | string  | "Too Many Requests" | Treść odpowiedzi HTTP dla żądań poddanych ograniczeniu przepustowości.                                                                                                  |
| RetryAfterHeader        | Boolean | True                | Gdy True, dodaje nagłówek <code>Retry-After</code> z obliczonym opóźnieniem ponownej próby w sekundach.                                                                 |
| IncludeRateLimitHeaders | Boolean | True                | Gdy wartość jest True, do każdej odpowiedzi dodawane są nagłówki <code>X-RateLimit-Limit</code> / <code>X-RateLimit-Remaining</code> / <code>X-RateLimit-Reset</code> . |

```
sgcWSRateLimiter1->Response->StatusCode = 429;  
sgcWSRateLimiter1->Response->Message = "Too Many Requests - please slow down";  
sgcWSRateLimiter1->Response->RetryAfterHeader = true;  
sgcWSRateLimiter1->Response->IncludeRateLimitHeaders = true;
```

## Statystyki

**Opis:** Obiekt licznika tylko do odczytu, który udostępnia na bieżąco statystyki ogranicznika szybkości. Wszystkie liczniki są aktualizowane atomowo w wewnętrznej sekcji krytycznej komponentu.

**Kiedy stosować:** Pulpit na żywo w interfejsie administratora. Odpytuj `Stats` z komponentu `TTimer` i wyświetlaj bieżące wartości `TotalRequests` / `TotalThrottled` / `ActiveKeys` w etykietach, wskaźnikach lub wykresach, aby operatorzy mogli obserwować obciążenie ruchem w czasie rzeczywistym.

| Właściwość         | Typ     | Opis                                                                                 |
|--------------------|---------|--------------------------------------------------------------------------------------|
| TotalRequests      | Int64   | Łączna liczba żądań ocenionych od ostatniego Resetu.                                 |
| TotalThrottled     | Int64   | Łączna liczba żądań odrzuconych przez strategię lub ochronę przed seriami.           |
| TotalQuotaExceeded | Int64   | Łączna liczba żądań odrzuconych z powodu osiągnięcia limitu.                         |
| ActiveKeys         | Integer | Liczba aktualnie śledzonych kluczy (zasobniki + liczniki okna + aktywne połączenia). |
| Czas działania     | Int64   | Liczba sekund od ostatniego zresetowania obiektu statystyk.                          |

```
// Poll from a TTimer and update a dashboard
void __fastcall TForm1::StatsTimer(TObject *Sender)
{
    LabelRequests->Caption = "Requests: " + IntToStr(sgcWSRateLimiter1->Stats->TotalRequests);
    LabelThrottled->Caption = "Throttled: " + IntToStr(sgcWSRateLimiter1->Stats->TotalThrottled);
    LabelActiveKeys->Caption = "Active keys: " + IntToStr(sgcWSRateLimiter1->Stats->ActiveKeys);
    LabelUptime->Caption = "Uptime: " + IntToStr(sgcWSRateLimiter1->Stats->Uptime) + " s";
}
```

## Zdarzenia

**OnThrottled:** Wywoływane, gdy żądanie jest odrzucane przez strategię lub ochronę przed skokami. Dostarcza klucz (adres IP, klucz API lub użytkownik) oraz ciąg przyczyny ("token\_bucket", "sliding\_window", "fixed\_window", "burst", "connection\_rate" lub "message\_rate"). Parametr *Allow* (var Boolean) umożliwia nadpisanie decyzji i przepuszczenie żądania.

**OnQuotaExceeded:** Wywoływane po osiągnięciu długoterminowego limitu przydziału. Dostarcza klucz i nazwę przekroczzonego limitu.

**OnStateChange:** Wywoływane, gdy zmienia się rozmiar zasobnika tokenu dla klucza. Udostępnia klucz wraz ze starym i nowym rozmiarem zasobnika. Przydatne do telemetrii, debugowania i wizualizacji w czasie rzeczywistym.

## Metody publiczne

| Metoda | Opis |
|--------|------|
|--------|------|

|                               |                                                                                                                                                                           |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IsAllowed(Key, Cost)          | Zwraca True, jeśli żądanie o podanym koszcie (domyślnie 1) jest dozwolone dla określonego klucza.                                                                         |
| Consume(Key, Cost)            | Próbuje pobrać tokeny kosztów i zwraca TsgcRateLimitResult (Allowed, Remaining, RetryAfterSec, Reason).                                                                   |
| Reset(Key)                    | Czyści liczniki dla podanego klucza.                                                                                                                                      |
| ResetAll                      | Czyści wszystkie wewnętrzne liczniki dla każdego śledzonego klucza i resetuje obiekt Stats.                                                                               |
| GetRemaining(Key)             | Zwraca liczbę żądań wciąż dostępnych dla podanego klucza.                                                                                                                 |
| GetRetryAfter(Key)            | Zwraca liczbę sekund, przez którą wywołujący powinien czekać przed ponowieniem próby.                                                                                     |
| SaveStateToFile(fileName)     | Zapisuje wszystkie zasobniki i liczniki do wskazanego pliku.                                                                                                              |
| LoadStateFromFile(fileName)   | Przywraca zasobniki i liczniki zapisane wcześniej do pliku.                                                                                                               |
| SaveStateToStream(Stream)     | Utrwala stan wewnętrzny do dowolnego strumienia.                                                                                                                          |
| LoadStateFromStream(Stream)   | Przywraca stan wewnętrzny z dowolnego strumienia.                                                                                                                         |
| IsConnectionAllowed(IP)       | Punkt podpięcia integracji z serwerem. Zwraca True, jeśli nowe połączenie z danego adresu IP przechodzi weryfikację limitów żądań. Wywoływane automatycznie przez serwer. |
| IsMessageAllowed(IP, Message) | Hak integracji serwera. Zwraca True, jeśli wiadomość z danego adresu IP przechodzi ograniczenie przepustowości. Wywoływane automatycznie przez serwer.                    |
| RegisterConnection(IP)        | Rejestruje nowe połączenie do śledzenia. Wywoływane automatycznie.                                                                                                        |
| UnregisterConnection(IP)      | Usuwa połączenie ze śledzenia. Wywoływane automatycznie po rozłączeniu.                                                                                                   |

## Automatyczna integracja

Po przypisaniu do właściwości RateLimiter serwera komponent automatycznie:

- **Przy nowym połączeniu:** wywołuje IsConnectionAllowed w celu oceny PerIP, ochrony przed serią żądań i globalnej strategii. Odrzuca połączenie z kodem HTTP 429 przed zdarzeniem OnConnect, jeśli limit został przekroczony. Wywołuje RegisterConnection w celu śledzenia klienta.
- **Po odebraniu wiadomości:** Wywołuje IsMessageAllowed, aby sprawdzić limity PerIP, PerEndpoint, PerAPIKey, PerUser, przydziały i ochronę przed skokowymi wzrostami. Rozłącza lub odrzuca wiadomość po przekroczeniu limitu.
- **Po rozłączeniu:** Wywołuje UnregisterConnection w celu zwolnienia zasobów śledzenia.

## Kompletny przykład

```

// Enable rate limiter
sgcWSRateLimiter1->Enabled = true;

// Global Token Bucket: 50 burst, 10/sec sustained
sgcWSRateLimiter1->TokenBucket->Enabled = true;
sgcWSRateLimiter1->TokenBucket->Capacity = 50;
sgcWSRateLimiter1->TokenBucket->RefillRate = 10;
sgcWSRateLimiter1->TokenBucket->RefillIntervalMs = 1000;

// Per-IP sliding window: 60 requests per minute per IP
sgcWSRateLimiter1->PerIP->Enabled = true;
sgcWSRateLimiter1->PerIP->Strategy = rlsSliding;
sgcWSRateLimiter1->PerIP->MaxRequests = 60;
sgcWSRateLimiter1->PerIP->WindowSec = 60;

// Per-endpoint: expensive report endpoint = 10 req/min
sgcWSRateLimiter1->PerEndpoint->Enabled = true;
TsgcRateLimitRuleItem *rule = (TsgcRateLimitRuleItem*)
    sgcWSRateLimiter1->PerEndpoint->Rules->Add();
rule->Name = "expensive-report";
rule->Pattern = "*/api/v1/expensive-report*";
rule->Strategy = rlsSliding;
rule->MaxRequests = 10;
rule->WindowSec = 60;

// Burst protection: 50 requests in 500ms triggers 30s cooldown
sgcWSRateLimiter1->BurstProtection->Enabled = true;
sgcWSRateLimiter1->BurstProtection->BurstThreshold = 50;
sgcWSRateLimiter1->BurstProtection->BurstWindowMs = 500;
sgcWSRateLimiter1->BurstProtection->CooldownSec = 30;

// Monthly quota per API key: 10000 requests/month
sgcWSRateLimiter1->Quotas->Enabled = true;
TsgcRateLimitQuotaItem *quota = (TsgcRateLimitQuotaItem*)
    sgcWSRateLimiter1->Quotas->Quotas->Add();
quota->Name = "free-tier-monthly";
quota->Scope = qscPerKey;
quota->Period = qpMonth;
quota->Limit = 10000;

// HTTP 429 response with Retry-After and X-RateLimit-* headers
sgcWSRateLimiter1->Response->StatusCode = 429;
sgcWSRateLimiter1->Response->Message = "Too Many Requests";
sgcWSRateLimiter1->Response->RetryAfterHeader = true;
sgcWSRateLimiter1->Response->IncludeRateLimitHeaders = true;

// Persist state across restarts
sgcWSRateLimiter1->StorageFile = "ratelimit.dat";
sgcWSRateLimiter1->LoadStateFromFile("ratelimit.dat");

// Assign to server
sgcWebSocketHTTPServer1->RateLimiter = sgcWSRateLimiter1;

// Handle throttled events
void __fastcall TForm1::RateLimiterThrottled(TObject *Sender,
    const String aKey, const String aReason, bool &Allow)
{
    Log("Throttled " + aKey + ": " + aReason);
}

// Handle quota exhausted events
void __fastcall TForm1::RateLimiterQuotaExceeded(TObject *Sender,
    const String aKey, const String aQuotaName)
{
    Log("Quota " + aQuotaName + " exceeded for " + aKey);
}

```

}

## Bezpieczeństwo wątkowe

Ogranicznik szybkości jest w pełni bezpieczny wątkowo. Wszystkie publiczne metody używają wewnętrznych sekcji krytycznych i bezpiecznych wątkowo liczników chroniących równoległy dostęp. Komponent może być współdzielony przez wiele instancji serwera i dostępny z dowolnego wątku (procedury obsługi zdarzeń serwera, wątki timerów, procesy działające w tle) bez zewnętrznej synchronizacji.

## Dokumentacja

- Właściwości — 13 właściwości dostępnych w komponencie.
- Metody — 12 metod publicznych.
- Zdarzenia — 3 zdarzenia do obsłużenia.



# TsgcWSCircuitBreaker

TsgcWSCircuitBreaker implementuje wzorzec odporności Circuit Breaker dla wywołań po stronie klienta do HTTP API.

## Wprowadzenie

TsgcWSCircuitBreaker implementuje wzorzec odporności Circuit Breaker dla wywołań po stronie klienta do API HTTP. Zapobiega kaskadowym awariom, gdy zdalny punkt końcowy staje się niedostępny, poprzez śledzenie niepowodzeń w oknie kroczącym, otwieranie obwodu po osiągnięciu konfigurowalnych progów i natychmiastowe odrzucanie wywołań w okresie ochładzania. Obsługuje progi liczby niepowodzeń i wskaźnika niepowodzeń, wykrywanie wolnych wywołań, próbne wywołania w stanie półotwartym, klasyfikację wyjątków (rejestruj/ignoruj), zastąpienia per-punkt końcowy, odpowiedzi zastępcze i metryki w czasie rzeczywistym.

Bezpiecznik automatycznie integruje się z dowolnym potomkiem klasy **TsgcHTTPAPI\_client** (OpenAI, Anthropic, Gemini, Grok, DeepSeek, Mistral, Ollama, Stripe, Binance, Bybit, Coinbase, Kraken i inne opakowania API) poprzez właściwość **CircuitBreaker**. Po przypisaniu każde wywołanie Get / Post / Put / Patch / Delete jest automatycznie opakowywane. Dla każdej unikalnej nazwy hosta utrzymywany jest oddzielny obwód.

## Szybki start

1. Umieść komponent **TsgcWSCircuitBreaker** na formularzu.
2. Skonfiguruj progi (kiedy otwierać) i mechanizm odzyskiwania (jak odzyskiwać):

```
sgcWSCircuitBreaker1.Thresholds.FailureCount := 5;  
sgcWSCircuitBreaker1.Recovery.CooldownSec := 30;
```

3. Przypisz circuit breaker do klienta HTTP API (dowolny potomek TsgcHTTPAPI\_client):

```
sgcAI_OpenAI1.CircuitBreaker := sgcWSCircuitBreaker1;
```

4. Wywołaj Get / Post / Delete / Patch / Put na kliencie. Wyłącznik automatycznie rejestruje sukcesy i błędy oraz odrzuca nowe wywołania, gdy obwód jest w stanie otwartym.

## Stany

**Co robi:** Wyłącznik jest maszyną o trzech stanach. Każdy host (klucz) ma własny niezależny stan, który kontroluje, czy wywołania wychodzące są dozwolone, odrzucane, czy wysyłane jako próbne.

| Stan       | Zachowanie                                                                                                   | Przejście                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| csClosed   | Wszystkie wywołania przechodzą. Błędy są zliczane w ruchomym oknie.                                          | Przechodzi do stanu csOpen po osiągnięciu wartości FailureCount, FailureRatePercent lub SlowCallRatePercent.         |
| csOpen     | Każde wywołanie jest natychmiast odrzucane. Dla każdej próby wywoływane jest zdarzenie OnCallRejected.       | Przechodzi do stanu csHalfOpen po upływie liczby sekund zdefiniowanej w Recovery.CooldownSec od momentu otwarcia.    |
| csHalfOpen | Dozwolona jest maksymalnie liczba Recovery.HalfOpenTrialCalls wywołań próbnych w celu zbadania odzyskiwania. | Przechodzi do csClosed, gdy próby kończą się powodzeniem; wraca do csOpen, gdy którakolwiek próba się nie powiedzie. |

## Właściwości najwyższego poziomu

**Działanie:** Główny przełącznik komponentu oraz dwie nazwy kluczy używane, gdy nie zostanie podany klucz dla konkretnego wywołania. Integracja po stronie klienta automatycznie używa docelowej nazwy hosta jako klucza — właściwości DefaultKey i ServerKey mają zastosowanie wyłącznie do ręcznych wywołań Execute/RecordSuccess/RecordFailure oraz do opcjonalnych procedur pomocniczych integracji z serwerem.

**Kiedy używać:** Wyłącz wyłącznik w czasie wykonywania podczas kontrolowanego testu obciążenia lub okna konserwacyjnego bez odłączania go od klienta HTTP. Ustaw sensowną wartość DefaultKey (na przykład 'openai-api'), gdy wywołujesz Execute ręcznie wewnątrz niestandardowego kodu.

| Właściwość  | Typ     | Domyslny  | Opis                                                                                                                                               |
|-------------|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączono    | Boolean | True      | Główny przełącznik. Gdy ma wartość False, każda metoda publiczna jest natychmiast zwracana jako „dozwolona” i żadne liczniki nie są aktualizowane. |
| Default-Key | string  | 'default' | Klucz używany przez bezparametrowe przeciążenia metod Execute, RecordSuccess i RecordFailure.                                                      |
| Server-Key  | string  | 'server'  | Klucz używany przez opcjonalne hooki integracji z serwerem (IsConnectionAllowed, IsMessageAllowed, RecordMessageError).                            |

```
sgcWSCircuitBreaker1->Enabled = true;  
sgcWSCircuitBreaker1->DefaultKey = "openai-api";  
sgcWSCircuitBreaker1->ServerKey = "server";
```

## Progi

**Działanie:** Definiuje warunki przenoszące obwód ze stanu Closed do Open. Bezpiecznik otwiera się, gdy zostanie osiągnięta liczba surowych błędów FailureCount, lub gdy odsetek nieudanych wywołań w oknie przekroczy FailureRatePercent (po MinCalls), lub gdy odsetek powolnych wywołań przekraczających SlowCallDurationMs wyniesie więcej niż SlowCallRatePercent.

**Kiedy stosować:** Twój serwer wywołuje API OpenAI. Gdy w ciągu 60 sekund wystąpi 5 błędów (lub 50% wywołań zakończy się niepowodzeniem po zaobserwowaniu co najmniej 10 wywołań), bezpiecznik otwiera się i zatrzymuje bombardowanie OpenAI — pozwala zdalnej usłudze na odzyskanie zamiast piętrzenia ponownych prób.

| Właściwość          | Typ     | Domyślny | Opis                                                                                                               |
|---------------------|---------|----------|--------------------------------------------------------------------------------------------------------------------|
| Włączone            | Boolean | True     | Włącza otwieranie obwodu na podstawie progu. Gdy ustawione na False, obwód nigdy nie otwiera się automatycznie.    |
| FailureCount        | Integer | 5        | Bezwzględna liczba błędów w oknie kroczącym powodująca wyzwolenie wyłącznika.                                      |
| FailureRatePercent  | Integer | 50       | Procent awarii (0-100), który wyzwala wyłącznik po osiągnięciu MinCalls.                                           |
| SlowCallDurationMs  | Integer | 3000     | Próg (w ms) powyżej którego pomyślne wywołanie jest również zaliczane jako "wolne".                                |
| SlowCallRatePercent | Integer | 0        | Procent (0–100) wolnych wywołań, który wyzwala wyłącznik. Wartość 0 wyłącza wykrywanie wolnych wywołań.            |
| MinCalls            | Integer | 10       | Minimalna liczba wywołań zaobserwowana przed oceną progów szybkości (pozwala uniknąć wyzwalań na małych próbkach). |

```
// Open after 5 failures, or 50% failure rate on at least 10 calls  
sgcWSCircuitBreaker1->Thresholds->Enabled = true;  
sgcWSCircuitBreaker1->Thresholds->FailureCount = 5;  
sgcWSCircuitBreaker1->Thresholds->FailureRatePercent = 50;  
sgcWSCircuitBreaker1->Thresholds->MinCalls = 10;  
// Also treat any call slower than 3 seconds as a "slow" call  
sgcWSCircuitBreaker1->Thresholds->SlowCallDurationMs = 3000;
```

```
sgcWSCircuitBreaker1->Thresholds->SlowCallRatePercent = 60;
```

## TimeWindow

**Co robi:** Definiuje ruchome okno używane do oceny progów. Okno jest podzielone na podzakresy BucketCount, dzięki czemu stare liczniki stopniowo wygasają w miarę upływu czasu. Progi są zawsze oceniane względem sumy aktywnych zasobników.

**Kiedy stosować:** Należy użyć 30-sekundowego okna kroczącego z 6 zasobnikami, aby chwilowe skoki nie wyzwalaly wyłącznika, ale trwałe błędy już tak. Dłuższe okna wygładzają szum; krótsze szybciej reagują na awarie. Więcej zasobników zapewnia płynniejsze starzenie, ale zużywa nieco więcej pamięci na hosta.

| Właściwość       | Typ     | Domyślny | Opis                                                                                                                               |
|------------------|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| RollingWindowSec | Integer | 60       | Całkowita szerokość okna kroczącego w sekundach. Starsze próbki są odrzucane.                                                      |
| BucketCount      | Integer | 10       | Liczba pod-przedziałów, na które podzielone jest okno. Każdy przedział przechowuje liczniki sukcesów, błędów i wolnych odpowiedzi. |

```
// 30-second window split into 6 buckets (5 seconds each)
sgcWSCircuitBreaker1->TimeWindow->RollingWindowSec = 30;
sgcWSCircuitBreaker1->TimeWindow->BucketCount = 6;
```

## Odzyskiwanie

**Opis:** Steruje sposobem odzyskiwania obwodu ze stanu Open. Po upływie czasu CooldownSec obwód przechodzi do stanu HalfOpen i przepuszcza HalfOpenTrialCalls próbnych wywołań. Jeżeli próby zakończą się sukcesem, obwód zostaje zamknięty; jeżeli zakończą się niepowodzeniem, obwód ponownie otwiera się i timer odliczania jest resetowany.

**Kiedy używać:** Po otwarciu, odczekaj 30 sekund, a następnie przepuść jedno próbne wywołanie w celu przetestowania odzyskania przed ponownym zamknięciem. Dłuższy czas oczekiwania chroni przeciążone

usługi nadrzędne przed kolejną falą ruchu; krótki czas oczekiwania szybciej przywraca dostęp do usług, które tylko chwilowo tracą stabilność.

| Właściwość          | Typ     | Domyślny | Opis                                                                                                                                    |
|---------------------|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Cooldown-Sec        | Integer | 30       | Liczba sekund oczekiwania w stanie Open przed przejściem do stanu HalfOpen.                                                             |
| HalfOpen-TrialCalls | Integer | 1        | Liczba próbnych wywołań dozwolonych w stanie HalfOpen przed podjęciem decyzji o zamknięciu lub ponownym otwarciu obwodu.                |
| AutoReset           | Boolean | True     | Gdy True, liczniki są czyszczone automatycznie podczas zmian stanu.                                                                     |
| MaxRetries          | Integer | 3        | Informacyjny pułap zużywany przez pomocnicze mechanizmy ponawiania. Wyłącznik sam w sobie nie wykonuje żadnych automatycznych ponowień. |

```
// After opening: wait 30s, then allow 1 trial call, then decide
sgcWSCircuitBreaker1->Recovery->CooldownSec = 30;
sgcWSCircuitBreaker1->Recovery->HalfOpenTrialCalls = 1;
sgcWSCircuitBreaker1->Recovery->AutoReset = true;
sgcWSCircuitBreaker1->Recovery->MaxRetries = 3;
```

## Fallback

**Działanie:** Konfiguruje alternatywny łańdunek serwowany w czasie, gdy układ jest otwarty. Gdy właściwość `Fallback.Enabled` ma wartość `True` i wywołanie `Execute / ExecuteWithResult` zostaje odrzucone, wywoływane jest zdarzenie `OnFallback` z odpowiedzią zainicjowaną z `CachedResponse` (lub `CustomMessage`). Procedury obsługi mogą zastąpić ten ciąg przed zwróceniem go do wywołującego.

**Kiedy stosować:** Zwraca zapisaną w pamięci podręcznej ostatnią poprawną odpowiedź, gdy OpenAI jest niedostępny, dzięki czemu użytkownicy widzą ograniczone, ale funkcjonalne zachowanie (np. poprzednio wygenerowane podsumowanie) zamiast błędu. Dla statycznej odpowiedzi w trybie konserwacji należy użyć właściwości `CustomMessage` z treścią JSON.

| Właściwość | Typ     | Domyślny | Opis                                                                                                                                             |
|------------|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączone   | Boolean | False    | Włącza ścieżkę zapasową. Gdy ma wartość <code>False</code> , odrzucone wywołania są zwracane bez wywoływania zdarzenia <code>OnFallback</code> . |

|                 |         |       |                                                                                                             |
|-----------------|---------|-------|-------------------------------------------------------------------------------------------------------------|
| CachedResponse  | string  | "     | Statyczny ładunek zwracany, gdy obwód jest w stanie Open. Ma pierwszeństwo przed CustomMessage.             |
| Custom-Message  | string  | "     | Zastępczy ładunek używany, gdy CachedResponse jest pusty.                                                   |
| UseLast-Success | Boolean | False | Gdy wartość wynosi True, ostatnia pomyślna odpowiedź zaobserwowana dla klucza jest serwowana jako fallback. |

```
// When api.openai.com is down, serve a safe JSON body instead of erroring
sgcWSCircuitBreaker1->Fallback->Enabled = true;
sgcWSCircuitBreaker1->Fallback->CachedResponse =
    "{\"error\":\"service unavailable\",\"retry_after\":30}";
sgcWSCircuitBreaker1->Fallback->CustomMessage =
    "{\"error\":\"ai service temporarily offline\"}";
sgcWSCircuitBreaker1->Fallback->UseLastSuccess = true;
```

## Klasyfikacja

**Działanie:** Klasyfikuje zgłaszane wyjątki jako „zarejestrowane jako błąd” lub „zignorowane”. Tylko zarejestrowane wyjątki zwiększają liczniki błędów. MatchMode określa sposób porównywania ciągów znaków w RecordAsFailure / IgnoreExceptions z tekstem wyjątku (ClassName: Message).

**Kiedy stosować:** Traktuj wyjątki EidHTTPProtocolException z kodami 500/502/503/504 jako błędy, ale ignoruj wyjątki EidHTTPProtocolException z kodami 400/401/404 (błędy biznesowe nie powinny wyzwalać obwodu). Pozostawienie parametru RecordAsFailure pustego oznacza, że każdy nieignorowany wyjątek jest rejestrowany jako błąd.

| Właściwość        | Typ                               | Domyslny     | Opis                                                                                                                                                         |
|-------------------|-----------------------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RecordAs-Failure  | TStringList                       | puste        | Wzorce dopasowywane względem tekstu wyjątku; pasujące wyjątki są liczone jako awarie. Pusta lista oznacza rejestrowanie wszystkiego, co nie jest ignorowane. |
| Ignore-Exceptions | TStringList                       | puste        | Wzorce dopasowywane do tekstu wyjątku; pasujące wyjątki są pomijane (nie są klasyfikowane ani jako sukces, ani jako błąd).                                   |
| MatchMode         | TsgcCircuitBreakerException-Match | cem-Contains | Tryb porównania: cemExact, cemContains, cemStartsWith lub cemWildcard (* / ?).                                                                               |

```
// Record HTTP 5xx and timeouts as failures; ignore client-side 4xx
sgcWSCircuitBreaker1->Classification->MatchMode = cemContains;
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("HTTP/1.1 500");
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("HTTP/1.1 502");
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("HTTP/1.1 503");
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("HTTP/1.1 504");
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("timeout");
sgcWSCircuitBreaker1->Classification->IgnoreExceptions->Add("HTTP/1.1 400");
sgcWSCircuitBreaker1->Classification->IgnoreExceptions->Add("HTTP/1.1 401");
sgcWSCircuitBreaker1->Classification->IgnoreExceptions->Add("HTTP/1.1 404");
```

## PerEndpoint

**Działanie:** Kolekcja zastąpień opartych na wzorcach. Każdy element `TsgcCircuitBreakerEndpointItem` ma własny wzorec (obsługiwane symbole wieloznaczne \* i ?), blok `OverrideThresholds` i blok `OverrideRecovery`. Gdy aktywny klucz (nazwa hosta) pasuje do włączonego elementu, ustawienia tego elementu zastępują globalne ustawienia `Thresholds` / `Recovery` na czas trwania wywołania.

**Kiedy stosować:** Oddzielne obwody na API: `api.openai.com` ma łagodne progi (od czasu do czasu zwalnia pod obciążeniem), ale `api.stripe.com` jest krytyczny dla płatności i powinien otwierać się szybciej, aby nie przepuszczać awarii. Należy dodać jeden element punktu końcowego na host z dostosowanymi progami.

| Właściwość | Typ         | Domyślny | Opis                                                                                                              |
|------------|-------------|----------|-------------------------------------------------------------------------------------------------------------------|
| Włączone   | Boolean     | False    | Włącza kolekcję nadpisywania punktów końcowych. Gdy ma wartość False, żadne elementy nie są oceniane.             |
| Elementy   | TCollection | puste    | Kolekcja obiektów <code>TsgcCircuitBreakerEndpointItem</code> . Pierwsze pasujące i włączone dopasowanie wygrywa. |

Każdy `TsgcCircuitBreakerEndpointItem` udostępnia:

| Właściwość         | Typ                          | Domyślny | Opis                                                                                                                       |
|--------------------|------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------|
| Wzorec             | string                       | "        | Wzorec wieloznaczny dopasowywany do klucza (nazwy hosta). Obsługuje * i ?.                                                 |
| Włączone           | Boolean                      | True     | Wyłącza ten element bez jego usuwania.                                                                                     |
| OverrideThresholds | TsgcCircuitBreakerThresholds | defaults | Progi stosowane w przypadku dopasowania wzorca. Aktywne tylko wtedy, gdy własne pole <code>Enabled</code> ma wartość True. |

|                  |                            |          |                                                          |
|------------------|----------------------------|----------|----------------------------------------------------------|
| OverrideRecovery | TsgcCircuitBreakerRecovery | defaults | Ustawienia odzyskiwania stosowane po dopasowaniu wzorca. |
|------------------|----------------------------|----------|----------------------------------------------------------|

```
sgcWSCircuitBreaker1->PerEndpoint->Enabled = true;

// OpenAI - lenient: 10 failures, 60s cooldown
TsgcCircuitBreakerEndpointItem *oOpenAI = (TsgcCircuitBreakerEndpointItem*)
    sgcWSCircuitBreaker1->PerEndpoint->Items->Add();
oOpenAI->Pattern = "api.openai.com";
oOpenAI->Enabled = true;
oOpenAI->OverrideThresholds->Enabled = true;
oOpenAI->OverrideThresholds->FailureCount = 10;
oOpenAI->OverrideThresholds->FailureRatePercent = 70;
oOpenAI->OverrideRecovery->CooldownSec = 60;

// Stripe - payment critical: open fast, recover slow
TsgcCircuitBreakerEndpointItem *oStripe = (TsgcCircuitBreakerEndpointItem*)
    sgcWSCircuitBreaker1->PerEndpoint->Items->Add();
oStripe->Pattern = "api.stripe.com";
oStripe->Enabled = true;
oStripe->OverrideThresholds->Enabled = true;
oStripe->OverrideThresholds->FailureCount = 3;
oStripe->OverrideThresholds->FailureRatePercent = 30;
oStripe->OverrideRecovery->CooldownSec = 120;
```

## Metryki

**Działanie:** Liczniki tylko do odczytu, aktualizowane atomowo w miarę przepływu wywołań przez wyłącznik. Agregowane dla wszystkich kluczy / nazw hostów. CurrentOpenBreakers jest odświeżany po każdej zmianie stanu; AverageLatencyMs jest bieżącą średnią czasów trwania zaobserwowanych przez Execute.

**Kiedy używać:** Pulpit na żywo pokazujący stany obwodów dla każdego hosta, współczynniki awarii i liczniki odrzuconych żądań. Należy cyklicznie odpytywać właściwość Metrics za pomocą TTimer, aby wyrysować wskaźniki dla operatorów, lub przesyłać dane do Prometheus / StatsD w celach alertowych.

| Właściwość          | Typ     | Opis                                                                     |
|---------------------|---------|--------------------------------------------------------------------------|
| TotalCalls          | Int64   | Łączna liczba przetworzonych wywołań (sukcesy + zarejestrowane błędy).   |
| TotalFailures       | Int64   | Łączna liczba błędów zarejestrowanych dla wszystkich kluczy.             |
| TotalSuccesses      | Int64   | Łączna liczba pomyślnych wywołań zarejestrowanych dla wszystkich kluczy. |
| TotalRejected       | Int64   | Łączna liczba wywołań odrzuconych z powodu otwartego stanu obwodu.       |
| CurrentOpenBreakers | Integer | Liczba kluczy aktualnie w stanie Otwartym.                               |



|                  |             |                                                                                        |
|------------------|-------------|----------------------------------------------------------------------------------------|
| AverageLatencyMs | Do-<br>uble | Bieżąca średnia opóźnień (ms) mierzona wokół wywołań Execute / Execute-<br>WithResult. |
|------------------|-------------|----------------------------------------------------------------------------------------|

```
// Poll from a TTimer and update a dashboard
void __fastcall TForm1::MetricsTimer(TObject *Sender)
{
    LabelCalls->Caption = "Calls: " +
        IntToStr(sgcWSCircuitBreaker1->Metrics->TotalCalls) + " (ok " +
        IntToStr(sgcWSCircuitBreaker1->Metrics->TotalSuccesses) + " / fail " +
        IntToStr(sgcWSCircuitBreaker1->Metrics->TotalFailures) + " / rejected " +
        IntToStr(sgcWSCircuitBreaker1->Metrics->TotalRejected) + ")";
    LabelOpen->Caption = "Open breakers: " +
        IntToStr(sgcWSCircuitBreaker1->Metrics->CurrentOpenBreakers);
    LabelLatency->Caption = "Avg latency: " +
        FormatFloat("0.0", sgcWSCircuitBreaker1->Metrics->AverageLatencyMs) + " ms";
}
```

## Zdarzenia

**OnStateChange:** Wywoływane, gdy wyłącznik przechodzi między stanami. Przekazuje klucz, poprzedni stan i nowy stan (csClosed, csOpen lub csHalfOpen). Przydatne do rejestrowania, alertowania i aktualizowania interfejsu użytkownika na bieżąco.

**OnCallRejected:** Wywoływane, gdy wywołanie zostaje odrzucone z powodu otwarcia obwodu lub wyczerpania budżetu prób HalfOpen. Dostarcza klucz oraz ciąg przyczyny (na przykład Circuit Open).

**OnFallback:** Wywoływane tuż przed zwróceniem odpowiedzi zastępczej. Parametr *aResponse* (var string) jest inicjowany wartością Fallback.CachedResponse / CustomMessage i może zostać zastąpiony przez procedurę obsługi.

**OnFailureRecorded:** Wywoływane za każdym razem, gdy zostanie zarejestrowana awaria. Dostarcza klucz oraz tekst wyjątku (ClassName: Message). Należy użyć do strukturyzowanego rejestrowania lub korelowania z zewnętrznym śledzeniem incydentów.

**OnSlowCall:** Wywoływane, gdy pomyślne wywołanie przekroczyło Thresholds.SlowCallDurationMs. Dostarcza klucz oraz zmierzony czas trwania w milisekundach.

## Metody publiczne

| Metoda                                     | Opis                                                                                                                                                                                              |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Execute(Key, Action)                       | Uruchamia TProc pod podanym kluczem. Najpierw sprawdza IsCallAllowed, a następnie automatycznie rejestruje sukces, niepowodzenie lub wolne wykonanie i zwraca True, jeśli akcja została wykonana. |
| Execute(Action)                            | Przeciążenie używające DefaultKey.                                                                                                                                                                |
| ExecuteWithResult(Key, Action, out Result) | (D2009+) Uruchamia TFunc<TObject> i zwraca jego wynik przez parametr out; ta sama księgowość co w metodzie Execute.                                                                               |
| RecordSuccess(Key)                         | Rejestruje pomyślne wywołanie. Przesuwa stan z HalfOpen do Closed po udanych próbach.                                                                                                             |
| RecordSuccess                              | Przeciążenie używające DefaultKey.                                                                                                                                                                |
| RecordFailure(Key, Exception)              | Rejestruje błąd z opcjonalnym tekstem wyjątku; wyzwala ocenę przejścia.                                                                                                                           |
| RecordFailure(Exception)                   | Przeciążenie używające DefaultKey.                                                                                                                                                                |
| ForceOpen(Key)                             | Ręcznie wymusza przejście obwodu w stan otwarty (Open).                                                                                                                                           |
| ForceClose(Key)                            | Ręcznie wymusza przejście obwodu w stan zamknięty (Closed).                                                                                                                                       |
| Reset(Key)                                 | Czyści liczniki i stan dla podanego klucza.                                                                                                                                                       |
| ResetAll                                   | Czyści wszystkie śledzone obwody i resetuje metryki.                                                                                                                                              |
| GetState(Key)                              | Zwraca aktualny stan TsgcCircuitState dla danego klucza.                                                                                                                                          |
| IsCallAllowed(Key)                         | Zwraca True, jeśli nowe wywołanie jest aktualnie dozwolone. Wywoływane automatycznie przez klienta HTTP API przed każdym żądaniem.                                                                |
| GetFailureRate(Key)                        | Zwraca bieżący procent awarii (0,0 - 100,0) obserwowany w oknie kroczącym.                                                                                                                        |
| GetTotalCalls(Key)                         | Zwraca liczbę wywołań zaobserwowanych w oknie czasowym dla danego klucza.                                                                                                                         |
| GetLastStateChange(Key)                    | Zwraca wartość TDateTime ostatniej zmiany stanu.                                                                                                                                                  |
| GetOpenBreakers                            | Zwraca TArray<string> kluczy aktualnie w stanie Open.                                                                                                                                             |
| SaveStateToFile(FileName)                  | Zapisuje stan i liczniki każdego obwodu do pliku.                                                                                                                                                 |
| LoadStateFromFile(FileName)                | Przywraca stany obwodów z wcześniej zapisanego pliku.                                                                                                                                             |
| IsConnectionAllowed(IP)                    | Hook samoochrony po stronie serwera. Pomocniczy względem głównego zastosowania po stronie klienta. Zwraca True, jeśli obwód ServerKey nie jest w stanie Open.                                     |
| IsMessageAllowed(IP, Message)              | Hook samoochrony po stronie serwera. Zwraca True, gdy obwód ServerKey nie jest otwarty.                                                                                                           |
| RegisterConnection(IP)                     | Przechwytywacz po stronie serwera zarezerwowany do przyszłych metryk per-IP (obecnie brak operacji).                                                                                              |

|                                   |                                                                                                      |
|-----------------------------------|------------------------------------------------------------------------------------------------------|
| UnregisterConnection(IP)          | Przechwytywacz po stronie serwera zarezerwowany do przyszłych metryk per-IP (obecnie brak operacji). |
| RecordMessageError(IP, Exception) | Punkt integracji zabezpieczeń po stronie serwera. Rejestruje błąd dla ServerKey.                     |
| RecordMessageSuccess(IP)          | Hak samoochrony po stronie serwera. Rejestruje sukces na ServerKey.                                  |

## Automatyczna integracja

Po przypisaniu do obiektu potomnego `TsgcHTTPAPI_client` za pomocą właściwości `CircuitBreaker`, klient automatycznie:

- **Przed każdym wywołaniem Get / Post / Put / Patch / Delete:** Wyznacza klucz na podstawie nazwy hosta docelowego URL (na przykład `api.openai.com`) i wywołuje `IsCallAllowed(host)`. Jeśli obwód jest otwarty, wywołanie jest odrzucane z wyjątkiem *"Circuit breaker open for <host> - call rejected"*, a zdarzenie `OnCallRejected` jest wywoływane.
- **Po pomyślnej odpowiedzi HTTP:** Wywołuje `RecordSuccess(host)` w celu aktualizacji liczników, co umożliwia próbom `HalfOpen` zamknięcie obwodu po odzyskaniu sprawności przez zdalny serwer.
- **W przypadku wyjątku:** Wywołuje `RecordFailure(host, exception.Message)`, stosuje reguły klasyfikacji, aby zdecydować czy zliczyć czy zignorować, ocenia przejścia stanu, a następnie ponownie zgłasza wyjątek do wywołującego.
- **Jeden obwód na hosta:** Wywołania do różnych hostów (na przykład `api.openai.com`, `api.anthropic.com`, `api.stripe.com`) śledzą w pełni niezależne obwody.

## Kompletny przykład

```
// Enable the circuit breaker
sgcWSCircuitBreaker1->Enabled = true;

// Thresholds: open after 5 failures, or 50% failure rate on 10+ calls
sgcWSCircuitBreaker1->Thresholds->Enabled = true;
sgcWSCircuitBreaker1->Thresholds->FailureCount = 5;
sgcWSCircuitBreaker1->Thresholds->FailureRatePercent = 50;
sgcWSCircuitBreaker1->Thresholds->SlowCallDurationMs = 3000;
sgcWSCircuitBreaker1->Thresholds->MinCalls = 10;

// Rolling window: 60 seconds / 10 buckets
sgcWSCircuitBreaker1->TimeWindow->RollingWindowSec = 60;
sgcWSCircuitBreaker1->TimeWindow->BucketCount = 10;

// Recovery: wait 30s in Open, then 1 trial call in HalfOpen
sgcWSCircuitBreaker1->Recovery->CooldownSec = 30;
sgcWSCircuitBreaker1->Recovery->HalfOpenTrialCalls = 1;

// Classification: record HTTP 5xx and timeouts, ignore 4xx
sgcWSCircuitBreaker1->Classification->MatchMode = cemContains;
sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("HTTP/1.1 5");
```

```

sgcWSCircuitBreaker1->Classification->RecordAsFailure->Add("timeout");
sgcWSCircuitBreaker1->Classification->IgnoreExceptions->Add("HTTP/1.1 404");

// Fallback: serve a cached payload while the circuit is Open
sgcWSCircuitBreaker1->Fallback->Enabled = true;
sgcWSCircuitBreaker1->Fallback->CustomMessage =
    "{ \"error\": \"service unavailable\" }";

// Per-endpoint overrides
sgcWSCircuitBreaker1->PerEndpoint->Enabled = true;
TsgcCircuitBreakerEndpointItem *oStripe = (TsgcCircuitBreakerEndpointItem*)
    sgcWSCircuitBreaker1->PerEndpoint->Items->Add();
oStripe->Pattern = "api.stripe.com";
oStripe->OverrideThresholds->Enabled = true;
oStripe->OverrideThresholds->FailureCount = 3;
oStripe->OverrideRecovery->CooldownSec = 120;

// Persist circuit state across restarts
sgcWSCircuitBreaker1->LoadStateFromFile("circuit.dat");

// Attach to an HTTP API client (any TsgcHTTAPI_client descendant)
sgcAI_OpenAI1->CircuitBreaker = sgcWSCircuitBreaker1;

// Use the client normally; the breaker intercepts Get / Post / etc.
try
{
    String Response = sgcAI_OpenAI1->Get("https://api.openai.com/v1/models");
}
catch (Exception &E)
{
    if (E.Message.Pos("Circuit breaker open") > 0)
        Log("Circuit open for api.openai.com - using fallback");
    else
        Log("Call failed: " + E.Message);
}

// Handle state changes
void __fastcall TForm1::CircuitStateChange(TObject *Sender,
    const String aKey, TsgcCircuitState aOldState, TsgcCircuitState aNewState)
{
    Log("Circuit " + aKey + ": " + IntToStr((int)aOldState) + " -> " +
        IntToStr((int)aNewState));
}

// Handle fallback responses
void __fastcall TForm1::CircuitFallback(TObject *Sender,
    const String aKey, String &aResponse)
{
    Log("Fallback used for " + aKey);
}

```

## Bezpieczeństwo wątkowe

Przerywnik obwodu jest w pełni bezpieczny wątkowo. Wszystkie metody publiczne używają wewnętrznych sekcji krytycznych i bezpiecznych wątkowo list w celu ochrony współbieżnego dostępu do stanu poszczególnych kluczy, okna przesuwne i liczników metryk. Jeden komponent może być współdzielony

przez wielu klientów HTTP API i dostępny z dowolnego wątku (procedury obsługi zdarzeń, timery, procesy działające w tle) bez zewnętrznej synchronizacji.

## **Dokumentacja**

- Właściwości — 10 właściwości dostępnych na komponencie.
- Metody — 17 metod publicznych.
- Zdarzenia — 5 zdarzeń do obsługi.

# TsgcWSAPIKeyManager

TsgcWSAPIKeyManager zapewnia pełne zarządzanie cyklem życia kluczy API wydawanych przez serwery sgcWebSockets.

## Wprowadzenie

TsgcWSAPIKeyManager zapewnia pełne zarządzanie cyklem życia kluczy API wydawanych przez serwery sgcWebSockets. Generuje kryptograficznie losowe klucze z konfigurowalnym, czytelnym dla człowieka prefiksem, zapisuje ich skróty przy użyciu SHA-256, SHA-512 lub Bcrypt, weryfikuje przychodzące żądania przez nagłówek lub ciąg zapytania, rotuje klucze z konfigurowalnym okresem przejściowym, unieważnia i odnawia klucze, wymusza autoryzację opartą na zakresach, prowadzi trwały dziennik audytu oraz obsługuje opcjonalną listę dozwolonych adresów IP. Klucze są przechowywane w pamięci lub w zaszyfrowanym pliku, który przeżywa restarty.

Menedżer dołącza do TsgcWebSocketHTTPServer i TsgcWSServer\_HTTPAPI za pośrednictwem właściwości **APIKeyManager**. Ponieważ automatyczne haki serwera widzą jedynie adres IP węzła, właściwe uwierzytelnianie żądań odbywa się przez wywołanie **IsRequestAuthorized** ze zdarzenia OnConnect serwera, co zapewnia dostęp do pełnego bloku nagłówków HTTP i ciągu URL/zapytania, umożliwiając parsowanie, walidację i ewentualnie sprawdzenie zakresu skonfigurowanego nagłówka **X-API-Key** (lub parametru zapytania).

## Szybki start

1. Upuścić komponent **TsgcWSAPIKeyManager** na formularz.
2. Skonfiguruj generowanie (przedrostek, długość), haszowanie (algorytm, sól) i walidację (nazwa nagłówka, lista dozwolonych adresów IP, FailClosed). Zapoznaj się z dokumentacją poszczególnych sekcji poniżej.
3. Przypisz menedżera do komponentu serwera:

```
sgcWebSocketHTTPServer1.APIKeyManager := sgcWSAPIKeyManager1;
```

4. Połącz zdarzenie OnConnect serwera z wywołaniem **IsRequestAuthorized** przy użyciu nagłówków, URL i IP połączenia. Odłącz klienta, gdy metoda zwróci False.

5. Wystawić pierwszy klucz dla klienta i dostarczyć mu wartość w postaci jawnej dokładnie jeden raz — w pamięci pozostaje wyłącznie postać skrótowa.

## Właściwości najwyższego poziomu

**Co robi:** Główny przełącznik całego komponentu. Gdy `Enabled` ma wartość `False`, każda metoda publiczna natychmiast zwraca "dozwolone", a automatyczne przechwytywanie serwera staje się operacją bez efektu, co w praktyce pomija menedżera kluczy.

**Kiedy stosować:** Tymczasowo wyłączyć egzekwowanie kluczy podczas kontrolowanego okna konserwacji lub wdrożenia w środowisku testowym bez odłączania komponentu od serwera. Włącz je ponownie po zakończeniu zmian — żadne klucze nie są tracone, żadne zakresy nie ulegają zmianie, a dziennik audytu wznowia działanie w momencie przywrócenia.

| Właściwość | Typ     | Domyślny | Opis                                                                                                                                                                                                                      |
|------------|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączone   | Boolean | True     | Główny przełącznik. Gdy ustawiony na <code>False</code> , funkcje <code>ValidateKey</code> / <code>IsRequestAuthorized</code> zawsze zwracają <code>True</code> , a automatyczne wpięcia serwera stają się nieoperacyjne. |

```
sgcWSAPIKeyManager1->Enabled = true;
```

## Generowanie

**Działanie:** Kontroluje sposób budowania ciągu klucza w postaci jawnej przez `IssueKey`. Klucz składa się z `KeyPrefix` i `KeyLength` losowych znaków pobranych z `Charset`. Gdy `IncludeChecksum` wynosi `True`, dołączana jest krótka suma kontrolna, dzięki czemu zniekształcone klucze można tanio odrzucić przed przeszukiwaniem skrótów.

**Kiedy stosować:** Markowe klucze dla oprogramowania SaaS — należy używać `sgc_live_aB3...` w środowisku produkcyjnym i `sgc_test_...` w klastrze testowym. Prefiks ułatwia pracownikom pomocy technicznej identyfikację środowiska, do którego należy ujawniony klucz, oraz automatyczne usuwanie go przez narzędzia czyszczące logi.

| Właściwość       | Typ     | Domyślny  | Opis                                                                                                                                                                      |
|------------------|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Włączone         | Boolean | True      | Gdy wartość wynosi False, metoda IssueKey zgłasza wyjątek zamiast generować nowy klucz.                                                                                   |
| KeyPrefix        | string  | "         | Czytelny dla człowieka prefiks dołączany przed losową częścią treści (na przykład sgc_live_).                                                                             |
| KeyLength        | Integer | 32        | Liczba losowych znaków w treści klucza.                                                                                                                                   |
| Zestaw znaków    | string  | A-Za-z0-9 | Alfabet używany do generowania losowej treści. Domyślnie stosowane są alfanumeryczne znaki bezpieczne dla URL, jeśli pole jest puste.                                     |
| Include-Checksum | Boolean | False     | Gdy wartość wynosi True, dodawany jest krótki sufiks sumy kontrolnej, dzięki czemu oczywiście zniekształcone klucze mogą być odrzucane bez przeszukiwania tabeli skrótów. |

```
// Branded production keys: sgc_live_aB3...
sgcWSAPIKeyManager1->Generation->Enabled = true;
sgcWSAPIKeyManager1->Generation->KeyPrefix = "sgc_live_";
sgcWSAPIKeyManager1->Generation->KeyLength = 32;
sgcWSAPIKeyManager1->Generation->IncludeChecksum = true;
```

## Storage

**Opis działania:** Kontroluje miejsce przechowywania haszowanych kluczy i dziennika audytu. Wartość *kstMemory* przechowuje wszystko w pamięci procesu (czyszczone po zamknięciu). Wartość *kstFile* zapisuje dane do pliku *FileName*, opcjonalnie zaszyfrowanego kluczem *EncryptionKey*, z automatycznym zapisem co *AutoSaveSeconds* sekund. Wartość *kstCustom* umożliwia implementację własnych procedur *SaveToFile* / *LoadFromFile*.

**Kiedy stosować:** Przechowuj klucze w pamięci podczas programowania dla szybkiej iteracji. Przełącz na *kstFile* z *EncryptAtRest := True* w środowisku produkcyjnym, aby klucze przetrwały restart serwera i nigdy nie były przechowywane jako tekst jawny na dysku.

| Właściwość   | Typ                    | Domyślny  | Opis                                                                                                                         |
|--------------|------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------|
| Storage-Type | TsgcAPIKey-StorageType | kstMemory | kstMemory (ulotna), kstFile (utrwalanie w FileName) lub kstCustom (dostarczane przez użytkownika SaveToFile / LoadFromFile). |
| FileName     | string                 | "         | Ścieżka pliku używana przez SaveToFile / LoadFromFile, gdy StorageType ma wartość kstFile.                                   |



|                 |         |       |                                                                                                                       |
|-----------------|---------|-------|-----------------------------------------------------------------------------------------------------------------------|
| EncryptAtRest   | Boolean | False | Gdy True, szyfruje magazyn kluczy na dysku przy użyciu EncryptionKey.                                                 |
| EncryptionKey   | string  | "     | Sekret używany do szyfrowania utwalonego pliku kluczy. Musi być podany, gdy właściwość EncryptAtRest ma wartość True. |
| AutoSaveSeconds | Integer | 0     | Gdy wartość jest większa od 0, menedżer co AutoSaveSeconds sekund ponownie zapisuje magazyn kluczy na dysk.           |

```
// Production: encrypted file that survives restarts, auto-saved every 60s
sgcWSAPIKeyManager1->Storage->StorageType = kstFile;
sgcWSAPIKeyManager1->Storage->FileName = "apikeys.dat";
sgcWSAPIKeyManager1->Storage->EncryptAtRest = true;
sgcWSAPIKeyManager1->Storage->EncryptionKey = "my-very-secret-master-key";
sgcWSAPIKeyManager1->Storage->AutoSaveSeconds = 60;
sgcWSAPIKeyManager1->LoadFromFile("apikeys.dat");
```

## Haszowanie

**Działanie:** Klucze w postaci zwykłego tekstu nigdy nie są utrwalane. IssueKey tworzy skrót wygenerowanego klucza przy użyciu algorytmu *Algorithm* z opcjonalną wartością *Salt* i liczbą iteracji *Iterations*. ValidateKey ponownie tworzy skrót przychodzącego klucza i wykonuje porównanie w stałym czasie względem przechowywanego skrótu.

**Kiedy stosować:** Nigdy nie przechowuj danych w postaci jawnej. SHA-256 to rozsądny domyślny wybór dla większości wdrożeń. Stosuj SHA-512 dla profili o wyższym poziomie bezpieczeństwa (FIPS 140-2 Level 3). Stosuj Bcrypt, gdy potrzebne jest celowe rozciąganie klucza — wolniejsza walidacja, ale znacznie trudniejsze ataki brute-force offline na ujawniony magazyn kluczy.

| Właściwość | Typ                     | Domyślny   | Opis                                                                                                      |
|------------|-------------------------|------------|-----------------------------------------------------------------------------------------------------------|
| Algorytm   | TsgcAPIKeyHashAlgorithm | kha-SHA256 | Funkcja skrótu używana do przechowywania danych w spoczynku: khaSHA256, khaSHA512 lub khaBcrypt.          |
| Salt       | string                  | "          | Statyczna sól mieszana z każdym hashem. Zmiana powoduje unieważnienie wszystkich istniejących kluczy.     |
| Iteracje   | Integer                 | 1          | Liczba rund mieszania. Zwiększenie tej wartości poprawia rozciąganie klucza kosztem opóźnienia walidacji. |

```
// SHA-512 with a static salt and 10k iterations for key stretching
sgcWSAPIKeyManager1->Hashing->Algorithm = khaSHA512;
```

```
sgcWSAPIKeyManager1->Hashing->Salt = "my-salt";
sgcWSAPIKeyManager1->Hashing->Iterations = 10000;
```

## Zakresy

**Opis działania:** Katalog dozwolonych ciągów zakresu. Gdy właściwość *Enabled* ma wartość *True*, każdy zakres dołączony do wystawionego klucza musi istnieć na tej liście; nieznane zakresy są odrzucane. Po szczególne wpisy katalogu mogą być dezaktywowane bez ich usuwania. Metody *ValidateKey* / *IsRequestAuthorized* mogą wymagać określonego zakresu dla każdego żądania.

**Kiedy stosować:** Precyzyjne uprawnienia. Klucz jednego klienta zawiera wyłącznie *read:orders* i zostanie odrzucony na punkcie końcowym zapisu. Integracja partnerska otrzymuje *read:orders* + *write:shipments*, ale nigdy *admin:\**. Każde żądanie dostarcza wskazówkę dotyczącą zakresu, a menedżer konsekwentnie ją egzekwuje.

| Właściwość | Typ         | Domyślny | Opis                                                                                               |
|------------|-------------|----------|----------------------------------------------------------------------------------------------------|
| Włączone   | Boolean     | True     | Włącza wymuszanie zakresu uprawnień. Gdy ustawiono <i>False</i> , parametry zakresu są ignorowane. |
| Zakresy    | TCollection | puste    | Kolekcja wpisów <i>TsgcAPIKeyScopeItem</i> definiujących dozwolone zakresy.                        |

Każdy element **TsgcAPIKeyScopeItem** udostępnia:

| Właściwość | Typ     | Domyślny | Opis                                                                                                  |
|------------|---------|----------|-------------------------------------------------------------------------------------------------------|
| Nazwa      | string  | "        | Identyfikator zakresu (na przykład <i>read:orders</i> ). Uwzględniana jest wielkość liter.            |
| Opis       | string  | "        | Czytelny dla człowieka opis wyświetlany w inspektorze obiektów i interfejsach administratora.         |
| Włączone   | Boolean | True     | Wyłącza ten zakres bez jego usuwania. Wyłączone zakresy nigdy nie spełniają warunku <i>HasScope</i> . |

```
sgcWSAPIKeyManager1->Scopes->Enabled = true;
```

```
TsgcAPIKeyScopeItem *scopeRead = (TsgcAPIKeyScopeItem*)
    sgcWSAPIKeyManager1->Scopes->Scopes->Add();
scopeRead->Name = "read:orders";
scopeRead->Description = "List and read order history";
```

```
TsgcAPIKeyScopeItem *scopeWrite = (TsgcAPIKeyScopeItem*)
    sgcWSAPIKeyManager1->Scopes->Scopes->Add();
scopeWrite->Name = "write:shipments";
scopeWrite->Description = "Create and update shipment records";
```

## Rotacja

**Co robi:** Sprawia, że rotacja kluczy jest przezroczysta. RotateKey wydaje nowy klucz dla tego samego właściciela i zakresów oraz oznacza stary klucz jako *kksRotated*. W oknie *GracePeriodSec* oba klucze są poprawne, dzięki czemu klient może dokonać zmiany bez przestojów. Gdy *AutoRotateDays* jest > 0, klucze starsze niż ten próg są rotowane przez czyszczenie w tle; *NotifyBeforeExpirySec* wywołuje OnKeyExpired wcześniej, aby można było wysłać powiadomienie e-mail do klienta.

**Kiedy stosować:** Klient zgłasza wyciek klucza, ale integracja nadal działa. Należy obrócić klucz: klient otrzymuje nowy, stary pozostaje aktywny przez 24 godziny, aby można go było wymienić w całej infrastrukturze, a następnie jest automatycznie unieważniany.

| Właściwość            | Typ     | Domyślny | Opis                                                                                                                   |
|-----------------------|---------|----------|------------------------------------------------------------------------------------------------------------------------|
| Włączone              | Boolean | False    | Włącza funkcje rotacji (okres przejściowy i automatyczne czyszczenie).                                                 |
| GracePeriodSec        | Integer | 86400    | Liczba sekund, przez którą poprzedni klucz pozostaje ważny po wykonaniu RotateKey. Wartość domyślna wynosi 24 godziny. |
| AutoRotateDays        | Integer | 0        | Gdy wartość jest większa od 0, automatyczne czyszczenie w tle rotuje klucze starsze niż podana liczba dni.             |
| NotifyBeforeExpirySec | Integer | 604800   | Wywołuje OnKeyExpired tyle sekund przed faktycznym wygaśnięciem klucza. Domyślnie wynosi 7 dni.                        |

```
// 24h grace period, auto-rotate keys older than 90 days, notify 7 days ahead
sgcWSAPIKeyManager1->Rotation->Enabled = true;
sgcWSAPIKeyManager1->Rotation->GracePeriodSec = 86400;
sgcWSAPIKeyManager1->Rotation->AutoRotateDays = 90;
sgcWSAPIKeyManager1->Rotation->NotifyBeforeExpirySec = 7 * 86400;
```

## Wygaśnięcie

**Działanie:** Kontroluje sposób wygasania wydanych kluczy. *DefaultTTLSec* jest stosowany do *IssueKey*, gdy wywołujący przekazuje 0 jako *ExpiresInSec*. *EnforceExpiry* przełącza, czy wygasłe klucze są odrzucane. *CheckIntervalSec* kontroluje, jak często skanowanie w tle sprawdza wygasłe wpisy, wywołuje *OnKeyExpired* i aktualizuje *Stats*.

**Kiedy używać:** Klucze integracji partnerów muszą wygasać po 90 dniach — partnerzy muszą ponownie podpisać umowę i pobrać nowy klucz. Ustawienie *DefaultTTLSec* na  $90 \times 86400$  oznacza, że każde wywołanie *IssueKey* respektuje tę politykę, chyba że wywołujący jawnie ją nadpisze.

| Właściwość       | Typ     | Domyślny | Opis                                                                                                          |
|------------------|---------|----------|---------------------------------------------------------------------------------------------------------------|
| DefaultTTL-Sec   | Integer | 0        | Domyślny czas życia nowo wystawionych kluczy w sekundach. Wartość 0 oznacza brak wygasania.                   |
| EnforceExpiry    | Boolean | True     | Gdy wartość wynosi True, <i>ValidateKey</i> odrzuca klucze, których <i>ExpiresAt</i> jest w przeszłości.      |
| CheckIntervalSec | Integer | 60       | Częstotliwość, z jaką cykliczne skanowanie sprawdza wygasłe klucze i wywołuje zdarzenie <i>OnKeyExpired</i> . |

```
// Partner keys expire after 90 days, enforced strictly, swept every minute
sgcWSAPIKeyManager1->Expiration->DefaultTTLSec = 90 * 86400;
sgcWSAPIKeyManager1->Expiration->EnforceExpiry = true;
sgcWSAPIKeyManager1->Expiration->CheckIntervalSec = 60;
```

## RateLimit

**Działanie:** Przechowuje domyślne limity szybkości dla poszczególnych kluczy (maksymalna liczba żądań na okno) wraz z każdym kluczem. Wartości te są wskazówkami informacyjnymi, które instancja *TsgcWSRateLimiter* może odczytać przez zakres *PerAPIKey* w celu stosowania ograniczania przepustowości odpowiedniego dla danego poziomu — menedżer kluczy API sam nie ogranicza ruchu.

**Kiedy stosować:** SaaS z poziomami cenowymi. Klienci poziomu Free mają limit 100 żądań/godzinę, poziomu Pro — 10 000 żądań/godzinę, poziomu Enterprise — 1 000 000 żądań/godzinę. Skonfiguruj *DefaultMaxRequests* / *DefaultWindowSec* w menedżerze kluczy API i sparuj z *TsgcWSRateLimiter.PerAPIKey*, aby limiter odczytywał budżet per-klucz automatycznie.

| Właściwość | Typ | Domyślny | Opis |
|------------|-----|----------|------|
|------------|-----|----------|------|

|                    |          |       |                                                                                        |
|--------------------|----------|-------|----------------------------------------------------------------------------------------|
| Włączone           | Boole-an | False | Umożliwia dodawanie metadanych limitu szybkości na klucz obok każdego wydanego klucza. |
| DefaultMaxRequests | Integer  | 100   | Domyślna maksymalna liczba żądań w oknie przypisywana nowym kluczom.                   |
| DefaultWindowSec   | Integer  | 60    | Domyślny rozmiar okna w sekundach powiązany z DefaultMaxRequests.                      |

```
// Default tier: 100 requests per hour per API key
sgcWSAPIKeyManager1->RateLimit->Enabled = true;
sgcWSAPIKeyManager1->RateLimit->DefaultMaxRequests = 100;
sgcWSAPIKeyManager1->RateLimit->DefaultWindowSec = 3600;
// Pair with TsgcWSRateLimiter.PerAPIKey for actual throttling
```

## Audyt

**Opis działania:** rejestruje każdą akcję związaną z cyklem życia klucza (wydanie, walidacja, odwołanie, rotacja, wygaśnięcie, przyznanie/odebranie zakresu, nieudana walidacja) w pamięciowym buforze pierścieniowym oraz, gdy ustawiona jest właściwość *LogFile*, dołącza te same wpisy do pliku. *IncludeIP* rejestruje adres IP wnioskodawcy; *IncludePayload* przechwytuje dodatkowe szczegóły. *RetentionDays* usuwa starsze wpisy; *MaxMemoryEntries* ogranicza bufor pierścieniowy.

**Kiedy stosować:** Wymóg zgodności: każda operacja na kluczu musi być rejestrowana i przechowywana przez 12 miesięcy. Należy włączyć audyt z parametrem *LogFile* wskazującym na katalog docelowy SIEM i ustawić *RetentionDays* na 365. Zdarzenie *OnAuditEvent* jest wywoływane dla każdego wpisu, dzięki czemu można je również przesyłać do magistrali komunikatów.

| Właściwość      | Typ      | Domyślny | Opis                                                                                                                              |
|-----------------|----------|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| Włączone        | Boole-an | True     | Włącza dziennik audytu. Gdy wartość jest False, żadne wpisy nie są rejestrowane, a <i>OnAuditEvent</i> nigdy nie jest wywoływane. |
| LogFile         | string   | "        | Opcjonalna ścieżka pliku, do którego dołączany jest dziennik audytu.                                                              |
| IncludeIP       | Boole-an | True     | Gdy wartość jest True, każdy wpis rejestruje adres IP żądającego.                                                                 |
| Include-Payload | Boole-an | False    | Gdy wartość wynosi True, pole <i>Details</i> każdego wpisu audytu zawiera dodatkowy kontekst.                                     |
| Retention-Days  | Integer  | 90       | Wpisy starsze niż podana liczba dni są usuwane przez zadanie czyszczące działające w tle.                                         |

|                  |         |       |                                                                                |
|------------------|---------|-------|--------------------------------------------------------------------------------|
| MaxMemoryEntries | Integer | 10000 | Maksymalna liczba wpisów przechowywanych w pamięciowym buforze pierścieniowym. |
|------------------|---------|-------|--------------------------------------------------------------------------------|

```
// 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;
```

## Walidacja

**Co robi:** Informuje menedżera, jak klienci przedstawiają swoje klucze i przy jakich warunkach żądania są akceptowane. *HeaderName* (domyślnie X-API-Key) i *QueryParamName* (domyślnie api\_key) to dwa miejsca, w których szukają ExtractKeyFromHeaders / ExtractKeyFromQuery. *RequireHTTPS* odrzuca ruch nieszyfrowany. *IPAllowlist* ogranicza klucze do określonych źródłowych adresów IP. *FailClosed* (domyślnie True) odrzuca każde żądanie, którego menedżer nie może pozytywnie autoryzować.

**Kiedy używać:** Korporacyjny API: wymagaj HTTPS dla każdego żądania i ogranicz klucze do firmowego zakresu CIDR, aby nawet wyciekły klucz nie mógł być nadużyty spoza sieci korporacyjnej.

| Właściwość     | Typ         | Domyślny  | Opis                                                                                                         |
|----------------|-------------|-----------|--------------------------------------------------------------------------------------------------------------|
| HeaderName     | string      | X-API-Key | Nazwa nagłówka HTTP używana do odczytu przychodzącego klucza.                                                |
| QueryParamName | string      | api_key   | Parametr ciągu zapytania używany jako zapasowy, gdy nagłówek jest nieobecny.                                 |
| RequireHTTPS   | Boolean     | False     | Gdy ustawione na True, odrzuca każde żądanie inne niż HTTPS, niezależnie od ważności klucza.                 |
| IPAllowlist    | TStringList | puste     | Jeśli wartość jest niepusta, autoryzowane są wyłącznie żądania, których adres IP znajduje się na liście.     |
| FailClosed     | Boolean     | True      | Gdy wartość wynosi True, każde żądanie, którego menedżer nie może jednoznacznie autoryzować, jest odrzucane. |

```
// Corporate API: HTTPS-only, allow only office egress IPs, fail closed
sgcWSAPIKeyManager1->Validation->HeaderName = "X-API-Key";
sgcWSAPIKeyManager1->Validation->QueryParamName = "api_key";
sgcWSAPIKeyManager1->Validation->RequireHTTPS = true;
```

```
sgcWSAPIKeyManager1->Validation->FailClosed = true;  
sgcWSAPIKeyManager1->Validation->IPAllowlist->Add("203.0.113.10");  
sgcWSAPIKeyManager1->Validation->IPAllowlist->Add("203.0.113.11");
```

## Statystyki

**Co robi:** Obiekt licznika tylko do odczytu, który udostępnia aktualne statystyki magazynu kluczy. Wszystkie liczniki są aktualizowane atomowo wewnątrz sekcji krytycznej komponentu.

**Kiedy używać:** Panel administracyjny na żywo. Należy odpytywać Stats za pomocą TTimer i wyświetlać bieżące wartości ActiveKeys / RevokedKeys / TotalValidations / TotalValidationFailures, aby operatorzy mogli monitorować obciążenie uwierzytelniania w czasie rzeczywistym.

| Właściwość              | Typ                 | Opis                                                                         |
|-------------------------|---------------------|------------------------------------------------------------------------------|
| TotalKeys               | Int64               | Łączna liczba kluczy kiedykolwiek wystawionych przez tę instancję menedżera. |
| ActiveKeys              | Int64               | Liczba kluczy aktualnie o statusie kksActive.                                |
| RevokedKeys             | Int64               | Liczba kluczy aktualnie w stanie kksRevoked.                                 |
| ExpiredKeys             | Int64               | Liczba kluczy aktualnie w statusie kksExpired.                               |
| TotalValidations        | Int64               | Łączna liczba wywołań ValidateKey / IsRequestAuthorized.                     |
| TotalValidationFailures | Int64               | Łączna liczba wywołań walidacyjnych, które odrzuciły klucz.                  |
| LastIssuedAt            | TDa-<br>teTi-<br>me | Znacznik czasu ostatniego wywołania IssueKey.                                |

```
// Poll from a TTimer and update a dashboard  
void __fastcall TForm1::StatsTimer(TObject *Sender)  
{  
    LabelActive->Caption = "Active: " + IntToStr(sgcWSAPIKeyManager1->Stats->ActiveKeys);  
    LabelRevoked->Caption = "Revoked: " + IntToStr(sgcWSAPIKeyManager1->Stats->RevokedKeys);  
    LabelValidations->Caption = "Validations: " + IntToStr(sgcWSAPIKeyManager1->Stats->TotalValidations);  
    LabelFailures->Caption = "Failures: " + IntToStr(sgcWSAPIKeyManager1->Stats->TotalValidationFailures);  
}
```

## Zdarzenia

**OnKeyIssued:** Wywoływane po wydaniu nowego klucza. Udostępnia właściciela, surowy klucz w postaci jawnej (ostatnia szansa na jego odczytanie) oraz tablicę zakresów. Z tego procedury obsługi należy przekazać klucz jawny klientowi.

**OnKeyValidated:** Wywoływane za każdym razem, gdy `ValidateKey` zakończy działanie. Dostarcza klucz, flagę logiczną *aValid* oraz ciąg zrozumiałego dla maszyny powodu.

**OnKeyRevoked:** Wywoływane po unieważnieniu klucza. Dostarcza klucz oraz powód unieważnienia.

**OnKeyRotated:** Wywoływane po rotacji klucza. Dostarcza stary klucz i nowy klucz, aby umożliwić powiadomienie klienta.

**OnKeyExpired:** Wywoływane przez działające w tle zadanie czyszczenia wygasłych kluczy, gdy upłynął czas `ExpiresAt` klucza (lub `NotifyBeforeExpirySec` sekund wcześniej).

**OnValidation:** Końcowy hook decyzyjny wywoływany podczas `ValidateKey`. Ustawienie `Allow := False` i podanie *Reason* powoduje odrzucenie w innym przypadku prawidłowego klucza (niestandardowe reguły biznesowe).

**OnAuditEvent:** Wywoływane dla każdego wpisu w dzienniku audytu. Dostarcza rekord `TsgcAPIKeyAuditEntry` (Timestamp, KeyPrefix, Action, IP, Details) — idealne do przekazywania danych do zewnętrznego systemu SIEM.

## Metody publiczne

| Metoda                                             | Opis                                                                                                                                                                                                             |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>IssueKey(Owner, Scopes, ExpiresInSec)</code> | Generuje, haszuje i przechowuje nowy klucz; zwraca tekst jawny (jedyna chwila, gdy można go zobaczyć). Należy przekazać <code>ExpiresInSec = 0</code> , aby odziedziczyć <code>Expiration.DefaultTTLSec</code> . |
| <code>ValidateKey(Key, RequiredScope, IP)</code>   | Weryfikuje nieprzetworzony klucz. Opcjonalnie wymusza wymagany zakres i rejestruje adres IP żądającego. Zwraca <code>True</code> w przypadku powodzenia.                                                         |



|                                                              |                                                                                                                                                     |
|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| RevokeKey(Key, Reason)                                       | Oznacza klucz jako unieważniony; kolejne wywołania ValidateKey zwracają False.                                                                      |
| RotateKey(OldKey, out NewKey)                                | Wydaje nowy klucz dla tego samego właściciela i zakresów; stary klucz jest oznaczany jako kksRotated i honorowany przez czas GracePeriodSec.        |
| RenewKey(Key, NewExpiresInSec)                               | Przedłuża ważność klucza o podaną liczbę sekund od chwili obecnej.                                                                                  |
| GetKeyInfo(Key)                                              | Zwraca rekord TsgcAPIKeyInfo z właścicielem, zakresami, statusem, znacznikami czasu i liczbą uzyć.                                                  |
| GetKeysByOwner(Owner)                                        | Zwraca identyfikatory kluczy skrótowych powiązane z podanym właścicielem.                                                                           |
| ListScopes(Key)                                              | Zwraca zakresy uprawnień przypisane do klucza.                                                                                                      |
| IsExpired(Key)                                               | Zwraca True, jeśli klucz przekroczył datę ważności.                                                                                                 |
| IsRevoked(Key)                                               | Zwraca True, jeśli klucz jest unieważniony.                                                                                                         |
| KeyExists(Key)                                               | Zwraca True, jeśli klucz jest znany (aktywny, unieważniony lub wygasły).                                                                            |
| GrantScope(Key, Scope)                                       | Dodaje zakres do istniejącego klucza.                                                                                                               |
| RevokeScope(Key, Scope)                                      | Usuwa zakres z istniejącego klucza.                                                                                                                 |
| HasScope(Key, Scope)                                         | Zwraca True, jeśli klucz zawiera podany zakres.                                                                                                     |
| GetAuditLog(Key, MaxEntries)                                 | Zwraca wpisy audytu; gdy Key jest puste, zwraca wpisy dla wszystkich kluczy.                                                                        |
| ClearAuditLog(Key)                                           | Czyści dziennik audytu dla określonego klucza lub (gdy pusty) dla wszystkich kluczy.                                                                |
| SaveToFile(FileName)                                         | Zapisuje klucze i dziennik audytu do wskazanego pliku (gdy pole jest puste, używa Storage.FileName).                                                |
| LoadFromFile(FileName)                                       | Przywraca klucze i dziennik audytu z pliku.                                                                                                         |
| ExportKeys(Stream)                                           | Eksport strumieniowy kluczy i dziennika audytu.                                                                                                     |
| ImportKeys(Stream)                                           | Strumieniowy import kluczy i dziennik audytu.                                                                                                       |
| PurgeExpired                                                 | Usuwa wszystkie klucze, których data ważności ExpiresAt już minęła.                                                                                 |
| Liczba                                                       | Zwraca liczbę kluczy aktualnie przechowywanych.                                                                                                     |
| IsRequestAuthorized(Headers, QueryString, IP, RequiredScope) | Jednorazowe sprawdzenie autoryzacji: wyodrębnia klucz z nagłówek lub ciągu zapytania, weryfikuje go i opcjonalnie wymusza zakres.                   |
| ExtractKeyFromHeaders(Headers)                               | Przetwarza wartość Validation.HeaderName z bloku surowego nagłówka HTTP.                                                                            |
| ExtractKeyFromQuery(QueryString)                             | Analizuje wartość Validation.QueryParamName z surowego ciągu zapytania.                                                                             |
| IsConnectionAllowed(IP)                                      | Przechwytywacz serwera: zwraca False, gdy FailClosed ma wartość True, IPAllowlist jest niepusta, a IP nie jest na liście. Wywoływany automatycznie. |

|                               |                                                                                                                      |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------|
| IsMessageAllowed(IP, Message) | Hook serwerowy wywoływany automatycznie dla każdej przychodzącej wiadomości. Domyślnie brak operacji na poziomie IP. |
| RegisterConnection(IP)        | Śledzi nowe połączenie. Wywoływane automatycznie przez serwer.                                                       |
| UnregisterConnection(IP)      | Zwalnia śledzenie połączenia. Wywoływane automatycznie po rozłączeniu.                                               |

## Automatyczna integracja

Po przypisaniu do właściwości APIKeyManager serwera, komponent automatycznie:

- **Przy nowym połączeniu:** Serwer wywołuje metodę IsConnectionAllowed(IP), która wymusza opcjonalną listę dozwolonych adresów IP (IPAllowlist), gdy właściwość FailClosed ma wartość True (w przeciwnym razie operacja jest zerowa). Następnie wywoływana jest metoda RegisterConnection(IP) w celu śledzenia aktywnego połączenia.
- **Przy odbieraniu wiadomości:** serwer wywołuje IsMessageAllowed(IP, Message), co również jest operacją neutralną na poziomie IP, chyba że lista dozwolonych odrzuci adres IP. Te procedury *nie* sprawdzają nagłówków HTTP, więc uwierzytelnianie za pomocą kluczy API nie jest tu wykonywane.
- **Autoryzacja żądania (do podłączenia przez użytkownika):** Wywołaj *IsRequestAuthorized(Connection.Headers.Request.Text, Connection.URL, Connection.IP, Required-Scope)* ze zdarzenia OnConnect serwera. W tym miejscu dostępne są surowe nagłówki HTTP, dlatego tutaj nagłówek X-API-Key jest wyodrębniany, weryfikowany i opcjonalnie sprawdzany pod kątem zakresu. Jeśli funkcja zwróci False, należy rozłączyć połączenie.
- **Strona klienta:** Należy ustawić nagłówek X-API-Key (lub skonfigurowaną nazwę nagłówka HeaderName) ze zdarzenia OnHandshake / OnBeforeConnect klienta, aby serwer mógł go odczytać podczas aktualizacji HTTP.
- **Po rozłączeniu:** Wywoływane jest UnregisterConnection(IP) w celu zwolnienia wewnętrznego śledzenia.

## Kompletny przykład

```
// Enable and configure the API key manager
sgcWSAPIKeyManager1->Enabled = true;

// Branded keys
sgcWSAPIKeyManager1->Generation->KeyPrefix = "sgc_live_";
sgcWSAPIKeyManager1->Generation->KeyLength = 32;

// Persist to an encrypted file that survives restarts
sgcWSAPIKeyManager1->Storage->StorageType = kstFile;
sgcWSAPIKeyManager1->Storage->FileName = "apikeys.dat";
sgcWSAPIKeyManager1->Storage->EncryptAtRest = true;
sgcWSAPIKeyManager1->Storage->EncryptionKey = "master-secret";
sgcWSAPIKeyManager1->Storage->AutoSaveSeconds = 60;
```

```

// SHA-256 with a static salt
sgcWSAPIKeyManager1->Hashing->Algorithm = khaSHA256;
sgcWSAPIKeyManager1->Hashing->Salt = "my-salt";

// Scope catalog
sgcWSAPIKeyManager1->Scopes->Enabled = true;
TsgcAPIKeyScopeItem *scopeRead = (TsgcAPIKeyScopeItem*)
    sgcWSAPIKeyManager1->Scopes->Scopes->Add();
scopeRead->Name = "read:orders";
scopeRead->Description = "Read orders";
TsgcAPIKeyScopeItem *scopeWrite = (TsgcAPIKeyScopeItem*)
    sgcWSAPIKeyManager1->Scopes->Scopes->Add();
scopeWrite->Name = "write:orders";
scopeWrite->Description = "Create orders";

// 90-day default TTL and 24-hour rotation grace period
sgcWSAPIKeyManager1->Expiration->DefaultTTLSec = 90 * 86400;
sgcWSAPIKeyManager1->Rotation->Enabled = true;
sgcWSAPIKeyManager1->Rotation->GracePeriodSec = 86400;

// Full audit log, 12-month retention
sgcWSAPIKeyManager1->Audit->Enabled = true;
sgcWSAPIKeyManager1->Audit->LogFile = "apikeys-audit.log";
sgcWSAPIKeyManager1->Audit->IncludeIP = true;
sgcWSAPIKeyManager1->Audit->RetentionDays = 365;

// Validation: header-based, fail closed
sgcWSAPIKeyManager1->Validation->HeaderName = "X-API-Key";
sgcWSAPIKeyManager1->Validation->QueryParamName = "api_key";
sgcWSAPIKeyManager1->Validation->FailClosed = true;

// Load previously saved keys and attach to server
sgcWSAPIKeyManager1->LoadFromFile("apikeys.dat");
sgcWebSocketHTTPServer1->APIKeyManager = sgcWSAPIKeyManager1;
sgcWebSocketHTTPServer1->Active = true;

// Issue a new key for a customer (plaintext returned only here)
DynamicArray<String> scopes;
scopes.Length = 2;
scopes[0] = "read:orders";
scopes[1] = "write:orders";
String LKey = sgcWSAPIKeyManager1->IssueKey("customer-123", scopes, 30 * 86400);
String LNewKey;
ShowMessage("Deliver to customer: " + LKey);

// Authorize every incoming connection in OnConnect
void __fastcall TForm1::ServerConnect(TsgcWSConnection *Connection)
{
    if (!sgcWSAPIKeyManager1->IsRequestAuthorized(
        Connection->HeadersRequest->Text,
        Connection->URL, Connection->IP, "read:orders"))
        Connection->Disconnect();
}

// Rotate a leaked key (old one valid 24h during grace)
sgcWSAPIKeyManager1->RotateKey(LKey, LNewKey);

// Revoke a key on subscription cancellation
sgcWSAPIKeyManager1->RevokeKey(LNewKey, "customer cancelled subscription");

```

# Bezpieczeństwo wątkowe

Menedżer kluczy API jest w pełni bezpieczny wątkowo. Wszystkie metody publiczne używają wewnętrznych sekcji krytycznych i list bezpiecznych wątkowo w celu ochrony współbieżnego dostępu do magazynu kluczy, indeksu zakresów, indeksu właścicieli i dziennika audytu. Pojedynczy komponent może być współdzielony przez wiele instancji serwera i dostępny z dowolnego wątku (procedury obsługi zdarzeń serwera, wątki timerów, procesy w tle) bez zewnętrznej synchronizacji.

## Dokumentacja

- Właściwości — 10 właściwości dostępnych w komponencie.
- Metody — 26 metod publicznych.
- Zdarzenia — 7 zdarzeń do obsługi.

# TsgcWebSocketLoadBalancerServer

Komponent TsgcWebSocketLoadBalancerServer.

## Wprowadzenie

Komponent **TsgcWebSocketLoadBalancerServer** umożliwia równoważenie obciążenia protokołów **WebSocket** i **HTTP**. W przypadku protokołu WebSocket dystrybuje komunikaty między grupą serwerów i rozdziela połączenia klientów przy użyciu losowej sekwencji lub algorytmu najmniejszej liczby połączeń.

Load Balancer Server dziedziczy wszystkie metody i właściwości z [TsgcWebSocketHTTPServer](#).

## Konfiguracja modułu równoważenia obciążenia

Serwer Load Balancer jest potomkiem [TsgcWebSocketHTTPServer](#); należy zapoznać się z dokumentacją TsgcWebSocketHTTPServer, aby dowiedzieć się, jak go skonfigurować.

Ponadto Load Balancer posiada właściwość LoadBalancer zawierającą następujące właściwości:

- **LoadBalancing:** konfiguracja dystrybucji połączeń
  - **lbRandom:** (domyślnie) za każdym razem, gdy nowy klient żąda połączenia, zwracany jest losowy serwer.
  - **lbConnections:** za każdym razem, gdy nowy klient żąda połączenia, zwracany jest serwer z najmniejszą liczbą połączonych klientów.
- **Protokoły:** skonfiguruj, które protokoły są włączone
  - **WebSocket:** jeśli true, połączenia websocket będą obsługiwane przez serwer Load Balancer.
  - **HTTP:** jeśli true, połączenia http będą obsługiwane przez serwer Load Balancer.

## Konfiguracja serwera zapasowego

Serwery zapasowe (serwery za modułem równoważenia obciążenia) mogą być [TsgcWebSocketServer](#), [TsgcWebSocketHTTPServer](#) lub [serwerem Datasnap](#).

Te serwery mają właściwość **LoadBalancer**, w której można skonfigurować połączenie między serwerem LoadBalancer a serwerami zapasowymi.

- **Enabled:** ustaw na true, jeśli ma być używany jako serwer zapasowy.
- **Host:** host, na którym znajduje się LoadBalancer.
- **Port:** port nasłuchu LoadBalancera.
- **Guid:** unikalny identyfikator tego serwera.

- **Bindings:** publiczne adresy, na które będą przekazywane połączenia. Przykład: jeśli zapasowy serwer WebSocket nasłuchuje na porcie 8000 i ma adres IP 1.1.1.1, należy użyć: ws://1.1.1.1:8000;
- **AutoRegisterBindings:** jeśli włączone, serwer LoadBalancer użyje właściwości Bindings serwera zapasowego do konfiguracji powiązań publicznych.
- **AutoRestart:** w sekundach; jeżeli wartość jest większa od zera, klient modułu równoważenia obciążenia serwera zapasowego włączy wewnętrzny watchdog, który co x sekund sprawdza, czy połączenie jest aktywne, a w przypadku jego zamknięcia próbuje ponownie się połączyć.

## Dokumentacja

- Właściwości — 18 właściwości dostępnych w komponencie.
- Metody — 9 publicznych metod.
- Zdarzenia — 22 zdarzenia do obsłużenia.

# TsgcWebSocketProxyServer

TsgcWebSocketProxyServer implementuje komponent serwera WebSocket, który nasłuchuje połączeń WebSocket od klientów i przekazuje połączenia danych do zwykłego serwera TCP/IP.

## Wprowadzenie

TsgcWebSocketProxyServer implementuje komponent serwera WebSocket, który nasłuchuje połączeń klientów WebSocket i przekazuje połączenia danych do zwykłego serwera TCP/IP. Jest to szczególnie przydatne dla połączeń przeglądarkowych, ponieważ umożliwia przeglądarce wirtualne połączenie z dowolnym serwerem.

## Dokumentacja

- Właściwości — 19 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 22 zdarzenia do obsługi.

# TsgcIWWebSocketClient

TsgcIWWebSocketClient implementuje komponent WebSocket IntraWeb, który może łączyć się z serwerem WebSocket.

## Wprowadzenie

TsgcIWWebSocketClient implementuje komponent WebSocket dla IntraWeb, który może połączyć się z serwerem WebSocket. Aby skonfigurować ten komponent, należy wykonać poniższe kroki:

1. Upuść komponent TsgcIWWebSocketClient na formularzu

2. Ustaw Host i Port (domyślnie 80) w celu połączenia z dostępnym serwerem WebSocket. Można ustawić właściwość URL, a Host, Port, Parameters... zostaną zaktualizowane na podstawie URL. Przykład: wss://127.0.0.1:8080/ws/ da w rezultacie:

```
oClient = new TsgcIWWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClient->TLS = true;
oClient->Options->Parameters = "/ws/";
```

3. Można wybrać, czy ma być używane połączenie TLS (bezpieczne), czy nie; domyślnie jest ono nieaktywne.

4. Ustaw dozwolone transporty.

**WebSockets:** używana będzie standardowa implementacja WebSocket

**Emulation:** jeśli przeglądarka nie obsługuje WebSockets, użyte zostanie pętlowe połączenie zwrotne AJAX

6. Utwórz procedurę asynchroniczną i ustaw właściwość Active := True



## Methods

**Open:** Otwiera połączenie WebSocket.

**Close:** Zamyka połączenie WebSocket.

**WriteData:** wysyła wiadomość do serwera WebSocket.

## Properties

**Connected:** jest zmienną tylko do odczytu i zwraca True, jeśli połączenie jest aktywne, w przeciwnym razie zwraca False.

**JSOpen:** tutaj można umieścić kod JavaScript po stronie klienta wywoływany po otwarciu połączenia.

**JSClose:** tutaj można dołączyć kod JavaScript po stronie klienta wywoływany przy zamknięciu połączenia.

**JSMessages:** tutaj można umieścić kod JavaScript po stronie klienta, który zostanie wykonany, gdy klienci otrzymają wiadomość z serwera. Ciąg wiadomości można uzyskać za pomocą zmiennej JavaScript "text".

**JSError:** tutaj można umieścić kod JavaScript po stronie klienta, gdy zostanie zgłoszony błąd. Komunikat błędu można uzyskać, używając zmiennej Javascript "text".

## Dokumentacja

- Właściwości — 11 właściwości dostępnych w komponencie.
- Metody — 4 metody publiczne.
- Zdarzenia — 5 zdarzeń do obsługi.

# TsgcWSConnection

TsgcWSConnection jest opakowaniem dla połączeń WebSocket po stronie klienta. Dostęp do tego obiektu można uzyskać w zdarzeniach serwera lub klienta.

## Metody

**WriteData:** wysyła wiadomość do klienta.

**Close:** wysyła wiadomość zamknięcia do drugiego peera. Opcjonalnie można podać „CloseCode”. Domyślnie wysyłana jest wartość NORMAL close code. Jeśli zostanie wysłany ujemny kod zamknięcia, powód zamknięcia nie zostanie wysłany.

**Disconnect:** zamknięcie połączenia klienta po stronie serwera. Opcjonalnie można podać „CloseCode”.

**Ping:** wysyła ping do klienta.

**AddTCPEndOfFrame:** jeśli połączenie jest zwykłym TCP, umożliwia ustawienie, które bajty/bajt definiują koniec wiadomości. Wiadomość jest buforowana do momentu pełnego odebrania.

**Subscribed:** zwraca informację, czy połączenie jest subskrybowane na niestandardowym kanale.

**Subscribe:** subskrybuje to połączenie do kanału. Następnie można rozgłaszać wiadomości z komponentu serwera do wszystkich połączeń subskrybujących ten kanał.

**UnSubscribe:** anuluje subskrypcję tego połączenia na kanale.

## Właściwości

**Protocol:** zwraca podprotokół używany w tym połączeniu.

**IP:** zwraca adres IP węzła zdalnego.

**Port:** zwraca port węzła (Peer Port).

**LocalIP:** zwraca adres IP hosta.

**LocalPort:** zwraca port hosta.

**URL:** zwraca adres URL żądany przez klienta.

**Guid:** zwraca identyfikator połączenia.

**HeadersRequest:** zwraca listę nagłówków odebranych w żądaniu.

**HeadersResponse:** zwraca listę nagłówków wysłanych jako odpowiedź.

**RecBytes:** liczba odebranych bajtów.

**SendBytes:** liczba wysłanych bajtów.

**Transport:** zwraca typ transportu połączenia:

**trpRFC6455:** normalne połączenie WebSocket.

**trpHixie76:** połączenie WebSocket przy użyciu wersji roboczej specyfikacji WebSocket.

**trpFlash:** połączenie WebSocket z użyciem Flash jako zapasowego.

**trpSSE:** połączenie Server-Sent Events.

**trpTCP:** zwykłe połączenie TCP.

**TCPEndOfFrameScanBuffer:** umożliwia zdefiniowanie metody wyszukiwania końca wiadomości (jeśli jako transport używany jest trpTCP).

**eofScanNone:** za każdym razem, gdy nadchodzi nowy pakiet, wywoływane jest zdarzenie OnBinary.

**eofScanLatestBytes:** jeśli ostatnie bajty są równe bajtom dodanym metodą AddTCPEndOfFrame, wywoływane jest zdarzenie wiadomości OnBinary; w przeciwnym razie pakiet jest buforowany.

**eofScanAllBytes:** przeszukuje cały pakiet w poszukiwaniu bajtów równych bajtom dodanym metodą AddTCPEndOfFrame. W przypadku znalezienia wywoływane jest zdarzenie OnBinary; w przeciwnym razie pakiet jest buforowany.

**Data:** obiekt danych sesji użytkownika; można tu przekazać obiekt i uzyskiwać do niego dostęp w dowolnym momencie, na przykład: połączenie z bazą danych, właściwości sesji użytkownika itd.

# Protokoły

Używając WebSockets, można implementować podprotokoły, umożliwiając tworzenie niestandardowej komunikacji. **Na przykład** można zaimplementować podprotokół oparty na protokole WebSocket do komunikacji z niestandardową aplikacją używającą wiadomości JSON, a inny podprotokół używający wiadomości XML.

Po otwarciu połączenia po stronie serwera sprawdzany jest protokół podrzędny przesłany przez klienta pod kątem obsługi przez serwer; jeśli protokół nie jest obsługiwany, połączenie jest zamykane. Serwer może implementować kilka protokołów podrzędnych, ale w jednym połączeniu może być używany tylko jeden.

Podprotokoły są bardzo przydatne do tworzenia spersonalizowanych aplikacji i zapewnienia, że wszyscy klienci obsługują ten sam interfejs komunikacyjny.

Mimo że nazwa protokołu jest dowolna, zaleca się używanie unikalnych nazw, takich jak „dataset.esegce.com”

Pakiet sgcWebSockets umożliwia budowanie własnych protokołów oraz korzystanie z wbudowanych protokołów podrzędnych:

**1. Protokół MQTT:** MQTT jest protokołem transportu wiadomości publish/subscribe dla klientów i serwerów. Jest lekki, otwarty, prosty i zaprojektowany tak, aby był łatwy do implementacji.

**2. Protokół AppRTC:** to demonstracyjna aplikacja WebRTC opracowana przez Google i Mozilla umożliwiająca przeglądarkom komunikację ze sobą za pomocą WebRTC API.

**3. Protokół WebRTC:** projekt open source mający na celu umożliwienie komunikacji w czasie rzeczywistym (RTC) w przeglądarkach internetowych.

**4. Pliki protokołu:** zaimplementowany z użyciem wiadomości binarnych; zapewnia obsługę wysyłania plików: rozmiar pakietu, autoryzacja, QoS, potwierdzenie wiadomości i inne.

**5. Protokół SGC:** zaimplementowany przy użyciu wiadomości JSON-RPC 2.0, udostępnia następujące wzorce: RPC, PubSub, wiadomości transakcyjne, potwierdzenia wiadomości i inne.

**6. Protocol Dataset:** dziedziczy po protokole domyślnym, umożliwia wysyłanie zmian w zbiorze danych (nowy rekord, zapis rekordu lub usunięcie rekordu) z serwera do klientów.

**7. Protocol Presence:** umożliwia sprawdzenie, kto subskrybuje kanał; przykłady zastosowania: czat, współpracownicy nad dokumentem, osoby przeglądające tę samą stronę internetową, gracze w tej samej grze...

**8. Protokół WAMP 1.0:** otwarty podprotokół WebSocket zapewniający dwa asynchroniczne wzorce przesyłania wiadomości: RPC i PubSub.

**9. Protokół WAMP 2.0:** otwarty podprotokół WebSocket udostępniający dwa asynchroniczne wzorce przesyłania wiadomości: RPC i PubSub.

**10. Protokół STOMP:** STOMP to prosty (lub strumieniowy) protokół przesyłania wiadomości zorientowany na tekst. STOMP zapewnia interoperacyjny format przesyłania danych, dzięki któremu klienci STOMP mogą komunikować się z dowolnym brokerem wiadomości STOMP, co umożliwia łatwą i powszechną interoperacyjność przesyłania wiadomości między wieloma językami, platformami i brokerami.

**10.1 STOMP dla RabbitMQ:** klient dla brokera RabbitMQ.

**10.2 STOMP dla ActiveMQ:** klient dla brokera ActiveMQ.

**11. Protocol AMQP:** Advanced Message Queuing Protocol (AMQP 0.9.1) jest otwarty standardem protokołu zapewniającym interoperacyjność komunikatów między systemami niezależnie od dostawcy brokera wiadomości lub używanej platformy.

**12. Protokół AMQP1:** Advanced Message Queuing Protocol (AMQP 1.0.0) to otwarty protokół standardowy umożliwiający interoperacyjność przesyłania wiadomości między systemami, niezależnie od producenta brokera wiadomości lub używanej platformy.

**13. Protocol E2EE:** Wiadomości wysyłane przez połączenie WebSocket są szyfrowane kompleksowo na poziomie aplikacji, dzięki czemu tylko komunikujące się strony mogą je odczytać, nawet jeśli serwer WebSocket przekazuje te dane.

Jeśli konieczne jest używanie **więcej niż jednego protokołu przez jedno połączenie** (przykład: może być potrzebne użycie **protokołu domyślnego** do obsługi zdalnych wywołań procedur i **protokołu Dataset** do obsługi połączeń z bazą danych), można przypisać „Broker” do każdego komponentu protokołu, a

wszystkie wiadomości będą wymieniane przez ten pośredni protokół (można sprawdzić „Tickets Demo” aby zobaczyć prosty przykład).

Protokoły można rejestrować w czasie **wykonywania**. Wystarczy wywołać metodę **RegisterProtocol** i przekazać komponent protokołu jako parametr.

## **Dokumentacja JavaScript**

Więcej informacji na temat wspólnej biblioteki JavaScript używanej w sgcWebSockets można znaleźć [tu-taj](#).

# Protokoły JavaScript

Domyślnie sgcWebSockets korzysta z pliku JavaScript **sgcWebSocket.js**.

Poniżej można znaleźć dostępne metody; zmienne `{%host%}` i `{%port%}` należy zastąpić odpowiednimi wartościami. Przykład: jeśli serwer sgcWebSocket nasłuchuje na porcie 80 pod adresem `www.example.com`, dostęp do pliku `sgcWebSocket.js` należy skonfigurować jako:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
```

## Otwórz połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');
</script>
```

sgcWebSocket posiada 3 parametry, z których tylko pierwszy jest wymagany:

```
sgcWebSocket(url, protocol, transport)
```

- **URL:** lokalizacja serwera WebSocket; można użyć "ws:" dla zwykłych połączeń WebSocket i "wss:" dla zabezpieczonych połączeń WebSocket.

```
sgcWebSocket('ws://127.0.0.1')
```

```
sgcWebSocket('wss://127.0.0.1')
```

- **Protocol:** jeśli serwer akceptuje jeden lub więcej protokołów, można zdefiniować, który protokół ma być używany.

```
sgcWebSocket('ws://127.0.0.1', 'esegece.com')
```



- **Transport:** domyślnie najpierw próbuje nawiązać połączenie WebSocket, a jeśli przeglądarka go nie obsługuje, próbuje użyć Server Sent Events jako transportu.

Użyj WebSocket, jeśli jest zaimplementowany, jeśli nie, użyj Server Sent Events:

```
sgcWebSocket('ws://127.0.0.1')
```

Używaj wyłącznie WebSocket jako transportu:

```
sgcWebSocket('ws://127.0.0.1', '', ['websocket'])
```

Używać wyłącznie transportu Server Sent Events:

```
sgcWebSocket('ws://127.0.0.1', '', ['sse'])
```

## Otwórz połączenie z uwierzytelnianiem

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  var socket = new sgcWebSocket({"host":"ws://{%host%}:{%port%}", "user":"admin", "password":"1234"});
</script>
```

## Wyślij wiadomość

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');
  socket.send('Hello sgcWebSockets!');
</script>
```

## Wyświetl alert z odebraną wiadomością

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');
  socket.on('message', function(event)
  {
    alert(event.message);
  });
</script>
```

```
}  
</script>
```

## Odebrano wiadomość binarną

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>  
<script>  
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');  
  socket.on('stream', function(event)  
  {  
    document.getElementById('image').src = URL.createObjectURL(event.stream);  
    event.stream = "";  
  }  
</script>
```

## Odebrana wiadomość binarna (nagłówek + obraz)

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>  
<script>  
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');  
  socket.on('stream', function(event)  
  {  
    sgcWSStreamRead(evt.stream, function(header, stream) {  
      document.getElementById('text').innerHTML = header;  
      document.getElementById('image').src = URL.createObjectURL(event.stream);  
      event.stream = "";  
    }  
  }  
</script>
```

## Wyświetl alert dla zdarzeń OnConnect, OnDisconnect i OnError

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>  
<script>  
  var socket = new sgcWebSocket('ws://{%host%}:{%port%}');  
  socket.on('open', function(event)  
  {  
    alert('sgcWebSocket Open!');  
  }  
  );  
  socket.on('close', function(event)  
  {  
    alert('sgcWebSocket Closed!');  
  }  
  );  
  socket.on('error', function(event)  
  {  
    alert('sgcWebSocket Error: ' + event.message);  
  }  
  );  
</script>
```

## Zamknij połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  socket.close();
</script>
```

## Pobierz status połączenia

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script>
  socket.state();
</script>
```

# Protokół MQTT

MQTT to protokół transportu wiadomości oparty na modelu publikuj/subskrybuj (klient-serwer). Jest lekki, otwarty, prosty i zaprojektowany z myślą o łatwości implementacji. Cechy te sprawiają, że jest idealny do zastosowań w wielu sytuacjach, w tym w środowiskach o ograniczonych zasobach, takich jak komunikacja M2M (Machine to Machine) i Internet rzeczy (IoT), gdzie wymagana jest mała zajętość kodu lub sieć o ograniczonej przepustowości.

Protokół działa na TCP/IP lub innych protokołach sieciowych zapewniających uporządkowane, niezawodne i dwukierunkowe połączenia. Jego funkcje obejmują:

- Zastosowanie wzorca wiadomości publikuj/subskrybuj, zapewniającego dystrybucję wiadomości jeden-do-wielu i rozdzielenie aplikacji.
- Transport wiadomości niezależny od zawartości ładunku.
- Trzy poziomy jakości usług dla dostarczania wiadomości:
  - "Co najwyżej raz", gdzie wiadomości są dostarczane zgodnie z możliwościami środowiska operacyjnego. Może wystąpić utrata wiadomości. Ten poziom mógłby być używany na przykład z danymi z czujników otoczenia, gdzie nie ma znaczenia, jeśli pojedynczy odczyt zostanie utracony, ponieważ następny zostanie opublikowany wkrótce potem.
  - „Co najmniej raz”, gdzie wiadomości mają gwarancję dotarcia, lecz mogą wystąpić duplikaty.
  - "Dokładnie raz", gdzie dostarczenie wiadomości jest gwarantowane dokładnie jeden raz. Ten poziom może być używany na przykład w systemach rozliczeniowych, gdzie zduplikowane lub utracone wiadomości mogłyby prowadzić do naliczenia nieprawidłowych opłat.
- Minimalny narzut transportowy i wymiana protokołów zminimalizowane w celu redukcji ruchu sieciowego.
- Mechanizm powiadamiania zainteresowanych stron o wystąpieniu nieprawidłowego rozłączenia.

## Funkcje

- Obsługuje wersje MQTT **3.1.1** i **5.0**.
- **Wzorzec publikowania/subskrypcji** zapewniający dystrybucję wiadomości jeden-do-wielu i rozdzielenie aplikacji.
- **Potwierdzenie** wysłanych wiadomości.
- Implementuje **QoS** (Quality of Service) dla dostarczania wiadomości (wszystkie poziomy: Co najwyżej raz, Co najmniej raz i Dokładnie raz)
- **Last Will Testament**.
- **Bezpieczne** połączenia.
- **HeartBeat** i **Watchdog**.
- **Uwierzytelnianie** do serwera.

## Komponenty

**TsgcWSPClient\_MQTT**: Komponent klienta MQTT.

## Najczęstsze zastosowania

- **Połączenie**
  - [Połączenie klienta MQTT](#)
  - [Połącz z serwerami Mosquitto MQTT](#)
  - [Sesje klienta MQTT](#)
  - [Wersja MQTT klienta](#)
- **Publish & Subscribe**
  - [MQTT Publish Subscribe](#)
  - [Tematy MQTT](#)
  - [Subskrypcja MQTT](#)
  - [MQTT Publish Message](#)
  - [Odbieranie wiadomości MQTT](#)
  - [MQTT Publikuj i czekaj na odpowiedź](#)
- **Inne**
  - [MQTT Wyczyść zachowane wiadomości](#)

# TsgcWSPClient\_MQTT

Składnik MQTT dostarcza lekką, w pełni funkcjonalną implementację klienta MQTT z obsługą wersji 3.

## Wprowadzenie

Komponent MQTT zapewnia lekką, w pełni funkcjonalną implementację klienta MQTT z obsługą wersji 3.1.1 i 5.0. Komponent obsługuje połączenia niezaszyfrowane i bezpieczne zarówno przez standardowe TCP, jak i WebSocket.

Połączenie z serwerem MQTT jest proste. Należy upuścić ten komponent na formularzu i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client. Należy ustawić host i port w TsgcWebSocketClient, a następnie ustawić Active := True, aby nawiązać połączenie.

MQTT v5.0 nie jest wstecznie kompatybilny z wersją v3.1.1. Wprowadzono zbyt wiele nowych elementów, więc istniejące implementacje wymagają przeglądu.

Zgodnie ze specyfikacją, MQTT v5.0 wprowadza znaczną liczbę nowych funkcji do MQTT, zachowując jednocześnie większość rdzenia w niezmienionej formie.

- Funkcjonalność flagi Clean Session jest podzielona na 2 właściwości, aby zapewnić dokładniejszą kontrolę nad danymi stanu sesji: parametr CleanStart oraz nowy SessionExpInterval.
- Rozłączenie przez serwer: umożliwia serwerowi wysłanie DISCONNECT w celu wskazania przyczyny zamknięcia połączenia.
- Wszystkie pakiety odpowiedzi (CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBACK, UNSUBACK, DISCONNECT) zawierają teraz kod przyczyny i ciąg przyczyny opisujące, dlaczego operacje zakończyły się powodzeniem lub niepowodzeniem.
- Ulepszone uwierzytelnianie: Zapewnia mechanizm umożliwiający uwierzytelnianie w stylu challenge/response, w tym uwierzytelnianie wzajemne. Pozwala to na stosowanie uwierzytelniania w stylu SASL, jeśli jest ono obsługiwane zarówno przez klienta, jak i serwer, oraz umożliwia klientowi ponowne uwierzytelnienie w ramach istniejącego połączenia.
- Wzorzec żądanie/odpowiedź jest sformalizowany przez dodanie tematu ResponseTopic.
- Subskrypcje współdzielone: dodano obsługę subskrypcji współdzielonych umożliwiających równoważenie obciążenia wśród konsumentów subskrypcji.
- Aliasy tematów mogą być wysyłane zarówno przez klienta, jak i serwer w celu odwoływania się do filtrów tematów za pomocą krótszych identyfikatorów numerycznych, co pozwala oszczędzać przepustowość.
- Serwery mogą informować o obsługiwanych funkcjach w ConnectionProperties.
- Odwołanie do serwera: umożliwia serwerowi wskazanie alternatywnego serwera do użycia w komunikatach CONNACK lub DISCONNECT. Może służyć jako przekierowanie lub do inicjowania obsługi urządzenia.
- Więcej: obsługiwane są wygasanie wiadomości, Receive Maximums i Maximum Packet Sizes oraz interwał Will Delay.

# Dokumentacja

- Properties — 10 właściwości dostępnych w komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 13 zdarzeń do obsłużenia.

# TsgcWSPClient\_MQTT | Połączenie klienta MQTT

Aby połączyć się z serwerem MQTT, należy najpierw utworzyć komponent [TsgcWebSocketClient](#) oraz [TsgcWSPClient\\_MQTT](#), a następnie podłączyć komponent MQTT do klienta WebSocket.

## Podstawowe użycie

Połącz z serwerem MQTT Mosquitto przy użyciu protokołu WebSocket. Po nawiązaniu połączenia zasubskrybuj temat: „topic1”.

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
    const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
    oMQTT->Subscribe("topic1");
}
```

## Identyfikator klienta

MQTT wymaga **identyfikatora klienta** do identyfikacji połączenia klienta. Składnik automatycznie ustawia **losową wartość**, jednak w razie potrzeby można podać własny identyfikator klienta. W tym celu należy obsłużyć zdarzenie **OnBeforeConnect** i ustawić własną wartość w parametrze aClientIdentifier.

```
void OnMQTTBeforeConnect(TsgcWSConnection *Connection, ref bool aCleanSession,
    ref string aClientIdentifier)
{
    aClientIdentifier = "your client id";
}
```

## Uwierzytelnianie

Niektóre serwery wymagają nazwy użytkownika i hasła do **autoryzacji połączeń MQTT**. Należy użyć właściwości **Authentication**, aby ustawić wartość nazwy użytkownika i hasła przed połączeniem z serwerem.

```
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Authentication->Enabled = true;
```



```
oMQTT->Authentication->UserName = "your user";  
oMQTT->Authentication->Password = "your password";
```

# TsgcWSPClient\_MQTT | Połączenie MQTT Mosquitto

Należy użyć poniższych przykładowych konfiguracji, aby nawiązać połączenie z serwerem Mosquitto MQTT.

## MOSQUITTO MQTT WebSockets

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;
```

## MOSQUITTO MQTT WebSockets TLS

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8081;
oClient->TLS = true;
oClient->TLSOptions->Version = tls1_2;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;
```

## MOSQUITTO MQTT Plain TCP

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 1883;
oClient->Specifications->RFC6455 = false;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;
```

## MOSQUITTO MQTT Plain TCP TLS

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8083;
oClient->Specifications->RFC6455 = false;
oClient->TLS = true;
oClient->TLSOptions->Version = tls1_2;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;
```

# TsgcWSPClient\_MQTT | Sesje klienta MQTT

## Clean Start

W zdarzeniu **OnMQTTBeforeConnect** znajduje się parametr **aCleanSession**. Jeśli wartość tego parametru wynosi **True**, oznacza to, że klient **chce rozpocząć nową sesję**, więc jeśli serwer przechowuje jakąkolwiek sesję, musi ją odrzucić. W takim przypadku parametr **aSession** w zdarzeniu **OnMQTTConnect** będzie miał wartość **false**. Jeśli wartość parametru wynosi **False** i istnieje sesja skojarzona z tym identyfikatorem klienta, serwer musi wznowić komunikację z klientem w stanie odpowiadającym istniejącej sesji.

Jeśli klient doświadczy **nieoczekiwanego rozłączenia** i wymagane jest **wznowienie sesji** w miejscu rozłączenia, należy w zdarzeniu **OnMQTTBeforeConnect** ustawić **aCleanSession = True** oraz **aClientIdentifier = identyfikator klienta sesji**.

## Sesja

Po pomyślnym połączeniu sprawdź zdarzenie **OnMQTTConnect** i wartość parametru **Session**.

**Session = true**, oznacza wznowienie sesji.

**Session = false** oznacza, że jest to nowa sesja.

```
void OnMQTTBeforeConnect(TsgcWSConnection *Connection, ref bool aCleanSession,
    ref string aClientIdentifier)
{
    aCleanSession = false;
    aClientIdentifier = "previous client id";
}

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
    const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
    if (Session == true)
    {
        WriteLn("Session resumed");
    }
    else
    {
        WriteLn("New Session");
    }
}
```

# TsgcWSPClient\_MQTT | Wersja MQTT klienta

Aktualnie klient MQTT obsługuje następujące specyfikacje:

- **MQTT 3.1.1:** <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>
- **MQTT 5.0:** <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>

Można wybrać wersję, której komponent klienta MQTT będzie używał, za pomocą właściwości MQTTVersion.

**MQTT 3.1.1:** TsgcWSPClient\_MQTT.Version = mqtt311

**MQTT 5.0:** sgcWSPClient\_MQTT.Version = mqtt5

# TsgcWSPClient\_MQTT | Publikowanie i subskrypcja MQTT

Wzorzec publikowania/subskrybowania (znany również jako pub/sub) stanowi alternatywę dla tradycyjnej architektury klient-serwer. W modelu klient-serwer klient komunikuje się bezpośrednio z punktem końcowym. Model pub/sub **oddziela klienta wysyłającego wiadomość (wydawcę) od klienta lub klientów odbierających wiadomości (subskrybentów)**. Wydawcy i subskrybenci nigdy nie kontaktują się bezpośrednio. W rzeczywistości nie są nawet świadomi nawzajem swojego istnienia. **Połączenie między nimi jest obsługiwane przez komponent trzeci (broker)**. Zadaniem brokera jest filtrowanie wszystkich przychodzących wiadomości i poprawne ich dystrybuowanie do subskrybentów.

Przy użyciu **TsgcWSPClient\_MQTT** można **publikować wiadomości i subskrybować tematy**.

## Subskrybuj temat

Subskrybuj temat "topic1" po pomyślnym nawiązaniu połączenia.

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
    const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
    oMQTT->Subscribe("topic1");
}
```

## Publikuj wiadomość

Opublikuj wiadomość do wszystkich subskrybentów kanału "topic1"

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
    const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
    oMQTT->Publish("topic1", "Hello Subscribers topic1");
}
```

# TsgcWSPClient\_MQTT | Tematy MQTT

## Tematy

W MQTT słowo „temat” (topic) odnosi się do ciągu UTF-8, którego broker używa do filtrowania wiadomości dla każdego podłączonego klienta. Temat składa się z jednego lub więcej poziomów tematu. Każdy poziom tematu jest oddzielony ukośnikiem (separatorem poziomów tematu).

myHome / groundfloor / livingroom / temperature

W porównaniu z kolejką wiadomości, tematy MQTT są bardzo lekkie. Klient nie musi tworzyć żądanego tematu przed opublikowaniem lub zasubskrybowaniem go. Broker akceptuje każdy prawidłowy temat bez żadnej wcześniejszej inicjalizacji. Należy pamiętać, że każdy temat musi zawierać co najmniej 1 znak i że ciąg tematu dopuszcza spacje. Tematy uwzględniają wielkość liter.

## Symbole wieloznaczne

Gdy klient subskrybuje temat, może subskrybować dokładny temat opublikowanej wiadomości lub użyć symboli wieloznacznych do jednoczesnej subskrypcji wielu tematów. Symbol wieloznaczny może być używany tylko do subskrypcji tematów, a nie do publikowania wiadomości. Istnieją dwa rodzaje symboli wieloznacznych: \_jednopoziomowy i \_wielopoziomowy.

### Jeden poziom: +

Jak sama nazwa wskazuje, symbol wieloznaczny pojedynczego poziomu zastępuje jeden poziom tematu. Symbol plus reprezentuje symbol wieloznaczny pojedynczego poziomu w temacie.

myHome / groundfloor / + / temperature

Każdy temat pasuje do tematu z jednopoziomowym symbolem wieloznacznym, jeśli zawiera dowolny ciąg zamiast symbolu wieloznacznego. Na przykład subskrypcja \_myhome/groundfloor/+/temperature może zwrócić następujące wyniki:

YES => myHome / groundfloor / livingroom / temperature

YES => myHome / groundfloor / kitchen / temperature

NO => myHome / groundfloor / livingroom / brightness

NIE => myHome / firstfloor / livingroom / temperature

NO => myHome / groundfloor / kitchen / fridge / temperature

### **Wielopoziomowy: #**

Wielopoziomowy symbol wieloznaczny obejmuje wiele poziomów tematu. Znak krzyżyka reprezentuje wielopoziomowy symbol wieloznaczny w temacie. Aby broker mógł określić, które tematy pasują, wielopoziomowy symbol wieloznaczny musi być umieszczony jako ostatni znak tematu i poprzedzony ukośnikiem.

myHome / groundfloor / #

YES => myHome / groundfloor / livingroom / temperature

YES => myHome / groundfloor / kitchen / temperature

YES => myHome / groundfloor / kitchen / brightness

NIE => myHome / firstfloor / kitchen / temperature

Gdy klient subskrybuje temat z wielopoziomowym symbolem wieloznacznym, otrzymuje wszystkie wiadomości dotyczące tematu rozpoczynającego się od wzorca przed znakiem wieloznacznym, niezależnie od długości lub głębokości tematu. Jeśli jako temat określony jest jedynie wielopoziomowy symbol wieloznaczny (\_#), otrzymywane są wszystkie wiadomości wysyłane do brokera MQTT.

# TsgcWSPClient\_MQTT | Subskrypcja MQTT

Subskrypcji tematu można dokonać za pomocą metody Subscribe komponentu TsgcWSPClient\_MQTT. Metoda ta przyjmuje następujące parametry:

**Topic:** nazwa tematu, do którego ma zostać dokonana subskrypcja.

**QoS:** jeden z 3 poziomów QoS (nie wszystkie brokery obsługują wszystkie 3 poziomy). Jeśli nie zostanie określony, używany jest poziom `mtqsAtMostOnce`. Więcej informacji na temat [poziomów QoS](#).

**SubscribeProperties:** w przypadku MQTT 5.0 są to dodatkowe właściwości subskrypcji.

## Subskrybuj QoS = Co najmniej raz

```
MQTT->Subscribe("topic1", mtqsAtLeastOnce);
```

## Subskrypcja MQTT 5.0

```
oProperties = new TsgcWSMQTTSubscribe_Properties();  
oProperties->SubscriptionIdentifier = 1234;  
oProperties->UserProperties->Add("name=value");  
  
MQTT->Subscribe("topic1", mtqsAtMostOnce, oProperties);
```



# TsgcWSPClient\_MQTT | Publikowanie wiadomości MQTT

Wiadomości można publikować do wszystkich subskrybentów tematu przy użyciu metody **Publish**, która przyjmuje następujące parametry:

**Topic:** nazwa tematu, w którym zostanie opublikowana wiadomość.

**Text:** to tekst wiadomości.

**QoS:** jeden z 3 poziomów QoS (nie wszystkie brokery obsługują wszystkie 3 poziomy). Jeśli nie zostanie określony, używany jest poziom `mtqsAtMostOnce`. Więcej informacji na temat [poziomów QoS](#).

**Retain:** jeśli wartość wynosi `true`, wiadomość zostanie zatrzymana. Za każdym razem, gdy nowy klient zasubskrybuje ten temat, ta wiadomość zostanie do niego wysłana.

**PublishProperties:** w przypadku MQTT 5.0 są to właściwości wiadomości.

## Opublikuj prostą wiadomość

```
MQTT->Publish("topic1", "Hello Subscribers topic1");
```

## Publish QoS = At Least Once

```
MQTT->Publish("topic1", "Hello Subscribers topic1", mtqsAtLeastOnce);
```

## Publikuj wiadomość Retained

```
MQTT->Publish("topic1", "Hello Subscribers topic1", mtqsAtMostOnce, true);
```

## TsgcWSPClient\_MQTT | Odbieranie wiadomości MQTT

Komunikaty wysyłane przez serwer są odbierane w zdarzeniu **OnMQTTPublish**. Zdarzenie to ma następujące parametry:

**Topic:** jest nazwą tematu powiązanego z tą wiadomością.

**Text:** to tekst wiadomości.

**PublishProperties:** w przypadku MQTT 5.0 są to właściwości opublikowanej wiadomości.

### Odczyt opublikowanych wiadomości

```
void OnMQTTPublish(TsgcWSConnection *Connection, string aTopic, string aText,
    TsgcWSMQTTPublishProperties *PublishProperties)
{
    WriteLine("Topic: " + aTopic + ". Message: " + aText);
}
```

## TsgcWSPClient\_MQTT | Odbiór wiadomości MQTT (rozszerzony)

Zdarzenie **OnMQTTPublishEx** udostępnia ładunek opublikowanej wiadomości w wielu formatach za pośrednictwem obiektu **TsgcWSMQTTPublishData**. Zdarzenie posiada następujące parametry:

**Topic:** jest nazwą tematu powiązanego z tą wiadomością.

**Data:** zawiera ładunek opublikowanej wiadomości. Posiada następujące właściwości:

- **Value:** ładunek jako ciąg znaków.
- **Bytes:** surowy ładunek jako TBytes.
- **Stream:** surowy ładunek jako TMemoryStream.

**PublishProperties:** w przypadku MQTT 5.0 są to właściwości opublikowanej wiadomości.

### Odczyt opublikowanych wiadomości (rozszerzony)

```
void OnMQTTPublishEx(TsgcWSConnection *Connection, string aTopic,
    TsgcWSMQTTPublishData *aData, TsgcWSMQTTPublishProperties *PublishProperties)
{
    // read as string
    WriteLine("Topic: " + aTopic + ". Message: " + aData->Value);
    // read as bytes
    WriteLine("Bytes Length: " + IntToStr(aData->Bytes.Length));
}
```

```
// read as stream
WriteLn("Stream Size: " + IntToStr(aData->Stream->Size));
}
```

# TsgcWSPClient\_MQTT | Publikowanie i oczekiwanie na odpowiedź

Klient MQTT obsługuje kilka poziomów QoS, z których każdy działa w inny sposób, aby zapewnić oczekiwane przetwarzanie wiadomości.

Dostępne są następujące poziomy QoS:

- **mtqsAtMostOnce:** (domyślnie) wiadomość jest dostarczana zgodnie z najlepszymi wysiłkami bazowej sieci TCP/IP. Odpowiedź nie jest oczekiwana i w protokole nie są zdefiniowane żadne semantyki ponownych prób. Wiadomość dociera do serwera raz lub wcale.
- **mtqsAtLeastOnce:** odbiór wiadomości przez serwer jest potwierdzany komunikatem ACKNOWLEDGMENT. Jeśli wystąpi zidentyfikowana awaria łącza komunikacyjnego lub urządzenia wysyłającego, albo komunikat potwierdzenia nie zostanie odebrany po określonym czasie, nadawca ponownie wysyła wiadomość. Wiadomość dociera do serwera co najmniej raz. Wiadomość z poziomem QoS 1 ma w wiadomości parametr ID.
- **mtqsExactlyOnce:** gdzie wiadomości mają gwarancję dostarczenia dokładnie raz. Ten poziom może być używany na przykład w systemach rozliczeniowych, gdzie duplikaty lub utracone wiadomości mogą prowadzić do naliczenia nieprawidłowych opłat. W przypadku wykrycia awarii łącza komunikacyjnego lub urządzenia wysyłającego lub gdy komunikat potwierdzenia nie zostanie odebrany po określonym czasie, nadawca ponownie wysyła wiadomość.

Możesz obsługiwać zdarzenia OnPubAck lub OnPubComp, aby dowiedzieć się, czy wiadomość została przetworzona przez serwer, albo użyć metody **PublishAndWait**, aby sprawdzić, czy wiadomość została przetworzona przez serwer.

Użycie metody **PublishAndWait** jest takie samo jak zwykłej metody Publish, ale zawiera dodatkowy parametr Timeout. Metoda zwraca False, jeśli po upływie określonego czasu nie otrzyma odpowiedzi z serwera. Domyślna wartość wynosi 10 sekund.

```
if mqtt->PublishAndWait("topic", "text")
{
    ShowMessage("Message processed")
}
else
{
    ShowMessage("Message error");
}
```

## TsgcWSPClient\_MQTT | MQTT Czyszczenie zachowanych wiadomości

Domyślnie każdy temat MQTT może mieć zachowaną wiadomość. Standardowy mechanizm MQTT do czyszczenia zachowanych wiadomości polega na wysłaniu zachowanej wiadomości z pustym ładunkiem do tematu. Spowoduje to usunięcie zachowanej wiadomości.

```
MQTT->Publish("topic1", "", mqttAtMostOnce, true);
```

# Protokół AMQP 0.9.1

**Advanced Message Queuing Protocol** (AMQP) to otwarty standardowy protokół warstwy aplikacji dla oprogramowania pośredniczącego zorientowanego na wiadomości. Kluczowymi cechami AMQP są orientacja na wiadomości, kolejkowanie, routing (w tym punkt-do-punktu i publikuj-subskrybuj), niezawodność i bezpieczeństwo.

AMQP jest binarnym protokołem warstwy aplikacji zaprojektowanym w celu efektywnej obsługi szerokiej gamy aplikacji przesyłania wiadomości i wzorców komunikacji. Zapewnia kontrolowany przepływ, zorientowany na wiadomości sposób komunikacji z gwarancjami dostarczania wiadomości, takimi jak co najwyżej raz (każda wiadomość jest dostarczana raz lub wcale), co najmniej raz (każda wiadomość jest na pewno dostarczona, ale może się to zdarzyć wielokrotnie) i dokładnie raz (wiadomość zawsze dotrze i tylko raz), a także uwierzytelnianie i/lub szyfrowanie oparte na SASL i/lub TLS. Zakłada obecność niezawodnej warstwy transportowej, takiej jak Transmission Control Protocol (TCP).

## Funkcje

AMQP może być używany w każdej sytuacji, gdy wymagana jest wysokiej jakości i bezpieczna dostawa wiadomości między klientem a brokerem.

AMQP zapewnia następujące funkcje:

- Monitorowanie i udostępnianie aktualizacji.
- Zapewnianie szybkiej odpowiedzi serwera na żądania i przekazywanie czasochłonnych zadań do dalszego przetwarzania.
- Dystrybuuj wiadomości do wielu odbiorców.
- Połączenie klientów offline w celu dalszego pobierania danych.
- Zwiększ niezawodność i sprawność działania aplikacji.
- Niezawodność dostarczania wiadomości.
- Wysoka szybkość dostarczania wiadomości.
- Akceptacja wiadomości.

## Komponenty

**TsgcWSPClient\_AMQP**: to komponent klienta implementujący protokół **AMQP 0.9.1**.

## Najczęstsze zastosowania

- **Połączenie**
  - [Client AMQP Connect](#)
  - [Client AMQP Disconnect](#)
- **Polecenia**
  - [Kanały AMQP](#)

- [Giełdy AMQP](#)
- [Kolejki AMQP](#)
- [Publikowanie wiadomości AMQP](#)
- [Konsumowanie wiadomości AMQP](#) (asynchroniczne)
- [Pobieranie wiadomości AMQP](#) (synchroniczne)
- [AMQP QoS](#)
- [Transakcje AMQP](#)

# TsgcWSPClient\_AMQP

AMQP 0.

## Wprowadzenie

Klient **TsgcWSPClient\_AMQP** implementuje pełny protokół **AMQP 0.9.1** zgodnie ze specyfikacją OASIS. Klient obsługuje połączenia poprzez zwykły TCP oraz WebSocket; obsługiwane są również połączenia TLS (szyfrowane).

## Połączenie

Protokół AMQP 0.9.1 definiuje koncepcję kanałów, która umożliwia współdzielenie jednego połączenia gniazda przez kilka wirtualnych kanałów. Klient implementuje wewnętrzny wątek odczytujący odebrane bajty i przekazujący każdą wiadomość do właściwego kanału (który działa we własnym wątku). Przy uruchomieniu połączenia AMQP z 5 kanałami klient uruchomi 6 wątków (5 wątków obsługujących dane każdego kanału i 1 wątek obsługujący dane połączenia).

Przed nawiązaniem połączenia z serwerem AMQP należy skonfigurować następujące właściwości protokołu AMQP

- **AMQPOptions.Locale:** jest to lokalizacja wiadomości do użycia; wartość jest negocjowana, więc może się różnić od lokalizacji obsługiwanych przez serwer. Wartość domyślna to "en\_US".
- **AMQPOptions.MaxChannels:** maksymalna liczba kanałów, które można otworzyć; jest to wartość negocjowana, więc może różnić się od konfiguracji serwera. Wartość domyślna to 65535.
- **AMQPOptions.MaxFrameSize:** maksymalny rozmiar ramki AMQP w bajtach; jest to wartość negocjowana, więc może się różnić od konfiguracji serwera. Wartość domyślna wynosi 2147483647.
- **AMQPOptions.VirtualHost:** nazwa wirtualnego hosta. Wartość domyślna to "/"

Funkcja HeartBeat AMQP może być również skonfigurowana przed połączeniem z serwerem. Można włączyć lub wyłączyć użycie heartbeatów.

- **HeartBeat.Enabled:** należy ustawić na true, jeśli klient obsługuje HeartBeats.
- **HeartBeat.Interval:** żądany interwał w sekundach.

Po skonfigurowaniu klienta AMQP należy dołączyć go do [TsgcWebSocketClient](#), a następnie skonfigurować właściwości połączenia serwera, aby połączyć się z serwerem AMQP.



Należy ustawić wartość właściwości **Specifications.RFC6455** na false w przypadku korzystania z połączenia zwykłego TCP zamiast połączenia WebSocket.

```
TsgcWSPClient_AMQP *oAMQP = new TsgcWSPClient_AMQP();
oAMQP->AMQPOptions->Locale = "en_US";
oAMQP->AMQPOptions->MaxChannels = 100;
oAMQP->AMQPOptions->MaxFrameSize = 16384;
oAMQP->AMQPOptions->VirtualHost = "/";
oAMQP->HeartBeat->Enabled = true;
oAMQP->HeartBeat->Interval = 60;

TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
oAMQP->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 5672;
oClient->Active = true;
```

## Kanały

Po nawiązaniu połączenia klienta AMQP możliwe jest otwarcie pierwszego kanału.

```
oAMQP->OpenChannel("channel_name");
```

## Giełdy

Po otwarciu kanału klient może deklarować nowe wymiany, sprawdzać ich istnienie itd. Należy użyć metody **DeclareExchange**, aby zadeklarować nową wymianę.

```
oAMQP->DeclareExchange("channel_name", "exchange_name");
```

## Kolejki

Po otwarciu kanału klient może deklarować nowe kolejki, weryfikować ich istnienie... metodą **DeclareQueue** należy zadeklarować nową kolejkę. Kolejki nie są domyślnie dostarczane przez serwer (w przeciwieństwie do giełd), dlatego zawsze wymagane jest zadeklarowanie nowej kolejki (chyba że kolejka została już utworzona przez innego klienta).

```
oAMQP->DeclareQueue("channel_name", "queue_name");
```

## Kolejki powiązań

Po skonfigurowaniu Exchange'ów i kolejek może być konieczne powiązanie kolejek z Exchange'ami, dzięki czemu Exchange'y będą wiedziały, do których kolejek kierować wiadomości.

Serwery AMQP automatycznie wiążą kolejki z wymianą „direct”, używając nazwy kolejki jako klucza routingu. Umożliwia to wysyłanie wiadomości do konkretnej kolejki bez konieczności deklarowania powiązania (wystarczy wywołać metodę `PublishMessage`, przekazując argument `Exchange` jako pustą wartość oraz nazwę kolejki w argumencie `RoutingKey`).

```
oAMQP->BindQueue("channel_name", "queue_name", "exchange_name", "routing_key");
```

## Wyślij wiadomości

Wywołaj metodę **`PublishMessage`**, aby opublikować nową wiadomość AMQP. Metoda umożliwia publikowanie wiadomości jako **`String`** lub **`TStream`**.

```
oAMQP->PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!!");
```

## Odbieranie wiadomości

AMQP umożliwia odbieranie wiadomości w 2 trybach:

- **Żądanie od klienta:** za pomocą metody **`GetMessage`**. Jeśli w kolejce nie ma wiadomości, zostanie wywołane zdarzenie **`OnAMQPBasicGetEmpty`**.
- **Wypychane przez serwer:** przy użyciu metody `Consume`.

## Żądanie klienta

```
oAMQP->GetMessage("channel_name", "queue_name");

void OnAMQPGetOk(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicGetOk *aGetOk, const TsgcAMQPMessageContent *aContent)
{
    DoLog("#AMQP_basic_GetOk: " + aChannel + " " + IntToStr(aGetOk->MessageCount) + " " + aContent->Body->AsString);
}
```

## Wysłane przez serwer

```
oAMQP->Consume("channel_name", "queue_name");

void OnAMQPGetOk(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicGetOk *aGetOk, const TsgcAMQPMessageContent *aContent)
{
    DoLog("#AMQP_basic_GetOk: " + aChannel + " " + IntToStr(aGetOk->MessageCount) + " " + aContent->Body->AsString);
}
```

## Dokumentacja

- Właściwości — 6 właściwości dostępnych w komponencie.

- Metody — 41 metod publicznych.
- Zdarzenia — 25 zdarzeń do obsługi.

# Połączenie | Połączenie klienta AMQP

Aby połączyć się z serwerem AMQP, należy najpierw utworzyć [TsgcWebSocketClient](#) oraz [TsgcWSPClient\\_AMQP](#), a następnie dołączyć komponent AMQP do klienta WebSocket.

## Podstawowe użycie

Połączenie z serwerem AMQP bez uwierzytelniania. Należy zdefiniować wartości właściwości AMQPOptions, wirtualny host, a następnie ustawić w TsgcWebSocketClient adres Host i Port serwera.

Jeśli używane jest połączenie TCP Plain, należy ustawić właściwość Specifications.RFC6455 klienta TsgcWebSocketClient na false.

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP->AMQPOptions->Locale = "en_US";
oAMQP->AMQPOptions->MaxChannels = 100;
oAMQP->AMQPOptions->MaxFrameSize = 16384;
oAMQP->AMQPOptions->VirtualHost = "/";
oAMQP->HeartBeat->Enabled = true;
oAMQP->HeartBeat->Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 5672;
oClient->Active = true;
```

## Uwierzytelnianie

Jeśli serwer wymaga uwierzytelnienia, należy użyć zdarzenia **OnAMQPAuthentication**, aby wybrać mechanizm uwierzytelniania (jeśli jest wymagany) i ustawić nazwę użytkownika oraz hasło.

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP->AMQPOptions->Locale = "en_US";
oAMQP->AMQPOptions->MaxChannels = 100;
oAMQP->AMQPOptions->MaxFrameSize = 16384;
oAMQP->AMQPOptions->VirtualHost = "/";
oAMQP->HeartBeat->Enabled = true;
oAMQP->HeartBeat->Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 5672;
oClient->Active = true;

void OnAMQPAuthentication(TObject *Sender, TsgcAMQPAuthentications *aMechanisms, TsgcAMQPAuthentication &Mechanism,
    String &User, String &Password)
{
    User = "user_value";
    Password = "password_value";
}
```



## Połączenie | Rozłączenie klienta AMQP

Klient może rozłączyć aktywne połączenie przy użyciu następujących metod:

### Wysyłanie powodu zamknięcia połączenia

Klient AMQP może poinformować serwer o zamiarze zamknięcia połączenia i podać powód tego zamknięcia. Aby zażądać zamknięcia połączenia od serwera, należy użyć metody Close.

```
oAMQP.Close(541, "Internal Error");
```

### Zamykanie połączenia gniazda

Wystarczy ustawić właściwość Active [TsgcWebSocketClient](#) na False. Więcej informacji o [zamykaniu połączeń](#).

## Commands | AMQP Channels

AMQP jest protokołem wielokanałowym. Kanały umożliwiają multipleksowanie ciężkiego połączenia TCP/IP na kilka lekkich połączeń. Dzięki temu protokół jest bardziej „przyjazny dla zapory sieciowej”, ponieważ użycie portów jest przewidywalne. Oznacza to również, że kształtowanie ruchu i inne funkcje QoS sieci mogą być łatwo stosowane.

Każdy kanał działa we własnym wątku — za każdym razem, gdy nadejdzie nowa wiadomość, klient najpierw identyfikuje kanał i umieszcza wiadomość w kolejce przetwarzanej przez wątek kanału.

Cykl życia kanału jest następujący:

1. Klient otwiera nowy kanał (Open).
2. Serwer potwierdza gotowość nowego kanału (Open-Ok).
3. Klient i serwer używają kanału w żądany sposób.
4. Jeden z uczestników (klient lub serwer) zamyka kanał (Close).
5. Drugi węzeł potwierdza zamknięcie kanału (Close-Ok).

### Otwórz kanał

Aby utworzyć nowy kanał, wystarczy wywołać metodę **OpenChannel** i przekazać nazwę kanału jako argument. Zdarzenie **OnAMQPChannelOpen** jest wywoływane jako potwierdzenie wysłane przez serwer, że kanał został otwarty.

```
AMQP->OpenChannel("channel_name");
```

```
private void OnAMQPChannelOpen(TObject *Sender, const string aChannel)
{
    DoLog("#AMQP_channel_open: " + aChannel);
}
```

Wywołanie synchroniczne można również wykonać, wywołując metodę **OpenChannelEx**; metoda ta zwraca wartość true, jeśli kanał został otwarty, lub false, jeśli potwierdzenie od serwera nie nadeszło.

```
if AMQP->OpenChannelEx("channel_name")
{
    DoLog("#AMQP_channel_open channel_name");
}
else
{
}
```

```
DoLog("#AMQP_channel_open_error");  
}
```

## Zamknij kanał

Aby zamknąć istniejący kanał, należy wywołać metodę **CloseChannel** i przekazać nazwę kanału jako argument. Zdarzenie **OnAMQPChannelClose** zostanie wywołane, gdy klient otrzyma potwierdzenie zamknięcia kanału.

Wywołanie synchroniczne można wykonać, wywołując metodę **CloseChannelEx**; metoda ta zwraca true, jeśli kanał został zamknięty, lub false, jeśli nie nadeszło potwierdzenie z serwera.

## Przepływ kanału

Kontrola przepływu to procedura awaryjna służąca do zatrzymania przepływu wiadomości od równorzędnego punktu. Działa w ten sam sposób między klientem a serwerem i jest implementowana przez polecenia **EnableChannel** / **DisableChannel**. Kontrola przepływu jest jedynym mechanizmem mogącym zatrzymać nadmiernie produkującego wydawcę.

Aby wyłączyć przepływ kanału, należy wywołać metodę **DisableChannel**. Zdarzenie **OnAMQPChannelFlow** zostanie wywołane, gdy klient otrzyma potwierdzenie, że przepływ kanału został wyłączony.

To samo dotyczy włączania przepływu kanału — należy wywołać metodę **EnableChannel**, a zdarzenie **OnAMQPChannelFlow** zostanie wywołane, gdy klient otrzyma potwierdzenie, że przepływ kanału został włączony.

Żądania synchroniczne są dostępne poprzez funkcje **EnableChannelEx** i **DisableChannelEx**.



## Polecenia | Giełdy AMQP

Klasa giełdy umożliwia aplikacji zarządzanie giełdami na serwerze. Pozwala ona aplikacji na skryptowe okablowanie własnych połączeń (zamiast polegania na interfejsie konfiguracyjnym). Uwaga: większość aplikacji nie wymaga tak wysokiego poziomu zaawansowania, a starsze oprogramowanie pośredniczące prawdopodobnie nie będzie obsługiwać tej semantyki.

Cykl życia giełdy jest następujący:

1. Klient prosi serwer o upewnienie się, że wymiana istnieje (Declare). Klient może doprecyzować żądanie: "utwórz wymianę, jeśli nie istnieje" lub "ostrzeż mnie, ale nie twórz jej, jeśli nie istnieje".
2. Klient publikuje wiadomości do wymiany.
3. Klient może zdecydować się na usunięcie wymiany (Delete).

### Deklaruj giełdę

Ta metoda tworzy nową giełdę lub weryfikuje, czy dana giełda już istnieje. Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **ExchangeName:** nazwa wymiany, nie może być dłuższa niż 255 znaków i nie może zaczynać się od „amq.” (chyba że parametr `passive` wynosi `True`).
- **ExchangeType:** to typ giełdy; wszystkie serwery AMQP obsługują giełdy "direct" i "fanout". Należy zapoznać się z dokumentacją serwera, aby dowiedzieć się, które typy giełd są obsługiwane.
- **Passive:** jeśli `passive` ma wartość `true`, serwer weryfikuje wyłącznie, czy wymiana jest już zadeklarowana. Jeśli `passive` ma wartość `false`, a wymiana nie istnieje, serwer utworzy nową.
- **Durable:** jeśli `true`, wymiana zostanie odtworzona po uruchomieniu serwera. Jeśli `false`, wymiana zostanie usunięta po zatrzymaniu serwera.
- **AutoDelete:** jeśli wartość wynosi `true`, wymiana zostanie usunięta, gdy wszystkie kolejki zostaną odwiązane.
- **Internal:** zawsze `false`.
- **NoWait:** jeśli ma wartość `true`, serwer nie wysyła potwierdzenia do klienta.
- **Arguments:** ciąg znaków zawierający niestandardowe argumenty; wartości muszą być przekazane jako ciąg JSON, przykład: `{"x-dead-letter-exchange":"my-dlx"}`.

Aby zadeklarować nową giełdę, należy wywołać metodę **DeclareExchange** i przekazać nazwę kanału, nazwę giełdy i typ giełdy jako argumenty. Zdarzenie **OnAMQPExchangeDeclare** jest wywoływane jako potwierdzenie wysłane przez serwer, że giełda została zadeklarowana.

```
AMQP->DeclareExchange("channel_name", "exchange_name", "direct");
```

```
private void OnAMQPExchangeDeclare(TObject *Sender, const string aChannel, const string aExchange)
{
    DoLog("#AMQP_exchange_declare: [" + aChannel + "] " + aExchange);
}
```

Synchroniczne wywołanie można wykonać również za pomocą metody **DeclareExchangeEx**, która zwraca True, jeśli wymiana została zadeklarowana, lub False, jeśli nie dotarło potwierdzenie z serwera.

```
if AMQP->DeclareExchangeEx("channel_name", "exchange_name", "direct")
{
    DoLog("#AMQP_exchange_declare: [" + aChannel + "] " + aExchange);
}
else
{
    DoLog("#AMQP_exchange_declare_error");
}
```

## Usuwanie giełdy

Ta metoda służy do usunięcia istniejącej giełdy. Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **ExchangeName:** nazwa wymiany, nie może być dłuższa niż 255 znaków i nie może zaczynać się od „amq.” (chyba że parametr passive wynosi True).
- **IfUnused:** serwer usuwa giełdę tylko wtedy, gdy nie ma żadnych kolejek z nią powiązanych.
- **NoWait:** jeśli ma wartość true, serwer nie wysyła potwierdzenia do klienta.

Aby usunąć istniejącą giełdę, należy wywołać metodę **DeleteExchange** i przekazać nazwę kanału oraz nazwę giełdy jako argumenty. Zdarzenie **OnAMQPExchangeDelete** jest wywoływane jako potwierdzenie wysłane przez serwer, że giełda została usunięta.

Wywołanie synchroniczne można również wykonać metodą **DeleteExchangeEx**; metoda ta zwraca true, jeśli giełda została usunięta, i false, jeśli nie otrzymano potwierdzenia od serwera.

## Polecenia | AMQP Queues

Klasa kolejki umożliwia aplikacji zarządzanie kolejkami wiadomości na serwerze. Jest to podstawowy krok w niemal wszystkich aplikacjach odbierających wiadomości, co najmniej w celu weryfikacji, czy oczekiwana kolejka wiadomości rzeczywiście istnieje.

Cykl życia trwałej kolejki wiadomości jest stosunkowo prosty:

1. Klient potwierdza istnienie kolejki komunikatów (Declare z argumentem „passive”).
2. Serwer potwierdza istnienie kolejki wiadomości (Declare-Ok).
3. Klient odczytuje wiadomości z kolejki wiadomości.

Cykl życia tymczasowej kolejki wiadomości jest bardziej interesujący:

1. Klient tworzy kolejkę wiadomości (Declare, często bez podawania nazwy kolejki, aby serwer przypisał ją automatycznie). Serwer potwierdza operację (Declare-Ok).
2. Klient uruchamia konsumenta w kolejce wiadomości. Dokładna funkcjonalność konsumenta jest definiowana przez klasę Basic.
3. Klient anuluje konsumenta, jawnie lub przez zamknięcie kanału i/lub połączenia.
4. Gdy ostatni konsument zniknie z kolejki wiadomości i po uprzejmym czasie oczekiwania, serwer usuwa kolejkę wiadomości.

AMQP implementuje mechanizm dostarczania dla subskrypcji tematycznych jako kolejki wiadomości. Umożliwia to tworzenie interesujących struktur, w których subskrypcja może być równoważona obciążeniem wśród puli współpracujących subskrybentów aplikacji.

Cykl życia subskrypcji obejmuje dodatkowy etap powiązania:

1. Klient tworzy kolejkę wiadomości (Declare), a serwer potwierdza (Declare-Ok).
2. Klient powiązuje kolejkę wiadomości z wymianą tematyczną (Bind), a serwer potwierdza (Bind-Ok).
3. Klient używa kolejki wiadomości tak jak w poprzednich przykładach.

## Zadeklaruj kolejkę

Ta metoda tworzy nową kolejkę lub weryfikuje, czy kolejka już istnieje. Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od "amq." (z wyjątkiem sytuacji, gdy parametr `passive` jest ustawiony na `true`).
- **Passive:** jeśli `passive` ma wartość `true`, serwer jedynie sprawdza, czy kolejka została już zadeklarowana. Jeśli `passive` ma wartość `false` i kolejka nie istnieje, serwer utworzy nową.
- **Durable:** jeśli `true`, kolejka zostanie odtworzona po uruchomieniu serwera. Jeśli `false`, kolejka zostanie usunięta po zatrzymaniu serwera.
- **Exclusive:** jeśli ustawiono na `true`, kolejka jest dostępna wyłącznie dla bieżącego połączenia.
- **AutoDelete:** jeśli `true`, kolejka zostanie usunięta, gdy wszyscy konsumenci przestaną z niej korzystać.
- **NoWait:** jeśli ma wartość `true`, serwer nie wysyła potwierdzenia do klienta.
- **Arguments:** ciąg znaków zawierający niestandardowe argumenty; wartości muszą być przekazane jako ciąg JSON, przykład: `{"x-dead-letter-exchange":"my-dlx"}`.

Aby zadeklarować nową kolejkę, należy wywołać metodę **DeclareQueue** i przekazać nazwę kanału oraz nazwę kolejki jako argumenty. Zdarzenie **OnAMQPQueueDeclare** jest wywoływane jako potwierdzenie od serwera, że wymiana została zadeklarowana.

```
AMQP->DeclareQueue("channel_name", "queue_name");
```

```
private void OnAMQPExchangeDeclare(TObject *Sender, const string aChannel, const string aQueue,
    int aMessageCount, int aConsumerCount)
{
    DoLog("#AMQP_queue_declare: [" + aChannel + "] " + aQueue));
}
```

Można również wykonać wywołanie synchroniczne, wywołując metodę **DeclareQueueEx**; metoda ta zwraca `true`, jeśli kolejka została zadeklarowana, i `false`, jeśli nie nadeszło potwierdzenie z serwera.

```
if AMQP->DeclareQueueEx("channel_name", "queue_name")
{
    DoLog("#AMQP_queue_declare: [" + aChannel + "] " + aQueue);
}
else
{
    DoLog("#AMQP_queue_declare_error");
}
```

## Usuń kolejkę

Ta metoda służy do usuwania istniejącej kolejki. Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od "amq." (z wyjątkiem sytuacji, gdy parametr `passive` jest ustawiony na `true`).
- **IfUnused:** serwer usuwa kolejkę tylko wtedy, gdy nie ma do niej podłączonych żadnych konsumentów.
- **IfEmpty:** serwer usuwa kolejkę tylko wtedy, gdy nie zawiera żadnych wiadomości.
- **NoWait:** jeśli ma wartość `true`, serwer nie wysyła potwierdzenia do klienta.

Aby usunąć istniejącą kolejkę, należy wywołać metodę **DeleteQueue**, podając jako argumenty nazwę kanału i nazwę kolejki. Zdarzenie **OnAMQPQueueDelete** jest wywoływane jako potwierdzenie wysłane przez serwer, że kolejka została usunięta.

Możliwe jest również wywołanie synchroniczne poprzez metodę **DeleteQueueEx**; metoda ta zwraca true, jeśli kolejka została usunięta, i false, jeśli nie otrzymano potwierdzenia od serwera.

## Powiązanie kolejki

Metoda ta służy do powiązania kolejki z wymianą. Wymiany używają powiązań, aby określić, do których kolejek będą kierowane wiadomości.

Wszystkie serwery AMQP automatycznie wiążą wszystkie kolejki z domyślną wymianą (jest to wymiana „direct” bez nazwy), używając nazwy kolejki jako klucza routingu powiązania. Umożliwia to wysyłanie wiadomości do konkretnej kolejki bez deklarowania powiązania. Wystarczy wywołać metodę **PublishMessage**, przekazać pustą wartość jako nazwę wymiany i ustawić właściwość **RoutingKey** na wartość odpowiadającą **nazwie kolejki**.

Metoda przyjmuje następujące argumenty:

- **ChannelName**: nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName**: nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od „amq.” (z wyjątkiem sytuacji, gdy parametr *passive* jest ustawiony na true).
- **ExchangeName**: nazwa wymiany, nie może być dłuższa niż 255 znaków i nie może zaczynać się od „amq.” (chyba że parametr *passive* wynosi True).
- **RoutingKey**: klucz routingu powiązania.
- **NoWait**: jeśli ma wartość true, serwer nie wysyła potwierdzenia do klienta.
- **Arguments**: ciąg znaków zawierający niestandardowe argumenty; wartości muszą być przekazane jako ciąg JSON, przykład: {"x-dead-letter-exchange": "my-dlx"}.

Aby powiązać kolejkę z wymianą, należy wywołać metodę **BindQueue**, przekazując nazwę kanału, nazwę kolejki, wymianę i klucz routingu jako argumenty. Zdarzenie **OnAMQPQueueBind** jest wywoływane jako potwierdzenie wysłane przez serwer, że kolejka została powiązana.

```
AMQP->BindQueueEx("channel_name", "queue_name", "exchange_name", "routing_key");

private void OnAMQPQueueBind(TObject *Sender, const string aChannel, const string aQueue,
    const string aExchange)
{
    DoLog("#AMQP_queue_bind: [" + aChannel + "] " + aQueue + " -->-- " + aExchange);
}
```

Synchroniczne wywołanie można wykonać też wywołując metodę **BindQueueEx**, która zwraca true, jeśli kolejka została powiązana, i false, jeśli nie dotarło potwierdzenie z serwera.

```
if AMQP->BindQueueEx("channel_name", "queue_name", "exchange_name", "routing_key")
{
    DoLog("#AMQP_queue_bind: [" + aChannel + "] " + aQueue + " --><-- " + aExchange);
}
else
{
    DoLog("#AMQP_queue_bind_error");
}
```

## Odwiązanie kolejki

Metoda ta usuwa istniejące powiązanie kolejki.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od "amq." (z wyjątkiem sytuacji, gdy parametr passive jest ustawiony na true).
- **ExchangeName:** nazwa wymiany, nie może być dłuższa niż 255 znaków i nie może zaczynać się od „amq.” (chyba że parametr passive wynosi True).
- **RoutingKey:** klucz routingu powiązania.

Aby anulować powiązanie kolejki, należy wywołać metodę **UnBindQueue** i przekazać jako argumenty nazwę kanału, nazwę kolejki, wymianę oraz klucz routingu. Zdarzenie **OnAMQPQueueUnBind** jest wywoływane jako potwierdzenie wysłane przez serwer, że powiązanie kolejki zostało usunięte.

Można też wykonać wywołanie synchroniczne, używając metody **UnBindQueueEx**; metoda ta zwraca true, jeśli kolejka została odpięta, lub false, jeśli nie otrzymano potwierdzenia z serwera.

## Wyczyść kolejkę

Metoda ta usuwa wszystkie wiadomości z kolejki. Wiadomości wysłane, lecz oczekujące na potwierdzenie odbioru, nie są objęte tym działaniem.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od "amq." (z wyjątkiem sytuacji, gdy parametr passive jest ustawiony na true).

- **NoWait:** jeśli ma wartość true, serwer nie wysyła potwierdzenia do klienta.

Aby wyczyścić kolejkę, należy wywołać metodę **PurgeQueue** i podać nazwę kanału oraz nazwę kolejki jako argumenty. Zdarzenie **OnAMQPQueuePurge** jest wywoływane jako potwierdzenie wysłane przez serwer o wyczyszczeniu kolejki.

Możliwe jest również wywołanie synchronicznej metody **PurgeQueueEx**; metoda ta zwraca true, jeśli kolejka została wyczyszczona, oraz false, jeśli nie nadeszło potwierdzenie z serwera.

# Polecenia | Publikowanie wiadomości AMQP

## Publikowanie wiadomości

Metoda PublishMessages służy do wysyłania wiadomości do serwera AMQP.

Serwery AMQP automatycznie wiążą kolejki z wymianą „direct”, używając nazwy kolejki jako klucza routingu. Umożliwia to wysyłanie wiadomości do konkretnej kolejki bez konieczności deklarowania powiązania (wystarczy wywołać metodę PublishMessage, przekazując argument Exchange jako pustą wartość oraz nazwę kolejki w argumencie RoutingKey).

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **ExchangeName:** nazwa giełdy; nie może być dłuższa niż 255 znaków i nie może zaczynać się od "amq." (chyba że parametr passive ma wartość true).
- **RoutingKey:** to nazwa klucza routingu dla powiązania.
- **Mandatory:** jeśli true, a wiadomość nie może być przekierowana do żadnej kolejki, serwer zwraca wiadomość i wywołane jest zdarzenie OnAMQPBasicReturn.
- **Immediate:** jeśli true i wiadomość nie może zostać przekierowana do żadnej kolejki, jest zwracana przez serwer i wywołane jest zdarzenie OnAMQPBasicReturn.

```
AMQP->PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!!");
```

```
private void OnAMQPBasicReturn(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicReturn *aReturn,
    const TsgcAMQPMessageContent *aContent)
{
    DoLog("#AMQP_basic_return: " + aChannel + " " + IntToStr(aReturn->ReplyCode) + " " + aReturn->ReplyText + " " + aContent->Text);
}
```

## Potwierdzenia publikacji

W przypadku awarii sieci podczas publikowania wiadomości jedynym sposobem na zagwarantowanie, że wiadomość nie zostanie utracona, jest użycie transakcji: dla każdej wiadomości należy **wybrać transakcję, wysłać wiadomość i zatwierdzić**. Potwierdzenie pomyślnej transakcji jest odbierane po wywołaniu zdarzenia **OnAMQPTransactionOk**.



# AMQP Odbieranie wiadomości

Konsumenci odbierają wiadomości z kolejek. Aby pobierać wiadomości, musi istnieć kolejka. Po dodaniu nowego konsumenta, przy założeniu że w kolejce są już gotowe wiadomości, dostarczanie rozpoczyna się natychmiast.

Docelowa kolejka może być pusta w momencie rejestracji konsumenta. W takim przypadku pierwsze dostarczenie nastąpi po dodaniu nowych wiadomości do kolejki.

Odbieranie wiadomości jest zadaniem **asynchronicznym**, co oznacza, że za każdym razem, gdy nowa wiadomość może zostać dostarczona do kolejki konsumenta, jest ona automatycznie wypychana przez serwer do klienta. Alternatywną metodę **synchronicznego odbierania wiadomości** można znaleźć w odpowiedniej sekcji.

## Odbieranie

Metoda **Consume** tworzy nowego konsumenta w kolejce, a każda nowa wiadomość jest automatycznie dostarczana do klienta-konsumenta.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, nie może być dłuższa niż 255 znaków i nie może zaczynać się od "amq." (chyba że parametr `passive` ma wartość `true`).
- **ConsumerTag:** nazwa konsumenta, która musi być unikalna. Jeśli nie jest ustawiona, serwer tworzy nową.
- **NoLocal:** jeśli ma wartość `true`, konsument nigdy nie odbiera wiadomości opublikowanych na tym samym kanale.
- **NoAck:** jeśli `true`, oznacza, że serwer nie oczekuje potwierdzenia dla każdej dostarczonej wiadomości.
- **Exclusive:** jeśli wartość wynosi `true`, uniemożliwia innym konsumentom odbieranie wiadomości z tej kolejki.
- **NoWait:** jeśli `true`, serwer nie wyśle potwierdzenia do klienta.
- **Arguments:** ciąg znaków zawierający niestandardowe argumenty; wartości muszą być przekazane jako ciąg JSON, przykład: `{"x-dead-letter-exchange":"my-dlx"}`.

Wiadomości są dostarczane przez zdarzenie **OnAMQPBasicDeliver**.

```
AMQP->Consume("channel_name", "queue_name", "consumer_tag");

void OnAMQPBasicDeliver(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicDeliver *aDeliver,
    const TsgcAMQPMessageContent *aContent)
{
    DoLog("#AMQP_basic_deliver: " + aChannel + " " + aDeliver->ConsumerTag + " " +
        " " + aContent->Body->AsString);
}
```

Możliwe jest również wywołanie synchroniczne poprzez metodę **ConsumeEx**; metoda ta zwraca `true`, jeśli konsument został utworzony, i `false`, jeśli nie otrzymano potwierdzenia od serwera.

## Anuluj konsumowanie

Metoda służy do anulowania istniejącej kolejki konsumenta.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **ConsumerTag:** nazwa konsumenta.
- **NoWait:** jeśli true, serwer nie wyśle potwierdzenia do klienta.

```
AMQP->CancelConsume("channel_name", "consumer_tag");
```

```
private void OnAMQPBasicCancelConsume(TObject *Sender, const string aChannel, const string aConsumerTag)
{
    DoLog("#AMQP_basic_cancel_consume: " + aChannel + " " + aConsumerTag);
}
```

Wywołanie synchroniczne można wykonać, wywołując metodę **CancelConsumeEx**. Metoda ta zwraca wartość true, jeśli konsument został anulowany, lub false, jeśli nie nadeszło potwierdzenie od serwera.

## Polecenia | Pobieranie wiadomości AMQP

Pobieranie wiadomości jest zadaniem **Synchronicznym**, co oznacza, że to klient pyta serwer, czy w kolejce znajdują się wiadomości. Alternatywną metodę opisano w sekcji [Asynchroniczne odbieranie wiadomości](#).

### Pobierz wiadomość

Metoda **GetMessage** wysyła żądanie do serwera AMQP z pytaniem, czy w kolejce są dostępne wiadomości. Jeśli są, zostaną dostarczone przez zdarzenie **OnAMQPBasicGetOk**; jeśli kolejka jest pusta, wywoływane jest zdarzenie **OnAMQPBasicGetEmpty**.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **QueueName:** nazwa kolejki, której długość nie może przekraczać 255 znaków i która nie może zaczynać się od "amq." (z wyjątkiem sytuacji, gdy parametr `passive` jest ustawiony na `true`).
- **NoWait:** jeśli `true`, serwer nie wyśle potwierdzenia do klienta.

```
AMQP->GetMessage("channel_name", "queue_name");

private void OnAMQPBasicGetOk(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicGetOk *aGetOk,
    const TsgcAMQPMessageContent *aContent)
{
    DoLog("#AMQP_basic_GetOk: " + aChannel + " " + IntToStr(aGetOk->MessageCount) + " " + aContent->Body->AsString);
}

private void OnAMQPBasicGetEmpty(TObject *Sender, const string aChannel)
{
    DoLog("#AMQP_basic_GetEmpty: " + aChannel);
}
```

Wywołanie synchroniczne można wykonać, wywołując metodę **GetMessageEx**; metoda zwraca `True`, jeśli w kolejce są dostępne wiadomości, w przeciwnym razie zwraca `False`.

## Polecenia | AMQP QoS

AMQP umożliwia ustawienie poziomu QoS w celu ograniczenia liczby komunikatów wysyłanych przez serwer do klienta przed oczekiwaniem na potwierdzenie odbioru wiadomości.

### Ustaw QoS

Metoda **SetQoS** służy do ograniczania liczby wiadomości, które serwer wysyła do klienta AMQP.

Metoda przyjmuje następujące argumenty:

- **ChannelName:** nazwa kanału (musi być otwarty przed wywołaniem tej metody).
- **PrefetchSize:** jest to rozmiar okna w bajtach; serwer nie wysyła wiadomości do klienta, jeśli łączny rozmiar wszystkich aktualnie niepotwierdzonych wysłanych wiadomości powiększony o rozmiar następnej wiadomości do wysłania przekracza argument **PrefetchSize**. Wartość zero oznacza brak limitu.
- **PrefetchCount:** to maksymalna liczba niepotwierdzonych wiadomości już wysłanych i niepotwierdzonych; jeśli liczba ta zostanie przekroczona, serwer zaprzestaje wysyłania wiadomości do klienta.
- **Global:** jeśli true, QoS dotyczy wszystkich istniejących i nowych konsumentów połączenia. Jeśli false, QoS dotyczy wszystkich istniejących i nowych konsumentów kanału.

Odpowiedź z serwera jest odbierana przez zdarzenie **OnAMQPBasicQoS**.

```
AMQP->SetQoS("channel_name", 1024000, 100, false);

private void OnAMQPBasicQoS(TObject *Sender, const string aChannel,
    const TsgcAMQPFramePayload_Method_BasicQoS *aQoS)
{
    DoLog("#AMQP_basic_qos: " + aChannel + " " + IntToStr(aQoS->PrefetchSize) + " "
        + IntToStr(aQoS->PrefetchCount) + " " + BoolToStr(aQoS->Global));
}
```

Wywołanie synchroniczne można wykonać, wywołując metodę **SetQoSEx**; metoda ta zwraca true, jeśli żądanie zostało przetworzone, a w przeciwnym razie zwraca false.

# Polecenia | Transakcje AMQP

AMQP obsługuje dwa rodzaje transakcji:

1. Automatyczne transakcje, w których każda opublikowana wiadomość i potwierdzenie są przetwarzane jako osobna transakcja.
2. Lokalne transakcje serwera, w których serwer buforuje opublikowane wiadomości i potwierdzenia, a następnie zatwierdza je na żądanie klienta.

Klasa `Transaction` ("tx") zapewnia aplikacjom dostęp do drugiego typu, mianowicie transakcji serwera. Semantyka tej klasy jest następująca:

1. Aplikacja żąda transakcji serwera w każdym kanale, w którym chce mieć te transakcje (`Select`).
2. Aplikacja wykonuje pracę (`Publish`, `Ack`).
3. Aplikacja zatwierdza lub wycofuje pracę (`Commit`, `Roll-back`).
4. Aplikacja wykonuje pracę bez końca.

Transakcje obejmują opublikowane treści i potwierdzenia, a nie dostarczenia. Wycofanie transakcji nie ponownie kolejkuje ani nie dostarcza ponownie żadnych wiadomości; klient jest uprawniony do potwierdzenia tych wiadomości w kolejnej transakcji.

Metody `Transaction` umożliwiają grupowanie operacji publikowania i potwierdzania w atomowe jednostki pracy. Założeniem jest, że wszystkie żądania publikowania i potwierdzania wydane w ramach transakcji albo zakończą się pomyślnie, albo żadne z nich nie zostanie wykonane.

## Rozpocznij transakcję

Metoda **`StartTransaction`** rozpoczyna nową transakcję na serwerze; klient używa tej metody co najmniej raz na kanale przed zastosowaniem metod `Commit` lub `Rollback`. Zdarzenie **`OnAMQPTransactionOk`** jest wywoływane, gdy serwer potwierdzi użycie transakcji.

```
AMQP->StartTransaction("channel_name");
```

Wywołanie synchroniczne można wykonać, wywołując metodę **`StartTransactionEx`**. Metoda zwraca `true`, jeśli żądanie zostało przetworzone, w przeciwnym razie zwraca `false`.

## Zatwierdź transakcję

Metoda ta zatwierdza wszystkie publikacje wiadomości i potwierdzenia wykonane w bieżącej transakcji. Nowa transakcja rozpoczyna się natychmiast po zatwierdzeniu. Zdarzenie **OnAMQPTransactionOk** jest wywoływane, gdy serwer potwierdzi użycie transakcji.

```
AMQP->CommitTransaction("channel_name");
```

Można również wykonać wywołanie synchroniczne, wywołując metodę **CommitTransactionEx**; metoda ta zwraca true, jeśli żądanie zostało przetworzone, w przeciwnym razie wynik będzie false.

## Wycofaj transakcję

Ta metoda porzuca wszystkie publikacje wiadomości i potwierdzenia wykonane w bieżącej transakcji. Nowa transakcja rozpoczyna się bezpośrednio po wycofaniu. Należy pamiętać, że niepotwierdzonych wiadomości wycofanie nie dostarczy automatycznie ponownie; jeśli jest to wymagane, należy jawnie wywołać metodę recover. Zdarzenie **OnAMQPTransactionOk** jest wywoływane, gdy serwer potwierdza użycie transakcji.

```
AMQP->RollbackTransaction("channel_name");
```

Wywołanie synchroniczne można wykonać, wywołując metodę **RollbackTransactionEx**; metoda ta zwraca wartość true, jeśli żądanie zostało przetworzone, w przeciwnym razie zwróci wartość false.

# Protocol Kafka

**Apache Kafka** to otwartoźródłowa rozproszona platforma strumieniowania zdarzeń używana do publikowania, przechowywania i konsumowania strumieni rekordów. Producenci zapisują wiadomości do tematów, broker przechowuje je w partycjonowanych, tylko dołączanych logach, a konsumenci odczytują je z powrotem w kolejności, indywidualnie lub jako część skoordynowanej grupy konsumentów.

Kafka używa binarnego protokołu warstwy aplikacji nad TCP. Każdy temat jest podzielony na jedną lub więcej partycji, każdy rekord w partycji jest identyfikowany przez monotonicznie rosnący offset, a broker zachowuje rekordy, tak aby konsumenci mogli czytać we własnym tempie i odtwarzać historię w razie potrzeby. To sprawia, że Kafka dobrze nadaje się do przesyłania wiadomości o wysokiej przepustowości, event sourcingu, agregacji logów i przetwarzania strumieniowego.

## Funkcje

Klient Kafka `sgcWebSockets` komunikuje się z brokerem bezpośrednio przez natywny protokół Kafka, bez potrzeby proxy REST ani zewnętrznej biblioteki.

- Publikuj (produkuj) rekordy do dowolnego tematu i partycji, z kluczem, wartością i konfigurowalnymi potwierdzeniami.
- Konsumuj rekordy z zarządzaną grupą konsumentów lub bez grupy (czytaj wszystkie partycje bezpośrednio).
- Koordynacja grupy konsumentów: odkrywanie koordynatora, dołączanie/synchronizacja, przypisywanie partycji oraz zatwierdzanie/pobieranie offsetów.
- Zarządzanie offsetami: odpytywanie najwcześniejszego, najnowszego i zatwierdzonego offsetu.
- Administracja tematami: tworzenie i usuwanie tematów, żądanie metadanych klastra oraz wyświetlanie listy grup konsumentów.
- Odczytuje i zapisuje format wsadu rekordów v2 używany przez współczesne brokery Kafka.

## Obsługiwane wersje

Klient używa formatu wsadu rekordów v2 wprowadzonego w **Apache Kafka 0.11** i został przetestowany względem **Apache Kafka 3.x**. Domyślny port brokera to 9092.

## Komponenty

**`TsgcWSPClient_Kafka`**: to komponent klienta implementujący protokół **Apache Kafka**.

## Najczęstsze zastosowania

- **Połączenie**
  - [Client Kafka Connect](#)
  - [Client Kafka Disconnect](#)
- **Polecenia**
  - [Kafka Produce Messages](#)
  - [Kafka Consume Messages](#)
  - [Kafka Consumer Groups](#)
  - [Kafka Topics](#)
  - [Kafka Offsets](#)

# TsgcWSPClient\_Kafka

Klient Apache Kafka.

## Wprowadzenie

Klient **TsgcWSPClient\_Kafka** implementuje protokół **Apache Kafka**, komunikując się z brokerem bezpośrednio przez natywny binarny protokół Kafka po zwykłym połączeniu TCP. Potrafi produkować i konsumować rekordy, zarządzać grupami konsumentów, administrować tematami i odpytywać offsety, bez proxy REST ani zewnętrznej biblioteki.

## Połączenie

Klient Kafka jest komponentem protokołu, który działa na bazie [TsgcWebSocketClient](#). Ponieważ Kafka nie jest protokołem WebSocket, ustaw **Specifications.RFC6455** na false, aby klient łączył się jako zwykłe gniazdo TCP. Następnie przypisz klienta do komponentu Kafka i ustaw **Host** oraz **Port** brokera (domyślnie 9092).

Zachowanie producenta i konsumenta jest konfigurowane poprzez właściwość **KafkaOptions**:

- **KafkaOptions.ClientId**: identyfikuje Twoją aplikację wobec brokera.
- **KafkaOptions.Producer.Acks**: wymagane potwierdzenia, kafkaAcksNone, kafkaAcksLeader lub kafkaAcksAll.
- **KafkaOptions.Producer.TimeoutMs**: jak długo broker czeka na wymagane potwierdzenia.
- **KafkaOptions.Consumer.GroupId**: nazwa grupy konsumentów. Pozostaw pustą, aby konsumować bez grupy.
- **KafkaOptions.Consumer.OffsetReset**: kafkaOffsetEarliest lub kafkaOffsetLatest, czyli miejsce, od którego zaczyna czytać zupełnie nowa grupa.

```
TsgcWSPClient_Kafka *oKafka = new TsgcWSPClient_Kafka();
oKafka->KafkaOptions->ClientId = "my-cbuilder-app";
oKafka->OnKafkaConnect = OnKafkaConnect;
oKafka->OnKafkaMessage = OnKafkaMessage;
oKafka->OnKafkaProduce = OnKafkaProduce;

TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
oKafka->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "127.0.0.1";
oClient->Port = 9092;
oClient->Active = true;
```

## Produce Messages

Wywołaj metodę **Produce**, aby opublikować rekord. Wynik każdej publikacji jest zgłaszany przez zdarzenie **OnKafkaProduce**, łącznie z offsetem, pod którym zapisano rekord.

```
oKafka->KafkaOptions->Producer->Acks = kafkaAcksLeader;
oKafka->Produce("my-topic", "Hello Kafka", "key-1");
```



## Konsumowanie wiadomości

Ustaw **GroupId** oraz **OffsetReset**, zasubskrybuj jeden lub więcej tematów i wywołaj **Poll**, aby pobrać dostępne rekordy. Poll zwraca listę TsgcKafkaMessages, a także wywołuje zdarzenie OnKafkaMessage dla każdego rekordu. Wywołuj Poll wielokrotnie (na przykład z timera), aby odbierać dane w sposób ciągły.

```
oKafka->KafkaOptions->Consumer->GroupId = "my-group";
oKafka->KafkaOptions->Consumer->OffsetReset = kafkaOffsetEarliest;
String topics[1] = { "my-topic" };
oKafka->Subscribe(topics, 0);

TsgcKafkaMessages *oMessages = oKafka->Poll(1000);
try {
    for (int i = 0; i < oMessages->Count; i++)
        DoLog(oMessages->Items[i]->GetValueString());
} __finally {
    oMessages->Free();
}
```

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 18 metod publicznych.
- Zdarzenia — 6 zdarzeń, które możesz obsługiwać.

## Connection | Client Kafka Connect

Aby połączyć się z brokerem Apache Kafka, należy najpierw utworzyć [TsgcWebSocketClient](#) oraz [TsgcWSPClient\\_Kafka](#). Następnie dołącz komponent Kafka do klienta WebSocket.

Kafka używa zwykłego połączenia TCP, więc musisz ustawić właściwość `Specifications.RFC6455` obiektu `TsgcWebSocketClient` na `false`. Ustaw `Host` i `Port` brokera (domyślny port Kafka to 9092), opcjonalnie ustaw `KafkaOptions.ClientId`, a następnie ustaw `Active` na `true`, aby rozpocząć połączenie.

Po nawiązaniu połączenia z brokerem wywoływane jest zdarzenie **OnKafkaConnect**.

### Podstawowe użycie

```
oKafka = new TsgcWSPClient_Kafka();
oKafka->KafkaOptions->ClientId = "sgc-client";

oClient = new TsgcWebSocketClient();
oKafka->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 9092;
oClient->Active = true;

void OnKafkaConnect(TObject *Sender)
{
    ShowMessage("Connected to Kafka broker");
}
```

## Connection | Client Kafka Disconnect

Aby rozłączyć się z brokerem Apache Kafka, ustaw właściwość Active klienta WebSocket na false. Zamyka to bazowe połączenie TCP z brokerem.

Po zamknięciu połączenia wywoływane jest zdarzenie **OnKafkaDisconnect**.

### Podstawowe użycie

```
oClient->Active = false;

void OnKafkaDisconnect(TObject *Sender)
{
    ShowMessage("Disconnected from Kafka broker");
}
```

## Commands | Kafka Produce Messages

Użyj metody **Produce(topic, value, key, partition)**, aby opublikować rekord w temacie Kafka. Argumenty key i partition są opcjonalne; gdy partycja nie jest ustawiona, broker wybiera ją sam (na przykład poprzez haszowanie klucza). Wywołaj **ProduceBytes**, aby opublikować ładunek binarny.

Skonfiguruj gwarancję dostarczenia za pomocą `KafkaOptions.Producer.Acks`: `kafkaAcksNone` (wyślij i zapomnij), `kafkaAcksLeader` (czekaj na lidera partycji) lub `kafkaAcksAll` (czekaj na wszystkie zsynchronizowane repliki). Możesz również ustawić `Producer.TimeoutMs` oraz `Producer.Compression` (`kafkaCompressionNone` lub `kafkaCompressionGzip`).

Po wysłaniu rekordu zdarzenie **OnKafkaProduce** zgłasza temat, partycję, offset i kod błędu wyprodukowanego rekordu.

### Podstawowe użycie

```
oKafka->KafkaOptions->Producer->Acks = kafkaAcksAll;
oKafka->KafkaOptions->Producer->TimeoutMs = 5000;
oKafka->KafkaOptions->Producer->Compression = kafkaCompressionGzip;

oKafka->Produce("my-topic", "hello world", "key1");

void OnKafkaProduce(TObject *Sender, const String Topic, int Partition, __int64 Offset, int ErrorCode)
{
    if (ErrorCode == 0)
        ShowMessage("Produced to " + Topic + " at offset " + IntToStr(Offset));
    else
        ShowMessage("Produce error " + IntToStr(ErrorCode));
}
```

# Commands | Kafka Consume Messages

Aby konsumować rekordy, ustaw `KafkaOptions.Consumer.GroupId` i `KafkaOptions.Consumer.OffsetReset` (`kafkaOffsetEarliest` lub `kafkaOffsetLatest`), a następnie wywołaj **`Subscribe([topic])`** z jednym lub większą liczbą tematów.

Po zasubskrybowaniu wywołaj **`Poll(timeoutMs)`**, aby pobrać rekordy. `Poll` zwraca listę `TsgcKafkaMessages` i wywołuje zdarzenie **`OnKafkaMessage`** raz na każdy rekord. Zwykle wywołujesz `Poll` wielokrotnie, na przykład z timera, aby kontynuować konsumowanie nowych rekordów w miarę ich napływania.

Każdy rekord na liście to `TsgcKafkaMessage`. Iteruj po liście za pomocą `Count` i odczytuj każdy rekord za pomocą `GetKeyString`, `GetValueString`, `Topic`, `Partition` i `Offset`.

## Subscribe i Poll

```
oKafka->KafkaOptions->Consumer->GroupId = "my-group";
oKafka->KafkaOptions->Consumer->OffsetReset = kafkaOffsetEarliest;
oKafka->Subscribe(ARRAYOFCONST(("my-topic")));

// call repeatedly, e.g. from a TTimer::OnTimer
TsgcKafkaMessages *oMessages = oKafka->Poll(1000);
for (int i = 0; i < oMessages->Count; i++)
    ShowMessage(oMessages->Items[i]->GetKeyString() + " = " + oMessages->Items[i]->GetValueString());
```

## Zdarzenie OnKafkaMessage

Zamiast iterować po liście zwracanej przez `Poll`, możesz obsłużyć każdy rekord w zdarzeniu `OnKafkaMessage`.

```
void OnKafkaMessage(TObject *Sender, TsgcKafkaMessage *Message)
{
    ShowMessage(Message->Topic + ": " + Message->GetKeyString() + " = " + Message->GetValueString());
}
```

## Commands | Kafka Consumer Groups

Grupa konsumentów pozwala wielu konsumentom dzielić się pracą polegającą na odczycie tematu. Broker śledzi zatwierdzone offsety per GroupId i przypisuje partycje tematu członkom grupy, tak aby każda partycja była odczytywana w danym momencie tylko przez jednego członka. Ustaw grupę za pomocą `KafkaOptions.Consumer.GroupId`.

Dla nowej grupy bez zatwierdzonych offsetów `KafkaOptions.Consumer.OffsetReset` decyduje, gdzie zacząć: `kafkaOffsetEarliest` czyta od początku partycji, `kafkaOffsetLatest` czyta tylko nowe rekordy. Po zatwierdzeniu offsetów konsumpcja wznowia się od ostatniej zatwierdzonej pozycji.

Wywołaj **CommitSync**, aby zatwierdzić offsety rekordów zwróconych przez ostatni Poll, tak aby grupa wznowiła po nich w następnej sesji. Jeśli pozostawisz GroupId pusty, klient konsumuje wszystkie partycje bezpośrednio, bez dołączania do grupy i bez zatwierdzania offsetów na brokerze.

### Podstawowe użycie

```
oKafka->KafkaOptions->Consumer->GroupId = "orders-group";
oKafka->KafkaOptions->Consumer->OffsetReset = kafkaOffsetEarliest;
oKafka->Subscribe(ARRAYOFCONST(("orders")));
```

```
TsgcKafkaMessages *oMessages = oKafka->Poll(1000);
if (oMessages->Count > 0)
    oKafka->CommitSync();
```

## Polecenia | Tematy Kafka

Klient Kafka może administrować tematami na brokerze. Użyj **CreateTopic(name, partitions, replication)**, aby utworzyć nowy temat z podaną liczbą partycji i współczynnikiem replikacji, oraz **DeleteTopic(name)**, aby usunąć temat.

Użyj **GetMetadata([topics])**, aby pobrać informacje o klastrze i tematach, takie jak brokery, tematy i ich układ partycji. Przekaż pustą tablicę, aby zażądać metadanych całego klastra.

### Podstawowe użycie

```
// create a topic with 3 partitions and replication factor 1
oKafka->CreateTopic("my-topic", 3, 1);

// request metadata for the topic
oKafka->GetMetadata(ARRAYOFCONST(("my-topic")));

// delete the topic
oKafka->DeleteTopic("my-topic");
```

## Commands | Kafka Offsets

Klient Kafka może odpytywać i ustawiać offsety dla danego tematu i partycji. **GetEarliestOffset(topic, partition)** zwraca najstarszy dostępny offset, **GetLatestOffset(topic, partition)** zwraca następny offset do zapisania, a **GetCommittedOffset(topic, partition)** zwraca ostatni offset zatwierdzony przez grupę konsumentów. Wszystkie trzy zwracają Int64.

Użyj **CommitOffset(topic, partition, offset)**, aby zatwierdzić określony offset dla grupy konsumentów, tak aby konsumpcja wznowiła się od tej pozycji w następnej sesji.

### Podstawowe użycie

```
__int64 vEarliest = oKafka->GetEarliestOffset("my-topic", 0);
__int64 vLatest = oKafka->GetLatestOffset("my-topic", 0);
__int64 vCommitted = oKafka->GetCommittedOffset("my-topic", 0);

// commit a specific offset for the consumer group
oKafka->CommitOffset("my-topic", 0, vLatest);
```



# Protokół AMQP 1.0.0

AMQP (Advanced Message Queuing Protocol) 1.0.0 to protokół przesyłania wiadomości zaprojektowany do niezawodnej, asynchronicznej komunikacji między systemami rozproszonymi. Ułatwia wymianę wiadomości między aplikacjami lub komponentami w sposób rozłączony, umożliwiając im komunikację bez bezpośrednich zależności. Poniżej przedstawiono techniczny opis kluczowych aspektów AMQP 1.0.0:

- **Komunikacja zorientowana na wiadomości:** AMQP 1.0.0 jest skoncentrowany na koncepcji wiadomości. Wiadomości mogą przenosić dane, instrukcje lub polecenia i są podstawowymi jednostkami komunikacji.
- **Brokerzy wiadomości:** Protokół działa w modelu przesyłania wiadomości za pośrednictwem brokera. Brokerzy, którymi mogą być serwery lub podmioty pośredniczące, zarządzają routingiem i dostarczaniem wiadomości między producentami a konsumentami.
- **Kolejki i giełdy:** Kolejki to elementy przechowujące wiadomości w brokerze, gdzie wiadomości są tymczasowo składowane. Giełdy definiują reguły routingu wiadomości od producentów do kolejek na podstawie kryteriów takich jak zawartość wiadomości lub klucze routingu.
- **Adresy i łącza:** Adresy identyfikują miejsca docelowe wiadomości w infrastrukturze komunikacyjnej. Łącza to kanały komunikacyjne między nadawcą (producentem) a odbiorcą (konsumentem) powiązane z określonym adresem.
- **Sesje i połączenia:** Sesje reprezentują logiczny kanał komunikacji, umożliwiając wiele strumieni wiadomości w ramach jednego połączenia. Połączenia zarządzają ogólnym łączem komunikacyjnym między aplikacjami klienckimi a brokerem wiadomości.
- **Bezpieczeństwo:** AMQP 1.0.0 obsługuje różne mechanizmy bezpieczeństwa, w tym uwierzytelnianie i autoryzację, zapewniając bezpieczną komunikację między klientami a brokerami.
- **Niezależność od transportu:** Protokół jest zaprojektowany niezależnie od warstwy transportowej, co oznacza, że może działać przez różne transporty sieciowe, takie jak TCP, TLS lub WebSockets, zapewniając elastyczność wdrożenia.
- **Kontrola przepływu:** AMQP 1.0.0 zawiera mechanizmy kontroli przepływu, umożliwiające konsumentom wskazanie ich zdolności do obsługi przychodzących wiadomości z określoną szybkością. Zapobiega to przeciążeniu konsumentów dużą liczbą wiadomości.
- **Obsługa błędów:** Protokół określa mechanizmy obsługi błędów, w tym potwierdzanie i odrzucanie wiadomości, zapewniające niezawodność dostarczania wiadomości.
- **Uwierzytelnianie SASL:** Simple Authentication and Security Layer (SASL) służy do uwierzytelniania i zabezpieczania połączeń między klientami a brokerami.

Ogólnie rzecz biorąc, AMQP 1.0.0 zapewnia ustandaryzowany i interoperacyjny sposób komunikacji między różnymi komponentami i systemami oprogramowania w sposób luźno powiązany, co czyni go odpowiednim do różnych rozproszonych i korporacyjnych zastosowań.

## Komponenty

**TsgcWSPClient\_AMQP1:** składnik klienta implementujący protokół **AMQP 1.0.0**.

## Najczęstsze zastosowania

- **Połączenie**
  - [Połączenie klienta AMQP1](#)

- Rozłączenie klienta AMQP1
- Limit czasu bezczynności połączenia klienta AMQP1
- Stan połączenia klienta AMQP1
- Uwierzytelnianie klienta AMQP1
- Azure Service Bus klienta AMQP1

- **Polecenia**

- Sesje AMQP1
- Łączy AMQP1
- Łączy nadawcy AMQP1
- Łączy odbiornika AMQP1
- AMQP1 Wysyłanie wiadomości
- AMQP1 Odczyt wiadomości

# TsgcWSPClient\_AMQP1

AMQP 1.

## Wprowadzenie

Klient **TsgcWSPClient\_AMQP1** implementuje protokół **AMQP 1.0.0** zgodnie ze specyfikacją OASIS. Klient obsługuje połączenia przez zwykły TCP oraz WebSocket; obsługiwane są również połączenia TLS (szyfrowane).

## Konfiguracja

Klient AMQP 1.0.0 posiada właściwość **AMQPOptions**, w której można skonfigurować połączenie.

- **ChannelMax:** Wartość channel-max określa najwyższy numer kanału, który może być używany w połączeniu. Ta wartość plus jeden stanowi maksymalną liczbę sesji, które mogą być jednocześnie aktywne w połączeniu
- **ContainerId:** (opcjonalny) nazwa kontenera źródłowego, jednoznacznie identyfikuje połączenie na serwerze.
- **CreditSize:** domyślny rozmiar przepływu kredytowego.
- **IdleTimeout:** Limit czasu jest wyzwalany przez lokalnego partnera, gdy po przekroczeniu wartości progowej nie są odbierane żadne ramki. Limit czasu bezczynności jest mierzony w milisekundach i liczony od momentu odebrania ostatniej ramki.
- **MaxFrameSize:** maksymalny akceptowany rozmiar ramki.
- **MaxLinksPerSession:** maksymalna liczba łączy na sesję.
- **WindowSize:** domyślny rozmiar okna.

Uwierzytelnianie AMQP musi być skonfigurowane we właściwości **Authentication**.

- **AuthType:** typ uwierzytelniania
  - **amqp1authNone:** nie skonfigurowano.
  - **amqp1authSASLAnonymous:** uwierzytelnianie anonimowe
  - **amqp1authSASLPlain:** uwierzytelnianie użytkownik/hasło. Ten typ uwierzytelniania wymaga wypełnienia następujących właściwości:
    - Username
    - Password
  - **amqp1authSASLExternal:** uwierzytelnianie zewnętrzne

## Połączenie

Połączenie rozpoczyna się od nawiązania przez klienta (zazwyczaj aplikację lub usługę do przesyłania wiadomości) połączenia TCP z serwerem (brokerem wiadomości). Klient łączy się z portem serwera — zazwyczaj 5672 dla połączeń bez TLS i 5671 dla połączeń zabezpieczonych TLS. Po ustanowieniu połączenia TCP klient i serwer negocjują wersję protokołu AMQP. AMQP 1.0.0 obsługuje różne wersje, a podczas negocjacji obie strony uzgadniają użycie wersji 1.0.0.

Po negocjacji protokołu klient może wymagać uwierzytelnienia wobec serwera, w zależności od konfiguracji serwera. Mechanizmy uwierzytelniania mogą obejmować mechanizmy SASL (Simple Authentication and Security Layer), takie jak PLAIN, EXTERNAL lub inne obsługiwane przez serwer.

**Przykład:** połączenie z serwerem AMQP nasłuchującym na bezpiecznym porcie 5671 z użyciem poświadczeń SASL

```
// Creating AMQP client
oAMQP = new TsgcWSPClient_AMQP1(this);
// Setting AMQP authentication options
oAMQP->AMQPOptions->Authentication->AuthType = amqp1authSASLPlain;
oAMQP->AMQPOptions->Authentication->Username = L"sgc";
oAMQP->AMQPOptions->Authentication->Password = L"sgc";
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient->Specifications->RFC6455 = false;
// Setting WebSocket client properties
oClient->Host = L"www.esegece.com";
oClient->Port = 5671;
oClient->TLS = true;
// Assigning WebSocket client to AMQP client
oAMQP->Client = oClient;
// Activating WebSocket client
oClient->Active = true;
```

## Sesje

Po uwierzytelnieniu klient otwiera sesję AMQP. Sesja jest logicznym kontekstem komunikacji między klientem a serwerem. Sesje służą do grupowania powiązanych operacji przesyłania wiadomości. Aby utworzyć nową sesję, należy użyć metody **CreateSession**. Metoda pozwala ustawić nazwę sesji lub pozostawić ją pustą — wówczas komponent przypisze ją automatycznie.

Jeśli sesja została pomyślnie utworzona, zdarzenie **OnAMQPSessionOpen** zostanie wywołane ze szczegółami sesji.

```
oAMQP->CreateSession("MySession");
oAMQP->OnAMQPSessionOpen = AMQP1AMQPSessionOpen;
void __fastcall TMyClass::AMQP1AMQPSessionOpen(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1FrameBegin *
{
    ShowMessage("#session-open: " + aSession->Id);
}
```

## Łączy

W ramach sesji klient tworzy łączy do komunikacji z określonymi jednostkami, takimi jak kolejki, tematy lub inne zasoby udostępniane przez serwer. Łączy są dwukierunkowymi kanałami komunikacyjnymi używanymi do wysyłania i odbierania wiadomości.

Komponent może działać jako węzeł nadawczy i odbiorczy. Umożliwia tworzenie dowolnej liczby łączy dla każdej sesji, do limitu ustawionego we właściwości **MaxLinksPerSession**.

### Łączy nadawcy

Aby utworzyć nowe łączy nadawcy, należy użyć metody **CreateSenderLink** i przekazać nazwę sesji oraz opcjonalnie nazwę łączy nadawcy. Jeśli łączy zostanie pomyślnie utworzone, zostanie wywołane zdarzenie **OnAMQPLinkOpen**.

```
oAMQP->CreateSenderLink("MySession", "MySenderLink");
oAMQP->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

### Łączy odbiornika

Aby utworzyć nowe łączy odbiorcy, należy użyć metody **CreateReceiverLink** i przekazać nazwę sesji oraz opcjonalnie nazwę łączy odbiorcy. Po pomyślnym utworzeniu łączy wywoływane jest zdarzenie **OnAMQPLinkOpen**.

```
oAMQP->CreateSenderLink("MySession", "MySenderLink");
oAMQP->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

## Wysyłanie wiadomości

Po nawiązaniu sesji i utworzeniu łączy klient może rozpocząć operacje przesyłania wiadomości, takie jak wysyłanie wiadomości do miejsca docelowego. Aby wysłać wiadomość za pomocą łączy nadawcy, należy użyć metody `SendMessage`.

```
oAMQP->SendMessage("MySession", "MySenderLink", "My first AMQP Message");
```

## Odbieranie wiadomości

Domyślnie łączy odbiorcze są tworzone w trybie **automatycznym**, co oznacza, że każda nowa wiadomość jest dostarczana do klienta.

Jeśli łączy odbiornika zostało utworzone w **trybie ręcznym**, należy użyć metody synchronicznej **Get-Message**, aby pobrać dane i oczekiwać na nadejście nowej wiadomości.

W trybie automatycznym i ręcznym każde przychodzące zdarzenie wyzwala zdarzenie **OnAMQPMessage**.

```
void __fastcall OnAMQPMessageEvent(TObject *Sender, TsgcAMQP1Session *const aSession,
    TsgcAMQP1ReceiverLink *const aLink, TsgcAMQP1Message *const aMessage,
    TsgcAMQP1MessageDeliveryState &DeliveryState)
{
    ShowMessage(aMessage->ApplicationData->AMQPValue->Value);
}
```

## Dokumentacja

- Właściwości — 5 właściwości dostępnych na komponencie.
- Metody — 12 metod publicznych.
- Zdarzenia — 14 zdarzeń do obsługi.

## Połączenie | Klient AMQP1 Connect

Aby połączyć się z serwerem AMQP, należy najpierw utworzyć [TsgcWebSocketClient](#) oraz [TsgcWSPClient\\_AMQP1](#). Następnie należy dołączyć komponent AMQP1 do klienta WebSocket.

Po pomyślnym nawiązaniu połączenia wywoływane jest zdarzenie OnAMQPConnect.

### Podstawowe użycie

Łącz z serwerem AMQP 1.0.0 bez uwierzytelniania. Należy zdefiniować wartości właściwości AMQPOptions, wirtualny host, a następnie ustawić w [TsgcWebSocketClient](#) adres Host i Port serwera.

Jeśli używane jest połączenie TCP Plain, należy ustawić właściwość Specifications.RFC6455 klienta [TsgcWebSocketClient](#) na false.

```
// Creating AMQP client
oAMQP = new TsgcWSPClient_AMQP1(this);
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient->Specifications->RFC6455 = false;
// Setting WebSocket client properties
oClient->Host = L"amqp_host_address";
oClient->Port = 5672;
// Assigning WebSocket client to AMQP client
oAMQP->Client = oClient;
// Activating WebSocket client
oClient->Active = true;
```

### Uwierzytelnianie

Jeśli serwer wymaga uwierzytelnienia, należy użyć właściwości AMQP:Authentication w celu ustawienia wartości Username/Password oraz ustawić AuthType na wartość "amqp1authSASLPlain".

```
// Creating AMQP client
oAMQP = new TsgcWSPClient_AMQP1(this);
// Setting AMQP authentication options
oAMQP->AMQPOptions->Authentication->AuthType = amqp1authSASLPlain;
oAMQP->AMQPOptions->Authentication->Username = L"sgc";
oAMQP->AMQPOptions->Authentication->Password = L"sgc";
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient->Specifications->RFC6455 = false;
// Setting WebSocket client properties
oClient->Host = L"www.esegece.com";
oClient->Port = 5671;
```

```
oClient->TLS = true;  
// Assigning WebSocket client to AMQP client  
oAMQP->Client = oClient;  
// Activating WebSocket client  
oClient->Active = true;
```



## Połączenie | Client AMQP1 Disconnect

Klient może rozłączyć aktywne połączenie przy użyciu następujących metod:

### Wysyłanie powodu zamknięcia połączenia

Klient AMQP może poinformować serwer o zamiarze zamknięcia połączenia i podać powód zamknięcia. Aby zażądać zamknięcia połączenia od serwera, należy użyć metody `Close`.

```
oAMQP.Close('invalid-frame', "The received frame has an invalid format.");
```

### Await Close

Domyślnie metoda **Close** jest **asynchroniczna**, więc po jej wywołaniu wykonanie kodu jest kontynuowane. Jeśli chcesz **poczekać na zakończenie metody Close** i odebranie potwierdzenia wysłanego przez serwer, ustaw właściwość **Await** na `True` w parametrze `Options`.

```
void Close(const System::UnicodeString aCondition, const System::UnicodeString aDescription)
{
    TsgcAMQP1MethodOptions_Close *oOptions = new TsgcAMQP1MethodOptions_Close();
    try
    {
        oOptions->ErrorCondition = aCondition;
        oOptions->ErrorDescription = aDescription;
        oOptions->Await = true;
        AMQP1->Close(oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

### Zamykanie połączenia gniazda

Wystarczy ustawić właściwość `Active` [TsgcWebSocketClient](#) na `False`. Więcej informacji o [zamykaniu połączeń](#).

## Połączenie | Limit czasu bezczynności

Połączenia podlegają progowi **limitu czasu bezczynności**. Limit czasu jest wyzwalany przez klienta, gdy przez czas przekraczający wartość progową nie są odbierane żadne ramki z serwera. Limit czasu bezczynności jest mierzony w milisekundach i liczony od momentu odebrania ostatniej ramki. Po przekroczeniu progu komponent wysyła do serwera ramkę Close. Jeśli serwer nie odpowie w ciągu 10 sekund, klient zamknie gniazdo TCP.

Wartość limitu czasu bezczynności można skonfigurować we właściwości:

### **AMQPOptions.IdleTimeout**

Wartość ustawiona w tej właściwości zostanie wysłana do serwera podczas otwierania połączenia AMQP. Jeśli wartość jest większa od zera i mniejsza od połowy wartości `MaxInt`, zostanie włączony wewnętrzny timer sprawdzający, czy limit czasu bezczynności nie został przekroczony.

**Przykład:** ustaw wartość `IdleTimeout` na 60 sekund

```
AMQPOptions.IdleTimeout = 60000
```

## Connection | Connection State

Standard AMQP 1.0.0 definiuje następujące stany połączenia:

- **amqp1csUnknown:** stan początkowy.
- **amqp1csStart:** W tym stanie połączenie istnieje, ale nic nie zostało wysłane ani odebrane. Jest to stan, w którym implementacja znajdowałaby się bezpośrednio po wykonaniu operacji connect lub accept na gnieździe.
- **amqp1csHeaderReceived:** W tym stanie nagłówek połączenia został odebrany od peera, ale nagłówek połączenia nie został jeszcze wysłany.
- **amqp1csHeaderSent:** W tym stanie nagłówek połączenia został wysłany do peera, ale nie odebrano żadnego nagłówka połączenia.
- **amqp1csHeaderExchanged:** W tym stanie nagłówek połączenia został wysłany do partnera i odebrano od niego nagłówek połączenia.
- **amqp1csOpenPipe:** W tym stanie zarówno nagłówek połączenia, jak i ramka open zostały wysłane, ale nic nie zostało odebrane.
- **amqp1csOpenClosePipe:** W tym stanie zostały wysłane nagłówek połączenia, ramka open, potokowy ruch połączenia oraz ramka close, lecz nie odebrano jeszcze żadnych danych.
- **amqp1csOpenReceived:** W tym stanie nagłówki połączenia zostały wymienione. Odebrano ramkę open od partnera, ale ramka open nie została jeszcze wysłana.
- **amqp1csOpenSent:** W tym stanie wymieniono nagłówki połączenia. Ramka open została wysłana do partnera, ale nie otrzymano jeszcze żadnej ramki open.
- **amqp1csClosePipe:** W tym stanie nagłówki połączenia zostały wymienione. Ramka open, potokowy ruch połączenia oraz ramka close zostały wysłane, ale nie odebrano jeszcze ramki open od peera.
- **amqp1csOpened:** W tym stanie nagłówek połączenia i ramka open zostały zarówno wysłane, jak i odebrane.
- **amqp1csCloseReceived:** W tym stanie odebrano ramkę zamknięcia, wskazującą, że peer zainicjował zamknięcie AMQP. Nie oczekuje się już żadnych dalszych ramek przychodzących na połączenie; jednak ramki mogą być nadal wysyłane. Jeśli jest to pożądane, implementacja MOŻE w tym momencie wykonać półzamknięcie TCP w celu wyłączenia strony odczytu połączenia.
- **amqp1csCloseSent:** W tym stanie ramka close została wysłana do partnera. Zapisywanie czegokolwiek na połączeniu jest niedozwolone, jednak mogą nadchodzić przychodzące ramki. W razie potrzeby implementacja MOŻE wykonać TCP half-close w tym momencie, aby zamknąć stronę zapisującą połączenia.
- **amqp1csDiscarding:** Stan DISCARDING jest wariantem stanu CLOSE SENT, w którym zamknięcie jest wyzwolone przez błąd. W tym przypadku wszystkie przychodzące ramki na połączeniu MUSZĄ być cicho odrzucane do czasu odebrania ramki zamknięcia od peera.

- **amqp1csEnd:** W tym stanie żaden punkt końcowy nie może zapisywać czegokolwiek do połączenia.  
Połączenie można bezpiecznie zamknąć i odrzucić.

Klient AMQP posiada właściwość **ConnectionState**, w której można sprawdzić aktualny stan połączenia komponentu klienta.

# Połączenie | Uwierzytelnianie AMQP1

Komponent obsługuje następujące metody uwierzytelniania:

- **amqp1authNone:** brak metody uwierzytelniania podczas łączenia z serwerem.
- **amqp1authSASLAnonymous:** łączy się anonimowo.
- **amqp1authSASLPlain:** wartość domyślna; używa uwierzytelniania za pomocą nazwy użytkownika i hasła.
- **amqp1authSASLExternal:** aktualnie nieobsługiwane.

## Uwierzytelnianie SASL

Najpowszechniejszym typem uwierzytelniania jest typ **amqp1authSASLPlain**. Ten typ uwierzytelniania można włączyć w komponencie AMQP1, uzyskując dostęp do właściwości `AMQPOptions.Authentication`.

- **AuthType:** wybierz `amqp1authSASLPlain`
- **Username:** użytkownik używany do uwierzytelniania SASL.
- **Password:** wartość tajnego hasła używanego do uwierzytelniania SASL.

Wynik uwierzytelniania SASL można uzyskać po wywołaniu zdarzenia **OnAMQPSASLAuthentication**.

```
void __fastcall OnAMQP1SASLAuthentication(System::TObject* Sender, TsgcAMQP1SaslCode aCode,
    const System::UnicodeString aDescription, bool &Handled)
{
    ShowMessage("#sasl-authentication: " + aDescription);
}
```

# Połączenie | Azure MessageBus

AMQP (Advanced Message Queuing Protocol) to niezawodny system przesyłania wiadomości zaprojektowany w celu ułatwienia komunikacji między różnymi kontenerami w różnych węzłach. Standaryzuje zarówno protokół przesyłania wiadomości, jak i strukturę samych wiadomości, zapewniając spójną i niezawodną komunikację. Aby zagłębić się w podstawy AMQP, należy zapoznać się z przewodnikiem **Pierwsze kroki z AMQP**.

Komponent AMQP w bibliotece eSeGeCe umożliwia bezproblemową integrację z wiodącymi brokerami chmurowych wiadomości, w tym Amazon MQ i Azure Service Bus. Niniejszy przewodnik skupia się na używaniu komponentu AMQP do łączenia się z Azure Service Bus i pokazuje, jak zbudować aplikację wielodostępną zdolną do wydajnego wysyłania i odbierania wiadomości.

Komponent oferuje kompleksową implementację z obsługą kluczowych funkcji, takich jak kolejki, tematy i subskrypcje, co czyni go idealnym wyborem dla nowoczesnych aplikacji IoT i korporacyjnych.

## Konfiguracja Azure

Aby rozpocząć, należy utworzyć zasób **Service Bus** w portalu Azure. Po ustanowieniu zasobu należy za-notować **nazwę domeny zasobu**, ponieważ będzie ona niezbędna do integracji i konfiguracji.

Po pomyślnym utworzeniu **przestrzeni nazw** można ją zarządzać i monitorować ją bezpośrednio z poziomu **przeglądu przestrzeni nazw** w portalu Azure. Ten scentralizowany interfejs zapewnia dostęp do kluczowych narzędzi zarządzania i ustawień, umożliwiając bezproblemową administrację zasobem Service Bus.

Podczas korzystania z **uwierzytelniania SAS**, **nazwa użytkownika to nazwa polityki SAS**, a **hasło to klucz podstawowy lub pomocniczy**.

```
// Create TCP client
TsgcWebSocketClient* oClient = new TsgcWebSocketClient(nullptr);
oClient->Specifications->RFC6455 = false;
oClient->Host = "esegece.servicebus.windows.net";
oClient->Port = 5671;
oClient->TLS = true;
// Create AMQP1 protocol client
TsgcWScClient_AMQP1* oAMQP1 = new TsgcWScClient_AMQP1(nullptr);
oAMQP1->Specifications->RFC6455 = false;
oAMQP1->AMQPOptions->Authentication->AuthType = amqp1authSASLPlain;
oAMQP1->AMQPOptions->Authentication->Username = "RootManageSharedAccessKey";
oAMQP1->AMQPOptions->Authentication->Password = "BhJ78+w8kMXhS/eE/nBy0cRzodx9tipbi+ASbAXIaH8=";
oAMQP1->Client = oClient;
// Connect to the server
```

```
oClient->Active = true;
```

## Uwierzytelnianie Azure CBS

Azure Service Bus implementuje Claims-Based Security (CBS) przez AMQP w celu autoryzacji nadawców i odbiorców po wstępnym uzgadnianiu SASL. Klient otwiera łączy zarządzania do węzła **\$cbs** i wysyła żądanie **put-token** zawierające token Shared Access Signature (SAS) lub token JWT wystawiony przez Microsoft Entra ID. Po zweryfikowaniu tokenu przez brokera autoryzacja jest buforowana na czas jego ważności, a aplikacja może przystąpić do tworzenia łączy nadawcy i odbiorcy dla kolejek, tematów lub subskrypcji.

Klient AMQP1 automatyzuje ten przepływ za pomocą dwóch metod pomocniczych:

- **CreateAzureCbsSasToken** ustanawia parę łączy nadawcy/odbiorcy CBS, generuje token SAS dla docelowej jednostki i publikuje go do **\$cbs**. Należy użyć tej metody podczas uwierzytelniania za pomocą zasad dostępu współdzielonego.
- **CreateAzureCbsJWT** wykonuje tę samą wymianę CBS, ale przed wysłaniem JWT do **\$cbs** pobiera token dostępu z Microsoft Entra ID (Azure AD) przy użyciu grantu poświadczeń klienta.

Obie metody wymagają aktywnego połączenia AMQP i akceptują następujące parametry:

- **aName**: identyfikator wewnętrznie tworzonej pary łączy CBS.
- **aNameSpace** i **aEntityName**: przestrzeń nazw Service Bus (bez sufiksu *.servicebus.windows.net*) oraz ścieżka kolejki, tematu lub subskrypcji używana do budowania odbiorcy tokenu.
- **aKeyName** / **aKeyValue**: Nazwa zasady dostępu współdzielonego i klucz do tokenów SAS. Komponent podpisuje token i wysyła go przy użyciu typu tokenu *servicebus.windows.net:sastoken*.
- **aTenant**, **aApplicationId**, **aSecret**: identyfikator katalogu Microsoft Entra (Azure AD), identyfikator aplikacji (klienta) i sekret klienta używane do żądania JWT z przepływem poświadczeń klienta.
- **aListeningPort** (JWT): lokalny port HTTP dla przekierowania OAuth 2.0 (domyślnie 8080, jeśli nie podano).
- **aExpiration** i **aTimeout**: Czas życia wystawionego tokenu (w sekundach) i maksymalny czas oczekiwania (w milisekundach) na negocjację CBS.
- **aRaiseIfError**: Gdy wartość wynosi *True*, metoda zgłasza wyjątek w przypadku niepowodzenia pozyskania tokenu lub błędu odpowiedzi CBS.

Poniższe przykłady ilustrują sposób uwierzytelniania przy użyciu CBS przed wysyłaniem wiadomości.

```
// Create TCP client
TsgcWebSocketClient* oClient = new TsgcWebSocketClient(nullptr);
oClient->Specifications->RFC6455 = false;
```

```

oClient->Host = "esegece.servicebus.windows.net";
oClient->Port = 5671;
oClient->TLS = true;
// Create AMQP1 protocol client
TsgcWSCClient_AMQP1* oAMQP1 = new TsgcWSCClient_AMQP1(nullptr);
oAMQP1->Specifications->RFC6455 = false;
oAMQP1->AMQPOptions->Authentication->AuthType = amqp1authSASLAnonymous;
oAMQP1->Client = oClient;
// Connect and publish SAS token through CBS
oClient->Active = true;
// wait the client is connected
oAMQP1->CreateAzureCbsSasToken("cbs", "esegece", "queue1",
    "RootManageSharedAccessKey",
    "BhJ78+w8kMXhS/eE/nBy0cRzodx9tipbi+ASbAXIaH8=",
    3600, 10000, true);

```

Poniższy przykład skupia się wyłącznie na uwierzytelnianiu przy użyciu Microsoft Entra ID (Azure AD) z tokenami JWT. Pokazuje, jak zażądać tokenu w przepływie poświadczeń klienta i opublikować go w węźle **\$cbs** przed utworzeniem łączy do wysyłania lub odbierania wiadomości.

```

// Create TCP client
TsgcWebSocketClient* oClient = new TsgcWebSocketClient(nullptr);
oClient->Specifications->RFC6455 = false;
oClient->Host = "esegece.servicebus.windows.net";
oClient->Port = 5671;
oClient->TLS = true;
// Create AMQP1 protocol client
TsgcWSCClient_AMQP1* oAMQP1 = new TsgcWSCClient_AMQP1(nullptr);
oAMQP1->Specifications->RFC6455 = false;
oAMQP1->AMQPOptions->Authentication->AuthType = amqp1authSASLAnonymous;
oAMQP1->Client = oClient;
// Connect and publish JWT through CBS
oClient->Active = true;
oAMQP1->CreateAzureCbsJWT("cbs", "esegece", "queue1",
    "00000000-0000-0000-0000-000000000000", // Tenant ID
    "11111111-1111-1111-1111-111111111111", // Application ID
    "client-secret", 8080, 3600, 10000, true);

```



# Polecenia | Sesje AMQP1

W kontekście specyfikacji AMQP (Advanced Message Queuing Protocol) 1.0.0 sesja reprezentuje logiczny kontekst komunikacji między klientem a brokerem wiadomości. Poniżej omówiono poszczególne elementy sesji AMQP 1.0.0:

- **Kontekst logiczny:** Sesja ustanawia logiczny kontekst dla operacji przesyłania wiadomości między klientem AMQP (producentem lub konsumentem) a brokerem AMQP. Umożliwia grupowanie powiązanych operacji przesyłania wiadomości w ramach jednego połączenia.
- **Kanał komunikacyjny:** Sesje służą jako kanały komunikacyjne, przez które wysyłane i odbierane są wiadomości. Obejmują one wymianę wiadomości, potwierdzeń i mechanizmów kontroli przepływu.
- **Granice transakcyjne:** Sesje definiują granice transakcyjne dla operacji na wiadomościach. Umożliwiają grupowanie wielu operacji wysyłania lub odbierania wiadomości w jedną jednostkę atomową, zapewniając, że albo wszystkie operacje w sesji zostaną przetworzone pomyślnie, albo żadna nie zostanie przetworzona.
- **Flow Control:** Sesje obsługują mechanizmy kontroli przepływu regulujące szybkość wymiany wiadomości między klientem a brokerem. Kontrola przepływu zapobiega przeciążeniu zasobów obu stron, zapewniając wydajne i niezawodne dostarczanie wiadomości.
- **Zarządzanie cyklem życia:** Sesje mają cykl życia, który rozpoczyna się w momencie ich utworzenia i kończy po zamknięciu. Klienci mogą ustanawiać wiele sesji w ramach jednego połączenia, aby zrównoleglić przetwarzanie wiadomości lub izolować strumienie wiadomości.
- **Przydzielanie zasobów:** Sesje mogą być powiązane z określonymi zasobami brokera, takimi jak kolejki, tematy lub subskrypcje. Wiadomości wysyłane lub odbierane w ramach sesji są powiązane z tymi zasobami, co umożliwia ukierunkowane trasowanie i dostarczanie wiadomości.

Podsumowując, sesja AMQP 1.0.0 zapewnia logiczny kontekst wymiany wiadomości między klientem AMQP a brokerem, ułatwiając integralność transakcyjną, kontrolę przepływu i zarządzanie zasobami w systemie przesyłania wiadomości. Definiuje granice, w ramach których wykonywane są operacje przesyłania wiadomości, i pomaga zapewnić wydajną i niezawodną wymianę wiadomości.

## Otwarta sesja

Metoda **CreateSession** tworzy nową sesję o podanej nazwie (lub, jeśli jest pusta, sesję z losową nazwą); jeśli sesja już istnieje, zgłaszany jest wyjątek. Klient umożliwia tworzenie wielu sesji przy użyciu tego samego połączenia AMQP.

Po pomyślnym utworzeniu sesji wywoływane jest zdarzenie **OnAMQPSessionOpen**.

```
oAMQP->CreateSession("MySession");
oAMQP->OnAMQPSessionOpen = OnAMQPSessionOpenEvent;
void __fastcall TMyClass::OnAMQPSessionOpenEvent(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1FrameBegin *co
```

```
{
    ShowMessage("#session-open: " + aSession->Id);
}
```

Metoda **CreateSession** zwraca klasę **TsgcAMQP1Session** zawierającą informacje o sesji.

## Await Open Session

Domyślnie metoda **CreateSession** jest **asynchroniczna**, więc po jej wywołaniu kod kontynuuje działanie. Aby **począkać na zakończenie metody CreateSession** i otrzymanie potwierdzenia od serwera, należy ustawić właściwość **Await** na True w parametrze Options.

```
void OpenSession(const System::UnicodeString aSession)
{
    TsgcAMQP1MethodOptions_SessionOpen *oOptions = new TsgcAMQP1MethodOptions_SessionOpen();
    try
    {
        oOptions->Await = true;
        AMQP1->CreateSession(aSession, oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

## Zamknij sesję

Aby zamknąć istniejącą sesję, należy użyć metody **CloseSession**, podając nazwę sesji do zamknięcia.

Po pomyślnym zamknięciu sesji wywoływane jest zdarzenie **OnAQMPSessionClose**.

```
oAMQP->CloseSession("MySession");
oAMQP->OnAMQPSessionClose = OnAMQPSessionCloseEvent;
// OnAMQPSessionCloseEvent implementation
void __fastcall TMyClass::OnAMQPSessionCloseEvent(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1FrameEnd *con
{
    ShowMessage("#session-close: " + aSession->Id + " [" + IntToStr(aSession->Channel) + "] reason: " + aEnd->Error->Conditio
}
```

## Oczekuj na zamknięcie sesji

Domyślnie metoda **CloseSession** jest **asynchroniczna**, więc po jej wywołaniu kod kontynuuje działanie. Jeśli wymagane jest **oczekiwanie do zakończenia metody CloseSession** i odebrania potwierdzenia od serwera, należy ustawić właściwość **Await** na True w parametrze Options.

```
void CloseSession(const System::UnicodeString aSession)
{
    TsgcAMQP1MethodOptions_SessionClose *oOptions = new TsgcAMQP1MethodOptions_SessionClose();
    try
    {
        oOptions->Await = true;
        AMQP1->CloseSession(aSession, oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

## Polecenia | Łącza AMQP1

W specyfikacji AMQP (Advanced Message Queuing Protocol) 1.0.0 łącze reprezentuje jednokierunkowy kanał komunikacyjny między klientem AMQP a brokerem komunikatów. Poniżej przedstawiono szczegółowe omówienie łącza AMQP 1.0.0:

- **Kanał komunikacji:** Łącze służy jako ścieżka, przez którą wiadomości przepływają między nadawcą a odbiorcą AMQP. Umożliwia transmisję wiadomości w jednym kierunku — od nadawcy do odbiorcy lub odwrotnie.
- **Przepływ jednokierunkowy:** Każde łącze jest jednokierunkowe, co oznacza, że wiadomości mogą przepływać tylko w jednym kierunku. Jeśli wymagana jest komunikacja dwukierunkowa, należy ustanowić dwa łącza — jedno dla każdego kierunku.
- **Message Transfer:** Wiadomości są przesyłane przez łącza zgodnie z regułami protokołu AMQP. Wiadomości te mogą zawierać ładunki, właściwości wiadomości oraz dodatkowe metadane wymagane do komunikacji.
- **Wiązanie zasobów:** Łącza są powiązane z określonymi zasobami w brokerze AMQP, takimi jak kolejki, tematy lub giełdy. Wiadomości wysyłane lub odbierane przez łącza są kierowane do tych zasobów lub z nich pochodzą.
- **Kontrola przepływu:** Łącza obsługują mechanizmy kontroli przepływu regulujące szybkość wysyłania i odbierania wiadomości. Kontrola przepływu zapewnia, że ani nadawca, ani odbiorca nie jest przeciążony wolumenem wymienianych wiadomości.
- **Zarządzanie cyklem życia:** Łącza posiadają cykl życia, który rozpoczyna się w momencie ich ustanowienia i kończy po ich zamknięciu. Mogą być tworzone dynamicznie w miarę potrzeb i zamykane, gdy nie są już wymagane.
- **Addressing:** Łącza są identyfikowane przez unikalne adresy określające źródłowe i docelowe punkty końcowe komunikacji. Adresy te umożliwiają klientom i brokerom identyfikację i nawiązywanie połączeń z właściwymi punktami końcowymi.
- **Granice transakcyjne:** Łącza definiują granice transakcyjne dla operacji na wiadomościach. Umożliwiają grupowanie wielu operacji wysyłania lub odbierania wiadomości w jedną niepodzielną jednostkę, zapewniając spójność i niezawodność dostarczania wiadomości.

Podsumowując, łącza AMQP 1.0.0 zapewniają środki do jednokierunkowej komunikacji między klientami AMQP a brokerami, umożliwiając przesyłanie wiadomości z obsługą kontroli przepływu, wiązania zasobów, adresowania i integralności transakcyjnej w systemie przesyłania wiadomości. Stanowią one podstawowe elementy składowe wymiany wiadomości w protokole AMQP.

Istnieją 2 typy łączy:

- **Łącza nadawcy:** te łącza są używane do wysyłania wiadomości.
- **Łącza odbiorcze:** te łącza służą do odbierania wiadomości.

Za każdym razem, gdy nowe łącze zostanie utworzone lub usunięte, wywoływane są następujące zdarzenia:

- **OnAMQPLinkOpen:** zdarzenie jest wywoływane podczas tworzenia nowego łącza. Właściwość `aLink.Mode` służy do sprawdzenia, czy łącze jest w trybie odbiorcy, czy nadawcy.
- **OnAMQPLinkClose:** zdarzenie wywoływane po zamknięciu łącza.

## Polecenia | Łączy nadawcy AMQP1

W protokole AMQP 1.0.0 **łącze nadawcy** jest **kanalem komunikacyjnym** ustanowionym między klientem AMQP a serwerem AMQP w celu **wysyłania wiadomości**. Działa w kontekście sesji AMQP, która reprezentuje logiczny kanał komunikacji między klientem a serwerem.

### Utwórz łącze nadawcy

Aby **utworzyć nowe łącze nadawcy**, należy wywołać metodę **CreateSenderLink**, która przyjmuje następujące parametry:

- **Session:** nazwa sesji, do której zostanie dołączone łącze nadawcy.
- **Name:** (opcjonalna) nazwa łącza nadawcy; jeśli nie zostanie ustawiona, zostanie automatycznie przypisana losowa nazwa.
- **Target:** (opcjonalnie) można określić miejsce docelowe, pod którym wiadomości mają być odbierane na zdalnym hoście, ustawiając parametr „target”. Jednak w pewnych scenariuszach podanie celu może nie być wymagane. W takich przypadkach wystarczy podać pusty ciąg znaków.
- **SndSettleMode:** (domyślnie mixed) AMQP oferuje możliwość negocjowania gwarancji dostarczania za pomocą mechanizmu rozliczania wiadomości. Podczas nawiązywania łącza nadawca i odbiorca negocjują i uzgadniają tryb rozliczania (osobny dla każdej roli). Nadawcy działają w jednym z następujących trybów:
  - **amqp1ssmSettled:** Wiadomość jest uważana za dostarczoną i potwierdzoną w momencie wysłania.
  - **amqp1ssmUnsettled:** Wiadomość nie jest rozliczona, dopóki odbiorca jej jawnie nie zaakceptuje lub nie odrzuci. Umożliwia to większą kontrolę nad przetwarzaniem i obsługą wiadomości.
  - **amqp1ssmMixed:** W ramach jednej sesji AMQP można stosować kombinację trybów rozliczonych i nierozliczonych. Parametr **MessageOptions** metody **SendMessage** umożliwia konfigurację, czy wiadomość jest rozliczona, czy nie.

Gdy łącze nadawcze zostanie pomyślnie utworzone, zostanie wywołane zdarzenie **OnAMQPLinkOpen**.

```
oAMQP1->CreateSenderLink("MySession", "MySenderLink");
oAMQP1->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink, T
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

### Oczekuj na utworzenie łącza nadawcy

Domyślnie metoda **CreateSenderLink** jest **asynchroniczna**, tzn. po jej wywołaniu kod kontynuuje wykonanie. Aby **począkać na ukończenie metody CreateSenderLink** i odebranie potwierdzenia od serwera, należy ustawić właściwość **Await** na True w parametrze Options.

```
void CreateSenderLink(const System::UnicodeString aSession, const System::UnicodeString aSender)
{
    TsgcAMQP1MethodOptions_CreateSenderLink *oOptions = new TsgcAMQP1MethodOptions_CreateSenderLink();
    try
    {
        oOptions->Await = true;
        AMQP1->CreateSenderLink(aSession, aSender, L"", oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

## Wysyłanie wiadomości

Aby wysłać nową wiadomość, należy wywołać metodę **SendMessage**, która przyjmuje następujące parametry:

- **Session:** nazwa sesji.
- **Link:** nazwa łącza nadawcy.
- **Text:** treść wiadomości tekstowej.

```
oAMQP1->SendMessage("MySession", "MySenderLink", "My first AMQP Message");
```

## Wysyłanie wiadomości w trybie mieszanym

Gdy łącze nadawcy jest utworzone w trybie mieszanym (domyślnym), podczas wysyłania wiadomości użytkownik może określić, czy wiadomość ma być **rozliczona** czy **nie**. Należy użyć parametru **MessageOptions**, aby określić, czy wiadomość jest rozliczona.

```
TsgcAMQP1MessageOptions* oMessageOptions = new TsgcAMQP1MessageOptions();
try
{
    oMessageOptions->Settled = true;
    oAMQP1->SendMessage("MySession", "MySenderLink", "MyMessage", "message-id", oMessageOptions);
}
__finally
{
    delete oMessageOptions;
}
```

## Zamknij łącze nadawcy

Aby zamknąć istniejące łącze nadawcze, należy wywołać metodę **CloseLink**, która przyjmuje następujące parametry:

- Session: nazwa sesji zawierającej łącze.
- Link: nazwa łącza nadawcy.
- Błąd: (opcjonalny) tutaj można podać przyczynę zamknięcia łącza.

Gdy łącze nadawcze zostanie pomyślnie zamknięte, zostanie wywołane zdarzenie **OnAMQPLinkClose**.

```
oAMQP->CloseLink("MySession", "MySenderLink");
void __fastcall TForm1::OnAMQPLinkCloseEvent(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink,
{
    ShowMessage("#link-close: " + aLink->Name);
}
```

## Oczekiwanie na zamknięcie łącza nadawcy

Domyślnie metoda **CloseLink** jest **asynchroniczna**, więc po jej wywołaniu kod wykonuje się dalej. Jeśli wymagane jest **oczekiwanie do zakończenia metody CloseLink** i otrzymania potwierdzenia od serwera, należy ustawić właściwość **Await** na True w parametrze Options.

```
void CloseSenderLink(const System::UnicodeString aSession, const System::UnicodeString aSenderLink)
{
    TsgcAMQP1MethodOptions_CloseLink *oOptions = new TsgcAMQP1MethodOptions_CloseLink();
    try
    {
        oOptions->Await = true;
        AMQP1->CloseLink(aSession, aSenderLink, oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```



## Polecenia | Łączy odbiornika AMQP1

W protokole AMQP 1.0.0 **łącze odbiorcy** (Receiver Link) to **kanał komunikacyjny** ustanowiony między klientem AMQP a serwerem AMQP w celu **odbierania wiadomości**. Działa w kontekście sesji AMQP, reprezentującej logiczny kanał komunikacji między klientem a serwerem.

### Tworzenie łącza odbiorcy

Aby **utworzyć nowe łącze odbiornika**, należy wywołać metodę **CreateReceiverLink**, która przyjmuje następujące parametry:

- **Session:** nazwa sesji, do której zostanie dołączone łącze nadawcy.
- **Name:** (opcjonalna) nazwa łącza nadawcy; jeśli nie zostanie ustawiona, zostanie automatycznie przypisana losowa nazwa.
- **Source:** (opcjonalnie) źródło można skonfigurować, aby wskazać lokalizację węzła na zdalnym hoście, który ma pełnić rolę nadawcy. W niektórych sytuacjach podanie tego adresu może nie być wymagane. W takich przypadkach wystarczy podać pusty ciąg jako wartość parametrów.
- **ReadMode:** (domyślnie `amqp1srmAuto`) Łącza odbiorcze mogą działać w jednym z dwóch trybów odbioru wiadomości:
  - **amqp1srmAuto:** Tryb automatyczny — w tym trybie odbiornik aktywnie dąży do niezwłocznego odbioru wiadomości, gdy tylko staną się dostępne. Automatycznie nasłuchuje i odbiera wiadomości bez konieczności jawnego powiadamiania przy każdym przybyciu nowej wiadomości.
  - **amqp1srmManual:** Tryb oparty na pobieraniu — w tym trybie odbiornik pobiera nową wiadomość wyłącznie na wyraźne polecenie. W odróżnieniu od trybu automatycznego odbiornik nie nasłuchuje aktywnie nowych wiadomości, lecz czeka na ręczne instrukcje pobrania kolejnej wiadomości.
- **RcvSettleMode:** (domyślnie `amqp1rsmFirst`) Łącza odbiornika działają w jednym z następujących trybów:
  - **amqp1rsmFirst:** Po nadejściu wiadomości są one natychmiast przetwarzane i potwierdzane. Jeśli wiadomość nie została jeszcze potwierdzona w momencie wysłania, nadawca zostanie poinformowany o jej odebraniu.
  - **amqp1rsmSecond:** Przychodzące wiadomości są potwierdzane dopiero po wcześniejszym potwierdzeniu przez nadawcę. Dodatkowo nadawca otrzymuje powiadomienie o odebraniu wiadomości, pod warunkiem że nie została ona jeszcze potwierdzona w momencie wysłania.

Po pomyślnym utworzeniu łącza odbiorczego wywoływane jest zdarzenie **OnAMQPLinkOpen**.

```
oAMQP1->CreateReceiverLink("MySession", "MyReceiverLink");
oAMQP1->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink, T
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

## Await Create Receiver Link

Domyślnie metoda **CreateReceiverLink** jest **asynchroniczna**, dlatego po jej wywołaniu wykonanie kodu jest kontynuowane. Jeśli wymagane jest **oczekiwanie na zakończenie metody CreateReceiverLink** i odebranie potwierdzenia wysłanego przez serwer, należy ustawić właściwość **Await** na True w parametrze Options.

```
void CreateReceiverLink(const System::UnicodeString aSession, const System::UnicodeString aReceiver)
{
    TsgcAMQP1MethodOptions_CreateReceiverLink *oOptions = new TsgcAMQP1MethodOptions_CreateReceiverLink();
    try
    {
        oOptions->Await = true;
        AMQP1->CreateReceiverLink(aSession, aReceiver, L"", oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

## Synchronizacja wiadomości

Gdy łącze odbiornika działa w trybie ręcznego ReadMode, należy wywołać metodę **GetMessage**, aby pobrać nowe wiadomości. Metoda ta jest synchroniczna, co oznacza, że czeka do upływu limitu czasu (domyślnie 10 sekund). Po wywołaniu metody komponent zwiększa kredyt o jedną jednostkę i czeka na nadejście nowej wiadomości lub upływanie limitu czasu. Jeśli żadna wiadomość nie nadejdzie, kredyt jest ponownie ustawiany na zero.

Metoda **GetMessage** posiada następujące parametry:

- **Session:** nazwa sesji zawierającej łącze.
- **Link:** nazwa łącza odbiorczego.
- **Timeout:** (domyślnie 1000 = 10 sekund) maksymalny czas oczekiwania funkcji na nową wiadomość.

## Zamknij łącze odbiornika

Aby zamknąć istniejące łącze odbiornika, należy wywołać metodę **CloseLink** z następującymi parametrami:

- **Session:** nazwa sesji zawierającej łącze.
- **Link:** nazwa łącza odbiorczego.
- **Error:** (opcjonalnie) tutaj można podać powód zamknięcia łącza.

Po pomyślnym zamknięciu łącza odbiornika zostanie wywołane zdarzenie **OnAMQPLinkClose**.

```
oAMQP1->CloseLink("MySession", "MyReceiverLink");
void __fastcall TForm1::OnAMQPLinkCloseEvent(TObject *Sender, TsgcAMQP1Session *const aSession, TsgcAMQP1Link *const aLink,
{
    ShowMessage("#link-close: " + aLink->Name);
}
```

## Oczekiwanie na zamknięcie łącza odbiornika

Domyślnie metoda **CloseLink** jest **asynchroniczna**, więc po jej wywołaniu kod wykonuje się dalej. Jeśli wymagane jest **oczekiwanie do zakończenia metody CloseLink** i otrzymania potwierdzenia od serwera, należy ustawić właściwość **Await** na True w parametrze Options.

```
void CloseReceiverLink(const System::UnicodeString aSession, const System::UnicodeString aReceiverLink)
{
    TsgcAMQP1MethodOptions_CloseLink *oOptions = new TsgcAMQP1MethodOptions_CloseLink();
    try
    {
        oOptions->Await = true;
        AMQP1->CloseLink(aSession, aReceiverLink, oOptions);
    }
    __finally
    {
        delete oOptions;
    }
}
```

# AMQP1 | Wysyłanie wiadomości

Należy najpierw zapoznać się z sekcją [AMQP1 Sender Links](#), aby dowiedzieć się, jak utworzyć łącze nadawcy.

## Wyślij wiadomość

Użyj metody **SendMessage**, podając nazwę sesji i łącza SenderLink, aby wysłać wiadomość tekstową do serwera AMQP1. Metoda ma następujące parametry:

- **Session:** nazwa sesji.
- **Link:** nazwa łącza nadawcy.
- **Text:** treść wiadomości.
- **MessageId:** (opcjonalny) identyfikator wiadomości; może być używany w trybie unsettled, aby sprawdzić, czy serwer przetworzył wiadomość.
- **Options:** (opcjonalne) umożliwia dostosowanie niektórych opcji podczas wysyłania wiadomości.
  - **Settled:** przy korzystaniu z łącza nadawcy w trybie mieszanym podczas wysyłania wiadomości właściwość Settled można dostosować.
  - **Await:** jeśli wiadomość jest nierozliczona i wartość wynosi true, kod będzie oczekiwał do momentu przetworzenia wiadomości przez serwer lub przekroczenia limitu czasu.
  - **Timeout:** wartość w milisekundach, gdy Await ma wartość true (domyślnie 10000).
  - **RaiseTimeoutException:** jeśli limit czasu zostanie przekroczony, zgłaszany jest wyjątek (domyślnie true).

```
oAMQP1->SendMessage("MySession", "MySenderLink", "My first AMQP Message");
```

## Oczekuj na wysłanie wiadomości

Domyślnie metoda **SendMessage** jest asynchroniczna podczas wysyłania wiadomości bez potwierdzenia. Ustawienie właściwości **Await** na true powoduje, że klient czeka na potwierdzenie od serwera o przetworzeniu wiadomości.

```
void SendMessageAwait(const System::UnicodeString aSession, const System::UnicodeString aSenderLink, const System::UnicodeString aText)
{
    TsgcAMQP1MethodOptions_SendMessageAck *oOptions = new TsgcAMQP1MethodOptions_SendMessageAck();
    try
    {
        oOptions->Settled = false;
        oOptions->Await = true;
        AMQP1->SendMessage(aSession, aSenderLink, aText, L"message-id", oOptions);
    }
}
```

```

    }
    __finally
    {
        delete oOptions;
    }
}

```

## Zdarzenia

Podczas wysyłania wiadomości dostępne są 2 zdarzenia umożliwiające sprawdzenie, kiedy wiadomość została wysłana i czy została przetworzona przez serwer (w trybie unsettled).

- **OnAMQPMessageSent:** zdarzenie wywoływane po wysłaniu wiadomości do serwera. Podczas wywoływania metody `SendMessage` wiadomość jest przechowywana w wewnętrznej kolejce i przetwarzana przez wątek pomocniczy; po wysłaniu wiadomości wywoływane jest to zdarzenie.
- **OnAMQPMessageSentAck:** zdarzenie to jest wywoływane, gdy klient otrzymuje potwierdzenie, że wiadomość została przetworzona przez serwer AMQP1.

```

void __fastcall OnAMQPMessageSentAck(System::TObject* Sender, TsgcAMQP1Session* const aSession,
    TsgcAMQP1SenderLink* const aLink, const System::UnicodeString aMessageId,
    TsgcAMQP1FrameDeliveryStates aDeliveryState, TsgcAMQP1FrameDisposition aDisposition)
{
    System::UnicodeString vMessageId = aMessageId;
    switch (aDeliveryState)
    {
        case amqp1fdtsAccepted:
            ShowMessage("#msg-accepted: " + vMessageId);
            break;
        case amqp1fdtsRejected:
            ShowMessage("#msg-rejected: " + vMessageId + " " +
                TsgcAMQP1FrameRejected(aDisposition.State)->Error->Condition + " " +
                TsgcAMQP1FrameRejected(aDisposition.State)->Error->Description);
            break;
        case amqp1fdtsReleased:
            ShowMessage("#msg-released: " + vMessageId);
            break;
        case amqp1fdtsModified:
            ShowMessage("#msg-modified: " + vMessageId + " " +
                TsgcAMQP1FrameModified(aDisposition.State)->MessageAnnotations);
            break;
        case amqp1fdtsReceived:
            ShowMessage("#msg-received: " + vMessageId);
            break;
    }
}

```

## AMQP1 | Odczyt wiadomości

Za każdym razem, gdy zostaje odebrana nowa wiadomość, wywoływane jest zdarzenie **OnAMQPMessage**.

Instancja `TsgcAMQP1Message` zawiera odebraną wiadomość. Dostęp do wiadomości tekstowej można uzyskać za pomocą właściwości **`aMessage.ApplicationData.AMQPValue.Value`**.

Aby określić wynik dostarczenia, użyj parametru **`DeliveryState`**. Domyślnie wszystkie wiadomości mają stan zaakceptowany, ale można ustawić jeden z następujących stanów:

- **`amqp1mdtsAccepted`**: Wiadomość została pomyślnie przetworzona.
- **`amqp1mdtsRejected`**: przetwarzanie wiadomości nie powiodło się. Ustaw błąd za pomocą właściwości `DeliveryState.Rejected`.
- **`amqp1mdtsReleased`**: Wiadomość nie została i nie zostanie przetworzona.
- **`amqp1mdtsModified`**: Tak samo jak `amqp1mdtsReleased`, ale można dodać dodatkowe dane przy użyciu właściwości `DeliveryState.Modified`.

```
void __fastcall OnAMQPMessage(System::TObject* Sender, TsgcAMQP1Session* const aSession,
    TsgcAMQP1ReceiverLink* const aLink, TsgcAMQP1Message* const aMessage,
    TsgcAMQP1MessageDeliveryState &DeliveryState)
{
    if (aMessage->ApplicationData->AMQPValue->Value == "xxx")
    {
        DeliveryState.State = amqp1mdtsRejected;
        DeliveryState.Rejected->Error->Condition = "amqp-error-processing";
        DeliveryState.Rejected->Error->Description = "Value received was not the expected.";
    }
    else
    {
        DeliveryState.State = amqp1mdtsAccepted;
    }
}
```

# Protokół STOMP

STOMP to Simple (lub Streaming) Text Orientated Messaging Protocol. STOMP zapewnia interoperacyjny format sieciowy, dzięki któremu klienty STOMP mogą komunikować się z dowolnym brokerem STOMP, zapewniając łatwą i powszechną interoperacyjność wiadomości między wieloma językami, platformami i brokerami.

Komponenty klienta STOMP obsługują następujące wersje STOMP: 1.0, 1.1 i 1.2.

## Komponenty

**TsgcWSPClient\_STOMP**: ogólny klient protokołu STOMP, umożliwiający połączenie z dowolnym serwerem STOMP.

**TsgcWSPClient\_STOMP\_RabbitMQ**: Klient STOMP dla brokera RabbitMQ.

**TsgcWSPClient\_STOMP\_ActiveMQ**: Klient STOMP dla brokera ActiveMQ.

# TsgcWSPClient\_STOMP

Jest to komponent protokołu klienta STOMP.

## Wprowadzenie

Jest to komponent protokołu klienta STOMP. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

## Dokumentacja

- Właściwości — 8 właściwości dostępnych w komponencie.
- Metody — 11 metod publicznych.
- Zdarzenia — 6 zdarzeń do obsługi.



# TsgcWSPClient\_STOMP\_RabbitMQ

Jest to komponent protokołu STOMP po stronie klienta dla brokera RabbitMQ.

## Wprowadzenie

Jest to komponent protokołu klienta STOMP dla brokera RabbitMQ. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketClient](#) za pomocą właściwości Client.

## Miejsca docelowe

Specyfikacja STOMP nie określa, jakie typy miejsc docelowych musi obsługiwać broker — zamiast tego wartość nagłówka destination w ramach SEND i MESSAGE jest specyficzna dla brokera. Adapter STOMP RabbitMQ obsługuje szereg różnych typów miejsc docelowych:

- **Topic:** SEND i SUBSCRIBE do tymczasowych i trwałych tematów.
- **Queue:** SEND i SUBSCRIBE do kolejek zarządzanych przez bramkę STOMP.
- **QueueOutside:** SEND i SUBSCRIBE do kolejek utworzonych poza bramką STOMP.
- **TemporaryQueue:** tworzenie tymczasowych kolejek (wyłącznie w nagłówkach reply-to).
- **Exchange:** SEND do dowolnych kluczy routingu i SUBSCRIBE do dowolnych wzorców powiązań.

## Dokumentacja

- Właściwości — 8 właściwości dostępnych w komponencie.
- Metody — 25 metod publicznych.
- Zdarzenia — 6 zdarzeń do obsługi.

# TsgcWSPClient\_STOMP\_ActiveMQ

Jest to komponent protokołu klienta STOMP dla brokera ActiveMQ.

## Wprowadzenie

Jest to komponent protokołu klienta STOMP dla brokera ActiveMQ. Należy umieścić ten komponent w formularzu i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

## Miejsca docelowe

Specyfikacja STOMP nie określa, jakie typy miejsc docelowych musi obsługiwać broker; wartość nagłówka destination w ramach SEND i MESSAGE jest specyficzna dla brokera. Adapter Active STOMP obsługuje wiele różnych typów miejsc docelowych:

- **Topic:** SEND i SUBSCRIBE do tymczasowych i trwałych tematów.
- **Queue:** SEND i SUBSCRIBE do kolejek zarządzanych przez bramkę STOMP.

## Opcje publikowania

Należy pamiętać, że STOMP jest zaprojektowany tak, aby był jak najprostszy — dzięki temu każdy język skryptowy czy platforma mogą bez większego wysiłku wymieniać wiadomości z innymi. STOMP umożliwia dodawanie niestandardowych nagłówków do każdego żądania, takich jak wysyłanie i odbieranie wiadomości. ActiveMQ posiada kilka rozszerzeń protokołu Stomp, które pozwalają na obsługę semantyki JMS przez klientów Stomp. Producent JMS OpenWire może wysyłać wiadomości do konsumenta Stomp, a producent Stomp może wysyłać wiadomości do konsumenta JMS OpenWire. Konfiguracje Stomp-to-Stomp mogą korzystać z bogatszej kontroli wiadomości JMS.

STOMP obsługuje następujące standardowe właściwości JMS dla wysyłanych wiadomości:

- **CorrelationId:** Właściwi konsumenci powinni dodawać ten nagłówek do wszelkich wysyłanych odpowiedzi.
- **Expires:** Czas wygaśnięcia wiadomości.
- **JMSXGroupID:** określa grupy wiadomości.
- **JMSXGroupSeq:** Opcjonalny nagłówek określający numer sekwencyjny w grupach wiadomości.
- **Persistent:** Określa, czy wiadomość jest trwała.
- **Priority:** Priorytet wiadomości.

- **ReplyTo:** Miejsce docelowe, na które należy wysłać odpowiedzi.
- **MsgType:** Typ komunikatu.

## Dokumentacja

- Właściwości — 9 właściwości dostępnych dla komponentu.
- Metody — 16 metod publicznych.
- Zdarzenia — 6 zdarzeń do obsługi.

# Protokół AppRTC

WebRTC (Web Real-Time Communication) to definicja API opracowywana przez World Wide Web Consortium (W3C) w celu umożliwienia komunikacji przeglądarka-przeglądarka dla połączeń głosowych, czatów wideo i udostępniania plików P2P bez wtyczek. Skrót RTC w WebRTC oznacza Real-Time Communications, technologię umożliwiającą strumieniowanie audio/wideo i udostępnianie danych między klientami przeglądarki (partnerami). Jako zestaw standardów, WebRTC zapewnia każdej przeglądarce możliwość udostępniania danych aplikacji i przeprowadzania telekonferencji peer-to-peer, bez konieczności instalowania wtyczek lub oprogramowania innych firm.

Dostęp do komponentów WebRTC odbywa się za pośrednictwem interfejsów JavaScript API. Aktualnie w fazie rozwoju są: Network Stream API reprezentujący strumień danych audio lub wideo, oraz PeerConnection API umożliwiające komunikację między dwoma lub więcej użytkownikami przeglądarka-przeglądarka. Opracowywany jest również interfejs DataChannel API, który umożliwia komunikację innych typów danych na potrzeby gier czasu rzeczywistego, czatu tekstowego, transferu plików i podobnych zastosowań.

[appr.tc](#) to demonstracyjna aplikacja WebRTC opracowana przez Google i Mozilla, umożliwiająca obu przeglądarkom komunikację ze sobą przy użyciu API WebRTC.

## Komponenty

[TsgcWSPServer\\_AppRTC](#): Komponent VCL protokołu serwera AppRTC.

# TsgcWSPServer\_AppRTC

Jest to komponent Protocol AppRTC po stronie serwera.

## Wprowadzenie

Jest to komponent AppRTC protokołu serwera. Należy upuścić ten komponent na formularzu i wybrać komponent [TsgcWebSocketServer](#) przy użyciu właściwości Server.

## Parametry

- **IceServers:** tutaj można skonfigurować serwery turn/stun dla połączeń WebRTC.
- **RoomLink:** Bazowy adres URL umożliwiający dostęp do pokoju. Przykład: <https://mydemo.com/r/>
- **WebSocketURL:** adres URL serwera WebSocket. Przykład: <wss://mydemo.com>

Protokół WebRTC wymaga serwera STUN/TURN; w demach używane są publiczne serwery STUN/TURN do celów testowych. W systemie produkcyjnym wymagany jest dedykowany serwer STUN/TURN.

**Zarejestrowani użytkownicy** mogą pobierać skompilowane pliki binarne **serwera Coturn dla Windows**. Więcej informacji na temat [COTURN STUN/TURN](#).

## Konfiguracja IceServers

Jeśli serwer STUN/TURN działa pod następującym adresem IP: 51.122.4.88 i nasłuchuje na porcie 3478, a użytkownik do połączenia to "apprtc" z hasłem "secret", należy skonfigurować IceServers w następujący sposób:

```
{
  "lifetimeDuration": "86400s",
  "iceServers": [{
    "urls": "stun:51.122.4.88:3478",
    "username": "apprtc",
    "credential": "secret"
  }, {
    "urls": "turn:51.122.4.88:3478",
    "username": "apprtc",
    "credential": "secret"
  }],
  "blockStatus": "NOT_BLOCKED",
  "iceTransportPolicy": "all"
}
```

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 0 metod publicznych.
- Events — 11 zdarzeń do obsłużenia.

# Protokół WebRTC

WebRTC (Web Real-Time Communication) to definicja API opracowywana przez World Wide Web Consortium (W3C) w celu umożliwienia komunikacji przeglądarka-przeglądarka dla połączeń głosowych, czatów wideo i udostępniania plików P2P bez wtyczek. Skrót RTC w WebRTC oznacza Real-Time Communications, technologię umożliwiającą strumieniowanie audio/wideo i udostępnianie danych między klientami przeglądarki (partnerami). Jako zestaw standardów, WebRTC zapewnia każdej przeglądarce możliwość udostępniania danych aplikacji i przeprowadzania telekonferencji peer-to-peer, bez konieczności instalowania wtyczek lub oprogramowania innych firm.

Dostęp do komponentów WebRTC odbywa się za pośrednictwem interfejsów JavaScript API. Aktualnie w fazie rozwoju są: Network Stream API reprezentujący strumień danych audio lub wideo, oraz PeerConnection API umożliwiające komunikację między dwoma lub więcej użytkownikami przeglądarka-przeglądarka. Opracowywany jest również interfejs DataChannel API, który umożliwia komunikację innych typów danych na potrzeby gier czasu rzeczywistego, czatu tekstowego, transferu plików i podobnych zastosowań.

## Komponenty

[TsgcWSPServer\\_WebRTC](#): Komponent VCL protokołu serwera WebRTC.

## Parametry

- **IceServers:** tutaj można skonfigurować serwery TURN/STUN dla połączeń WebRTC. Domyślnie używane są następujące publiczne serwery STUN

```
{"iceServers": [{"url": "stun:stun.l.google.com:19302"}]}
```

## Test w przeglądarce

Jeśli chcesz przetestować ten protokół w ulubionej przeglądarce internetowej, wpisz ten adres URL (na-leży zdefiniować własny host i port)

`http://host:port/webrtc.esegece.com.html`

# TsgcWSPServer\_WebRTC

Jest to serwerowy komponent protokołu WebRTC.

## Wprowadzenie

Jest to składnik protokołu serwera WebRTC. Należy umieścić go w formularzu i wybrać składnik [TsgcWebSocketServer](#) za pomocą właściwości Server.

Protokół WebRTC wymaga serwera STUN/TURN; w demach używane są publiczne serwery STUN/TURN do celów testowych. W systemie produkcyjnym wymagany jest dedykowany serwer STUN/TURN.

**Zarejestrowani użytkownicy** mogą pobierać skompilowane pliki binarne **serwera Coturn dla Windows**. Więcej informacji na temat [COTURN STUN/TURN](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 2 publiczne metody.
- Zdarzenia — 11 zdarzeń do obsłużenia.



# Protocol WebRTC Javascript

Poniżej znajdują się dostępne metody. Należy zastąpić zmienne {%host%} i {%port%} odpowiednimi wartościami. Przykładowo: jeśli serwer sgcWebSocket jest skonfigurowany do nasłuchiwania na porcie 80 w witrynie www.example.com, należy skonfigurować:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
<script src="http://www.example.com:80/webrtc.esegece.com.js"></script>
```

## Otwórz połączenie

Gdy połączenie WebSocket zostaje otwarte, przeglądarka żąda dostępu do lokalnej kamery i mikrofonu; należy zezwolić na dostęp.

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/webrtc.esegece.com.js"></script>
<script>
    var socket = new sgcws_webrtc('ws://{%host%}:{%port%}');
</script>
```

## Otwórz kanał WebRTC

Gdy przeglądarka ma dostęp do lokalnej kamery i mikrofonu, wywoływane jest zdarzenie 'sgcmediastart', a następnie można podjąć próbę połączenia z innym klientem przy użyciu procedury webrtc\_connect

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/webrtc.esegece.com.js"></script>
<script>
    var socket = new sgcws_webrtc('ws://{%host%}:{%port%}');
    socket.on('sgcmediastart', function(event)
    {
        socket.webrtc_connect('custom channel');
    }
</script>
```

## Zamknij kanał WebRTC

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/webrtc.esegece.com.js"></script>
<script>
  socket.webrtc_disconnect('custom channel');
</script>
```

# Protokół WAMP

WAMP to otwarty protokół WebSocket oferujący dwa asynchroniczne wzorce przesyłania wiadomości: RPC i PubSub.

Technicznie WAMP jest oficjalnie zarejestrowanym subprotokołem WebSocket (działającym na wierzchu WebSocket), który używa JSON jako formatu serializacji wiadomości.

## Czym jest RPC?

Remote Procedure Call (RPC) to wzorzec przesyłania wiadomości obejmujący uczestników pełniących dwie role: klient i serwer.

Serwer udostępnia metody lub procedury do wywołania pod dobrze znanymi punktami końcowymi.

Klient wywołuje zdalne metody lub procedury, podając punkt końcowy metody lub procedury oraz wszelkie argumenty wywołania.

Serwer wykona metodę lub procedurę z użyciem dostarczonych argumentów wywołania i zwróci wynik do klienta.

## Co to jest PubSub?

Publish & Subscribe (PubSub) to wzorzec przesyłania wiadomości obejmujący peerów w trzech rolach: wydawcy, subskrybenta i brokera.

Wydawca wysyła (publikuje) zdarzenie, podając temat (znany też jako kanał) jako abstrakcyjny adres, nie konkretnego odbiorcę.

Subskrybent odbiera zdarzenia, podając najpierw tematy (czyli kanały), którymi jest zainteresowany. Następnie subskrybent będzie otrzymywał wszelkie zdarzenia opublikowane w danym temacie.

Broker działa między wydawcami a subskrybentami i pośredniczy w dostarczaniu wiadomości subskrybentom. Broker utrzymuje listy subskrybentów dla każdego tematu, dzięki czemu może przekazywać nowo opublikowane zdarzenia do odpowiednich subskrybentów.

Broker może również samodzielnie wysyłać zdarzenia, na przykład gdy broker pełni również rolę serwera RPC, a metoda wykonana na serwerze powinna wyzwolić zdarzenie PubSub.

Podsumowując, PubSub odspiera wydawców od odbiorców za pośrednictwem pośrednika, jakim jest broker.

## Komponenty

[TsgcWSPServer\\_WAMP](#): Składnik VCL protokołu WAMP po stronie serwera.

[TsgcWSPClient\\_WAMP](#): Komponent VCL protokołu klienta WAMP.

[Komponent Javascript](#): Dokumentacja klienta Javascript.

## Najczęstsze zastosowania

- **RPC**
  - [Proste RPC](#)
  - [Wyniki postępu RPC](#)
- **PubSub**
  - [Subskrybenci](#)
  - [Wydawcy](#)

## Test w przeglądarce

Aby przetestować ten protokół przy użyciu ulubionej przeglądarki internetowej, należy wpisać następujący adres URL (wymagane jest zdefiniowanie niestandardowego hosta i portu)

`http://host:port/wamp.esegece.com.html`

# TsgcWSPServer\_WAMP

To jest komponent protokołu serwera WAMP.

## Wprowadzenie

Jest to komponent protokołu serwera WAMP. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebsocketServer](#) za pomocą właściwości Server.

## Dokumentacja

- Właściwości — 4 właściwości dostępne w komponencie.
- Metody — 4 metody publiczne.
- Zdarzenia — 14 obsługiwanych zdarzeń.

# TsgcWSPClient\_WAMP

Jest to komponent protokołu klienta WAMP.

## Wprowadzenie

Jest to komponent klienta protokołu WAMP. Należy umieścić ten komponent na formularzu i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w składniku.
- Metody — 7 metod publicznych.
- Zdarzenia — 13 zdarzeń do obsługi.

# Protokół WAMP Javascript

Poniżej znajdują się dostępne metody. Należy zastąpić zmienne {%host%} i {%port%} odpowiednimi wartościami. Przykładowo: jeśli serwer sgcWebSocket jest skonfigurowany do nasłuchiwania na porcie 80 w witrynie www.example.com, należy skonfigurować:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
<script src="http://www.example.com:80/wamp.esegece.com.js"></script>
```

## Otwórz połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
</script>
```

## Wyślij nowy prefiks

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.prefix('sgc', 'http://www.esegece.com');
</script>
```

## Żądanie RPC (zdalne wywołanie procedury)

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.call('', 'sgc:CallTest', '20')
</script>
```

## Subskrybuj TopicURI

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.subscribe('sgc:test')
</script>
```

## Wypisanie z TopicURI

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.unsubscribe('sgc:test')
</script>
```

## Publikuj wiadomość

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.publish('sgc:channel', 'Test Message', [], []);
</script>
```

## Wyświetl alert z odebraną wiadomością

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.on('sgcmessage', function(event)
    {
        alert(event.message);
    }
</script>
```

## Pokaż alert przy OnCallResult lub OnCallError

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
```



```

var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
socket.on('wampcallresult', function(event)
{
    alert('call result: ' + event.CallId + ' - ' + event.CallResult);
}
socket.on('wampcallprogressresult', function(event)
{
    alert('call progress result: ' + event.CallId + ' - ' + event.CallResult);
}
socket.on('wampcallerror', function(event)
{
    alert('call error: ' + event.CallId + ' - ' + event.ErrorURI + ' - ' + event.ErrorDesc +
        ' - ' + event.ErrorDetails);
}
</script>

```

## Pokaż alert przy zdarzeniu

```

<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.on('wampevent', function(event)
    {
        alert('call result: ' + event.TopicURI + ' - ' + event.Event);
    }
</script>

```

## Wyświetl alert dla zdarzeń OnConnect, OnDisconnect i OnError

```

<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    var socket = new sgcws_wamp('ws://{%host%}:{%port%}');
    socket.on('open', function(event)
    {
        alert('sgcWebSocket Open!');
    }
    );
    socket.on('close', function(event)
    {
        alert('sgcWebSocket Closed!');
    }
    );
    socket.on('error', function(event)
    {
        alert('sgcWebSocket Error: ' + event.message);
    }
    );
</script>

```

## Zamknij połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    socket.close();
</script>
```

## Pobierz status połączenia

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/wamp.esegece.com.js"></script>
<script>
    socket.state();
</script>
```

# WAMP | Subskrybenci

Subskrybent odbiera zdarzenia, podając najpierw tematy (czyli kanały), którymi jest zainteresowany. Następnie subskrybent będzie otrzymywał wszelkie zdarzenia opublikowane w danym temacie.

Aby odbierać zdarzenia z tematu, subskrybent musi najpierw zasubskrybować ten temat.

## Klient WAMP

```
void OnMessageEvent(TsgcWSConnection *Connection, string Text)
{
    ShowMessage(Text);
}

oClient = new TsgcWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClientWAMP = new TsgcWSPClient_WAMP();
oClientWAMP->Client = oClient;
oClientWAMP->OnMessage = OnMessageEvent;
oClient->Active = true;

// Subscribe to topic after successful connect
oClient->Subscribe("myTopic");
```

## Serwer WAMP

```
void OnSubscriptionEvent(TsgcWSConnection *Connection, string Subscription)
{
    ShowMessage("Subscribed: " + Subscription);
}

oServer = new TsgcWebSocketServer();
oServer->Port = 80;
oServerWAMP = new TsgcWSPServer_WAMP();
oServerWAMP->OnSubscription = OnSubscriptionEvent;
oServerWAMP->Server = oServer;
oServerWAMP->Active = true;
```

## WAMP | Wydawcy

Wydawca wysyła (publikuje) zdarzenie, podając temat (czyli kanał) jako abstrakcyjny adres, a nie konkretny element równorzędny. Wystarczy wywołać metodę Publish i przekazać jako argumenty nazwę tematu i wiadomość do wysłania. Wiadomość ta zostanie dostarczona do wszystkich subskrybentów tego tematu. Warto zaznaczyć, że subskrypcja tematu nie jest wymagana do publikowania wiadomości w tym temacie.

Nie ma potrzeby konfigurowania czegokolwiek po stronie serwera, ponieważ wiadomości są automatycznie rozgłaszane do klientów po odebraniu wiadomości publikowania.

### Klient WAMP

```
oClient = new TsgcWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClientWAMP = new TsgcWSPClient_WAMP();
oClientWAMP->Client = oClient.
oClientWAMP->OnMessage = OnMessageEvent;
oClient->Active = true;

// Publish a message to all subscribers
oClient->Publish("myTopic", "Hello subscribers myTopic");
```

# WAMP | Proste RPC

Najczęstszym zastosowaniem komponentu WAMP jest żądanie przez klienta metody od serwera, a następnie wysłanie przez serwer odpowiedzi do klienta. Klient może przekazać wyłącznie nazwę metody lub dodatkowo parametry wymagane przez serwer do obliczenia wyniku. Serwer przetwarza żądania i w przypadku sukcesu wysyła do klienta odpowiedź z wynikiem. W przypadku błędu serwer wysyła do klienta odpowiedź z błędem.

Jak widać, istnieje tylko jedno żądanie i jedna odpowiedź (pomyślna lub nie).

**Example:** serwer posiada metodę o nazwie **GetTime**, więc za każdym razem, gdy klient wywoła tę metodę, serwer zwraca bieżący czas serwera.

## Serwer WAMP

```
void OnServerCall(TsgcWSConnection *Connection, const string CallId, const string ProcUri, const string Arguments)
{
    if (ProcUri == "GetTime")
    {
        oServerWAMP->CallResult(CallId, FormatDateTime("yyyyymmdd hh:nn:ss", Now));
    }
    else
    {
        oServer->WAMP->CallError(CallId, "Unknown method");
    }
}

oServer = new TsgcWebSocketServer();
oServer->Port = 80;
oServerWAMP = new TsgcWSPServer_WAMP();
oServerWAMP->OnCall = OnServerCallEvent();
oServerWAMP->Server = oServer;
oServer->Active = true;
```

## Klient WAMP

```
void OnCallResultClient(TsgcWSConnection *Connection, string CallId, string Result);
{
    ShowMessage(Result);
}

void OnCallErrorClient(TsgcWSConnection *Connection, string Error)
{
    ShowMessage(Error);
}

oClient = new TsgcWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClientWAMP = new TsgcWSPClient_WAMP();
oClientWAMP->OnCallResult = OnCallResultClient;
oClientWAMP->OnCallError = OnCallErrorClient;
oClientWAMP->Client = oClient;
oClient->Active = true;
```

```
// After client has connected, request GetTime from server  
oClientWAMP->Call("GetTime");
```

# WAMP | Progresywne wyniki RPC

Czasem zdalne wywołania procedur wymagają więcej niż jednego wyniku do zakończenia żądań. Domyślnie protokół WAMP 1.0 nie obsługuje wyników częściowych w wywołaniu — jest to funkcja dostępna wyłącznie w bibliotece `sgcWebSockets`.

Przepływ jest bardzo podobny do prostego wywołania RPC, jednak tutaj przed finalnym wywołaniem `CallResult` pojawia się jeden lub więcej wyników częściowych.

Ogólnie rzecz biorąc, klient wysyła żądanie procedury do serwera, a serwer może odesłać wynik lub błąd. Jeśli odsyła wynik, może to być wynik końcowy lub musi odesłać więcej wyników później. Jeśli jest to wynik końcowy, zostanie wywołana metoda **CallResult** i proces dobiegnie końca. Jeśli jest więcej wyników do wysłania, zostanie wywołana metoda **CallProgressResult**.

**Przykład:** klient żąda od serwera metody, która będzie zwracać czas serwera co sekundę i zatrzyma się po 20 wiadomościach.

## Serwer WAMP

```
void OnServerCall(TsgcWSConnection *Connection, const string CallId, const string ProcUri, const string Arguments)
{
    if (ProcUri == "GetProgressiveTime")
    {
        int vNum = StrToInt(Arguments);
        for (int i = 1; i = vNum; i++)
        {
            if (i == 20)
            {
                oServerWAMP->CallResult(CallId, FormatDateTime("yyyymmdd hh:nn:ss", Now));
            }
            else
            {
                oServerWAMP->CallProgressiveResult(CallId, FormatDateTime("yyyymmdd hh:nn:ss", Now));
            }
        }
    }
    else
    {
        oServer->WAMP->CallError(CallId, "Unknown method");
    }
}

oServer = new TsgcWebSocketServer();
oServer->Port = 80;
oServerWAMP = new TsgcWSPServer_WAMP();
oServerWAMP->OnCall = OnServerCallEvent();
oServerWAMP->Server = oServer;
oServer->Active = true;
```

## Klient WAMP

```

void OnCallResultClient(TsgcWSConnection *Connection, string CallId, string Result);
{
    ShowMessage(Result);
}
void OnCallProgressResultClient(TsgcWSConnection *Connection, string CallId, string Result);
{
    ShowMessage(Result);
}
void OnCallErrorClient(TsgcWSConnection *Connection, string Error)
{
    ShowMessage(Error);
}
oClient = new TsgcWebSocketClient();
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClientWAMP = new TsgcWSPClient_WAMP();
oClientWAMP->OnCallResult = OnCallResultClient;
oClientWAMP->OnCallProgressResult = OnCallProgressResultClient;
oClientWAMP->OnCallError = OnCallErrorClient;
oClientWAMP->Client = oClient;
oClient->Active = true;
// After client has connected, request GetTime from server
oClientWAMP->Call("GetProgressTime");

```



# Protokół WAMP 2

WAMP zapewnia ujednolicone trasowanie aplikacji w otwartym protokole WebSocket, który współpracuje z różnymi językami programowania.

Używając WAMP, można budować systemy rozproszone z komponentów aplikacji, które są luźno powiązane i komunikują się w (zbliżonym do) czasie rzeczywistym.

Zasadniczo WAMP oferuje dwa wzorce komunikacji dla komponentów aplikacji:

- Publikuj i subskrybuj (PubSub)
- Zdalne wywoływanie procedur (RPC)

WAMP jest łatwy w użyciu, prosty we wdrożeniu i oparty na nowoczesnych standardach internetowych: WebSocket, JSON i URI.

## Komponenty

[TsgcWSPClient\\_WAMP2](#): Komponent VCL protokołu klienta WAMP2.

# TsgcWSPClient\_WAMP2

Jest to komponent protokołu klienta WAMP2.

## Wprowadzenie

To jest komponent protokołu WAMP2 dla klienta. Należy go upuścić na formularz i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

### Metody sesji

- **ABORT:** Zarówno router, jak i klient mogą przerwać otwieranie sesji WAMP, wysyłając wiadomość ABORT.

**Reason** MUSI być URI.

**Details** MUSI być słownikiem umożliwiającym podanie dodatkowych, opcjonalnych informacji o zamknięciu (patrz poniżej).

Na wiadomość ABORT nie oczekuje się żadnej odpowiedzi.

- **GOODBYE:** Sesja WAMP rozpoczyna swój czas życia wraz z wysłaniem przez router wiadomości WELCOME do klienta i kończy się, gdy bazowy transport zostaje zerwany lub gdy sesja WAMP jest jawnie zamykana komunikatem GOODBYE wysłanym przez jeden z węzłów i odpowiadającym komunikatem GOODBYE wysłanym przez drugi węzeł.

**Reason** MUSI być URI.

**Details** MUSI być słownikiem umożliwiającym dostarczenie dodatkowych, opcjonalnych informacji dotyczących zamknięcia.

### Metody publikowania/subskrybowania

- **PUBLISH:** Gdy wydawca chce opublikować zdarzenie w danym temacie, wysyła wiadomość PUBLISH do brokera:

**Request** to losowy, efemeryczny identyfikator wybrany przez wydawcę, służący do korelacji odpowiedzi brokera z żądaniem.

**Options** to słownik umożliwiający podanie dodatkowych szczegółów żądania publikacji w rozszerzalny sposób. Zostało to opisane szczegółowo poniżej.

**Topic** to temat, do którego publikowana jest wiadomość.

**Arguments** to lista elementów ładunku zdarzenia na poziomie aplikacji. Lista może mieć zerową długość.

**ArgumentsKw** to opcjonalny słownik zawierający ładunek zdarzenia na poziomie aplikacji, dostarczany jako argumenty nazwane. Słownik może być pusty.

Jeśli broker jest w stanie zrealizować i zezwolić na publikację, wyśle zdarzenie do wszystkich bieżących subskrybentów tematu opublikowanego zdarzenia.

Domyślnie publikacje są niepotwierdzone, a broker nie odpowie niezależnie od tego, czy publikacja zakończyła się sukcesem.

- **SUBSCRIBE:** Subskrybent informuje brokera o zainteresowaniu tematem, wysyłając wiadomość SUBSCRIBE:

**Żądanie** MUSI być losowym, efemerycznym identyfikatorem wybranym przez subskrybenta i używanym do korelowania odpowiedzi brokera z żądaniem.

**Options** MUSI być słownikiem umożliwiającym podanie dodatkowych szczegółów żądania subskrypcji w rozszerzalny sposób.

**Topic** to temat, który subskrybent chce subskrybować, i MUSI być identyfikatorem URI.

- **UNSUBSCRIBE:** Gdy subskrybent nie jest już zainteresowany odbieraniem zdarzeń dla danej subskrypcji, wysyła wiadomość UNSUBSCRIBE

**Żądanie** MUSI być losowym, efemerycznym identyfikatorem wybranym przez subskrybenta i używanym do korelowania odpowiedzi brokera z żądaniem.

**SUBSCRIBED.Subscription** MUSI być identyfikatorem subskrypcji, z której należy zrezygnować, pierwotnie przydzielonym przez brokera subskrybentowi.

## Metody RPC

- **CALL:** Gdy Caller chce wywołać procedurę zdalną, wysyła do Dealera wiadomość CALL:

**Request** to losowy, efemeryczny identyfikator wybrany przez wywołującego, służący do korelowania odpowiedzi dealera z żądaniem.

**Options** to słownik umożliwiający podanie dodatkowych szczegółów żądania wywołania w rozszerzalny sposób. Zostało to szczegółowo opisane poniżej.

**Procedure** to URI wywoływanej procedury.

**Arguments** to lista pozycyjnych argumentów wywołania (każdy dowolnego typu). Lista może mieć zerową długość.

**ArgumentsKw** to słownik słownikowych argumentów wywołania (każdy dowolnego typu). Słownik może być pusty.

- **REGISTERCALL**: wywoływany ogłasza dostępność punktu końcowego implementującego procedurę u Dealera, wysyłając wiadomość REGISTER:

**Request** to losowy, efemeryczny identyfikator wybierany przez Callee i służący do korelowania odpowiedzi Dealera z żądaniem.

**Options** to słownik umożliwiający podanie dodatkowych szczegółów żądania rejestracji w rozszerzalny sposób. Zostało to opisane szerzej poniżej.

**Procedure** to procedura, którą Callee chce zarejestrować

- **UNREGISTERCALL**: gdy Callee nie chce już udostępniać implementacji zarejestrowanej procedury, wysyła komunikat UNREGISTER do Dealera:

**Request** to losowy, efemeryczny identyfikator wybierany przez Callee i służący do korelowania odpowiedzi Dealera z żądaniem.

**REGISTERED.Registration** to identyfikator rejestracji do unieważnienia, pierwotnie przyznany przez Dealera Callee.

- **INVOCATION**: Jeśli Dealer jest w stanie zrealizować (zapośredniczyć) wywołanie i je dopuszcza, wysyła wiadomość INVOCATION do odpowiedniego Callee implementującego procedurę:

**Request** to losowy, efemeryczny identyfikator wybierany przez Dealera służący do powiązania odpowiedzi Callee z żądaniem.

**REGISTERED.Registration** to identyfikator rejestracji, pod którym procedura została zarejestrowana u Dealera.

**Details** to słownik umożliwiający podanie dodatkowych szczegółów żądania wywołania w rozszerzalny sposób. Zostało to szczegółowo opisane poniżej.

**CALL.Arguments** to oryginalna lista pozycyjnych argumentów wywołania przekazana przez wywołującego.

**CALL.ArgumentsKw** to oryginalny słownik argumentów wywołania podanych przez wywołującego (Caller).

- **YIELD**: Jeśli wywoływany jest w stanie pomyślnie przetworzyć i zakończyć wykonanie wywołania, odpowiada wysyłając wiadomość YIELD do dealera:

**INVOCATION.Request** to identyfikator z oryginalnego żądania wywołania.

**Options** to słownik umożliwiający podanie dodatkowych opcji.

**Arguments** to lista pozycyjnych elementów wyniku (każdy dowolnego typu). Lista może mieć zerową długość.

**ArgumentsKw** to słownik elementów wyników ze słowami kluczowymi (każdy o dowolnym typie). Słownik może być pusty.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 11 metod publicznych.
- Zdarzenia — 20 zdarzeń do obsługi.

# Protokół domyślny

Jest to domyślny podprotokół zaimplementowany przy użyciu komunikatów „JSONRPC 2.0”. Za każdym razem, gdy wysyłana jest wiadomość przy użyciu tego protokołu, tworzony jest obiekt JSON o następujących właściwościach:

**jsonrpc:** Ciąg znaków określający wersję protokołu JSON-RPC. MUSI mieć dokładnie wartość "2.0".

**method:** Ciąg zawierający nazwę metody, która ma zostać wywołana. Nazwy metod zaczynające się od słowa rpc, po którym następuje znak kropki (U+002E lub ASCII 46), są zarezerwowane dla metod wewnętrznych rpc i rozszerzeń i NIE MOGĄ być używane do żadnych innych celów.

**params:** Wartość strukturalna zawierająca wartości parametrów używanych podczas wywoływania metody. Ten element MOŻE być pominięty.

**id:** identyfikator ustanowiony przez klienta, który MUSI zawierać wartość String, Number lub NULL (jeśli jest podany). Jeśli nie jest podany, zakłada się, że jest to powiadomienie. Wartość POWINNA normalnie nie być Null [1], a wartości liczbowe NIE POWINNY zawierać części ułamkowych [2].

## Przykład obiektu JSON:

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

## Funkcje

- **Publish/subscribe** — wzorzec wiadomości umożliwiający dystrybucję wiadomości jeden-do-wielu i oddzielenie aplikacji. Obsługuje symbole wieloznaczne, dzięki czemu można subskrybować hierarchię kanałów. Przykład: aby subskrybować wszystkie kanały zaczynające się od 'news', należy wywołać `Subscribe('news*')`.
- Transport wiadomości **niezależny** od zawartości ładunku
- **Potwierdzenie** wysłanych wiadomości.
- Obsługuje **wiadomości transakcyjne** poprzez lokalne transakcje serwera. Gdy klient zatwierdza transakcję, serwer przetwarza wszystkie wiadomości w kolejce. Jeśli klient wycofa transakcję, wszystkie wiadomości zostaną usunięte.
- Implementuje **QoS** (Quality of Service) na potrzeby dostarczania wiadomości.

## **Komponenty**

[TsgcWSPClient\\_sgc](#): Komponent VCL domyślnego protokołu serwera.

[TsgcWSPClient\\_sgc](#): Domyślny komponent VCL protokołu klienta.

[Komponent Javascript](#): Dokumentacja klienta Javascript.

## **Test w przeglądarce**

Aby przetestować ten protokół przy użyciu ulubionej przeglądarki internetowej, należy wpisać następujący adres URL (wymagane jest zdefiniowanie niestandardowego hosta i portu)

`http://host:port/esegece.com.html`

# TsgcWSPServer\_sgc

Jest to domyślny komponent protokołu serwera.

## Wprowadzenie

Jest to domyślny komponent protokołu serwera. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketServer](#) za pomocą właściwości Server.

## Dokumentacja

- Właściwości — 6 właściwości dostępnych na komponencie.
- Metody — 6 metod publicznych.
- Zdarzenia — 16 zdarzeń do obsługi.



# TsgcWSPClient\_sgc

Jest to domyślny komponent protokołu klienta.

## Wprowadzenie

Jest to domyślny komponent protokołu klienta. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 11 metod publicznych.
- Zdarzenia — 15 zdarzeń do obsługi.

# TsgcIWWSPClient\_sgc

Jest to domyślny komponent protokołu klienta IntraWeb.

## Wprowadzenie

Jest to domyślny komponent protokołu klienta IntraWeb. Należy umieścić ten komponent w formularzu i wybrać komponent [TsgcIWWebSocketClient](#) przy użyciu właściwości Client.

## Dokumentacja

- Właściwości — 2 właściwości dostępne dla komponentu.
- Metody — 3 metody publiczne.
- Zdarzenia — 6 zdarzeń do obsługi.

# Domyślny protokół JavaScript

Domyślny protokół JavaScript: sgcWebSockets używa plików **sgcWebSocket.js** i **esegece.com.js**.

Poniżej znajdują się dostępne metody. Należy zastąpić zmienne {%host%} i {%port%} odpowiednimi wartościami. Przykładowo: jeśli serwer sgcWebSocket jest skonfigurowany do nasłuchiwania na porcie 80 w witrynie [www.example.com](http://www.example.com), należy skonfigurować:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
<script src="http://www.example.com:80/esegece.com.js"></script>
```

## Otwórz połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
</script>
```

## Wyślij wiadomość

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
  socket.send('Hello sgcWebSockets!');
</script>
```

## Wyświetl alert z odebraną wiadomością

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
  socket.on('sgcmessage', function(event)
  {
    alert(event.message);
  }
</script>
```

## Publikuj wiadomość na kanale testowym

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.publish('Hello sgcWebSockets!', 'test');
</script>
```

## Wyświetl alert z odebraną wiadomością zdarzenia

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.on('sgcevent', function(event)
    {
        alert('channel:' + event.channel + '. message: ' + event.message);
    }
</script>
```

## Wywołaj RPC

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    var params = {param:10};
    socket.rpc(GUID(), 'test', JSON.stringify(params));
</script>
```

## Obsługa odpowiedzi RPC

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.on('sgcrpcresult', function(event)
    {
        alert('result:' + event.result);
    }
    socket.on('sgcrpcerror', function(event)
    {
        alert('error:' + event.code + ' ' + event.message);
    }
</script>
```

## Wywołaj Notify

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
  var params = {param:10};
  socket.notify('test', JSON.stringify(params));
</script>
```

## Wysyłaj wiadomości w ramach transakcji

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');

  socket.starttransaction('sgc:test');
  socket.publish('Message1', 'sgc:test');
  socket.publish('Message2', 'sgc:test');
  socket.publish('Message3', 'sgc:test');
  socket.commit('sgc:test');
</script>
```

## Wyświetl alert przy subskrypcji lub rezygnacji z subskrypcji kanału

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
  socket.on('sgcsubscribe', function(event)
  {
    alert('subscribed: ' + event.channel);
  }
  socket.on('sgcunsubscribe', function(event)
  {
    alert('unsubscribed: ' + event.channel);
  }
</script>
```

## Wyświetl alert dla zdarzeń OnConnect, OnDisconnect i OnError

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
  var socket = new sgcws('ws://{%host%}:{%port%}');
  socket.on('open', function(event)
  {
    alert('sgcWebSocket Open!');
  }
  );
  socket.on('close', function(event)
  {
    alert('sgcWebSocket Closed!');
  }
  );
</script>
```

```
};
socket.on('error', function(event)
{
    alert('sgcWebSocket Error: ' + event.message);
});
</script>
```

## Pobierz sesję

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.on('sgcsession', function(event)
    {
        alert(event.guid);
    });
    socket.getsession();
</script>
```

## Zamknij połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    socket.close();
</script>
```

## Pobierz status połączenia

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');

    socket.state();
</script>
```

## Ustaw QoS

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');

    socket.qoslevel1();
```

```
    socket.publish('message', 'channel');  
</script>
```

## Ustaw poziom kolejki

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>  
<script src="http://{%host%}:{%port%}/esegece.com.js"></script>  
<script>  
    var socket = new sgcws('ws://{%host%}:{%port%}');  
  
    socket.queuelevel2();  
    socket.publish('message1', 'channel1');  
    socket.publish('message2', 'channel1');  
</script>
```

# Zestaw danych protokołu

Ten protokół dziedziczy po Protokole Domyślnym i jest przydatny, gdy zachodzi potrzeba rozgłaszania zmian zestawu danych do klientów podłączonych do tego protokołu. Może być używany w 2 trybach:

**1. Replikacja bazy danych:** zmiany w bazie danych są replikowane do wszystkich baz danych klientów. Przykład: serwer posiada bazę danych z notowaniami giełdowymi, a wszyscy podłączeni klienci otrzymują zmiany notowań. Istnieje jedna baza danych (na serwerze), a każdy klient ma własną bazę. Za każdym razem, gdy notowanie jest aktualizowane, zmiana ta jest rozsyłana do wszystkich podłączonych klientów, a każdy klient aktualizuje własny rekord w bazie. Należy używać UpdateMode: upWhereAll lub upWhereChanged dla tego typu trybu.

**2. Aktualizacje bazy danych:** istnieje jedna baza danych współdzielona przez serwer i klientów; za każdym razem gdy klient aktualizuje rekord w bazie danych, wszyscy pozostali klienci chcą otrzymać powiadomienie o tej aktualizacji. W tym trybie należy użyć UpdateMode: upRefreshAll.

## Najczęstsze zastosowania

- Tryb aktualizacji
  - [Jak replikować tabelę](#)
  - [Jak powiadamiać o aktualizacjach](#)

Używa obiektu "JSON-RPC 2.0", a przy każdej zmianie zbioru danych wysyła wszystkie wartości pól (\* obsługiwane są tylko pola) za pomocą obiektu Dataset.

Aby umożliwić komponentowi wyszukiwanie rekordów w zestawie danych, należy określić, które pola stanowią klucz. **Przykład:** jeśli w zestawie danych pole ID jest kluczem, należy napisać kod podobny do poniższego:

```
void OnAfterOpenDataSet(TDataSet *DataSet)
{
    DataSet->FieldByName("ID")->ProviderFlags =
        DataSet->FieldByName("ID")->ProviderFlags + [pfInKey];
}
```

## Komponenty

[TsgcWSPServer\\_Dataset](#): Komponent VCL protokołu Dataset po stronie serwera.



[TsgcWSPClient\\_Dataset](#): Komponent VCL protokołu klienta Dataset.

[Komponent JavaScript](#): Dokumentacja referencyjna JavaScript dla klienta.

## **Test w przeglądarce**

Aby przetestować ten protokół przy użyciu ulubionej przeglądarki internetowej, należy wpisać następujący adres URL (wymagane jest zdefiniowanie niestandardowego hosta i portu)

```
http://host:port/dataset.esegece.com.html
```

# TsgcWSPServer\_Dataset

To jest komponent protokołu serwera Dataset.

## Wprowadzenie

Jest to komponent protokołu Dataset po stronie serwera. Należy umieścić ten komponent na formularzu i wybrać komponent [TsgcWebsocketServer](#) przy użyciu właściwości Server oraz wybrać komponent Dataset przy użyciu właściwości Dataset.

Ten komponent dziedziczy z [TsgcWSProtocol\\_Server\\_sgc](#) wszystkie metody i właściwości.

## Dokumentacja

- Właściwości — 13 właściwości dostępnych w komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 22 zdarzenia do obsługi.

# TsgcWSPClient\_Dataset

To jest komponent zestawu danych protokołu klienta.

## Wprowadzenie

Jest to komponent protokołu Dataset po stronie klienta. Należy upuścić ten komponent na formularzu i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client oraz komponent Dataset przy użyciu właściwości Dataset.

Ten komponent dziedziczy z [TsgcWSProtocol\\_Client\\_sgc](#) wszystkie metody i właściwości.

## Dokumentacja

- Właściwości — 13 właściwości dostępnych w komponencie.
- Metody — 14 metod publicznych.
- Zdarzenia — 25 zdarzeń do obsługi.

# TsgcIWWSPClient\_Dataset

Jest to komponent zestawu danych protokołu klienta IntraWeb.

## Wprowadzenie

Jest to komponent klienta protokołu Dataset dla IntraWeb. Należy umieścić ten komponent na formularzu, wybrać komponent [TsgcIWWWebSocketClient](#) za pomocą właściwości Client oraz wybrać komponent Dataset za pomocą właściwości Dataset.

Ten komponent dziedziczy po [TsgcIWWSPClient\\_sgc](#) wszystkie metody i właściwości.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 6 metod publicznych.
- Zdarzenia — 9 zdarzeń do obsługi.

# Protocol Dataset Javascript

Protokół Dataset JavaScript sgcWebSockets używa plików **sgcWebSocket.js** i **dataset.esegece.com.js**.

Poniżej znajdują się dostępne metody. Należy zastąpić zmienne {%host%} i {%port%} odpowiednimi wartościami. Przykładowo: jeśli serwer sgcWebSocket jest skonfigurowany do nasłuchiwania na porcie 80 w witrynie www.example.com, należy skonfigurować:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
<script src="http://www.example.com:80/dataset.esegece.com.js"></script>
```

## Otwórz połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    var socket = new sgcws_dataset('ws://{%host%}:{%port%}');
</script>
```

## Wyślij wiadomość

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    var socket = new sgcws_dataset('ws://{%host%}:{%port%}');
    socket.send('Hello sgcWebSockets!');
</script>
```

## Wyświetl alert z odebraną wiadomością

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    var socket = new sgcws('ws://{%host%}:{%port%}');
    socket.on('sgcdataset', function(event)
    {
        alert(event.dataset);
    }
</script>
```

## Pokaż alert z odebrany datasetem

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
  var socket = new sgcws_dataset('ws://{%host%}:{%port%}');
  socket.on('sgcmessage', function(event)
  {
    alert(event.message);
  }
  </script>
```

## Wyświetl alert przy subskrypcji lub rezygnacji z subskrypcji kanału

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
  var socket = new sgcws_dataset('ws://{%host%}:{%port%}');
  socket.on('sgcsubscribe', function(event)
  {
    alert('subscribed: ' + event.channel);
  }
  socket.on('sgcunsubscribe', function(event)
  {
    alert('unsubscribed: ' + event.channel);
  }
  </script>
```

## Wyświetl alert dla zdarzeń OnConnect, OnDisconnect i OnError

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
  var socket = new sgcws_dataset('ws://{%host%}:{%port%}');
  socket.on('open', function(event)
  {
    alert('sgcWebSocket Open!');
  }
  );
  socket.on('close', function(event)
  {
    alert('sgcWebSocket Closed!');
  }
  );
  socket.on('error', function(event)
  {
    alert('sgcWebSocket Error: ' + event.message);
  }
  );
  </script>
```

## Subskrybuj wszystkie zmiany zbioru danych

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
```

```
    socket.subscribe_all();
</script>
```

## Anulowanie subskrypcji wszystkich zmian Dataset

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    socket.unsubscribe_all();
</script>
```

## Obsługa zmian w zbiorze danych

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
var socket = new sgcws_dataset('ws://{%host%}:{%port%}');

socket.on('sgcdataset', function(evt){

if ((evt.channel == "sgc@dataset@new") || (evt.channel == "sgc@dataset@update")) {

... here you need to implement your own code insert/update records ...
}
else if (evt.channel == "sgc@dataset@delete") {

... here you need to implement your own code to delete records ...

}
});
</script>
```

## Zamknij połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    socket.close();
</script>
```

## Pobierz status połączenia

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/dataset.esegece.com.js"></script>
<script>
    socket.state();
</script>
```

# Protokół Dataset | Replikacja tabeli

Ten tryb rozwiązuje typowy scenariusz, w którym **tabela jest replikowana** dla wszystkich połączonych klientów, **przykład**: jeśli serwer posiada **tabelę notowań giełdowych**, chcemy **rozgłaszać zmiany notowań** do wszystkich **klientów**, ale nie chcemy, aby klient mógł łączyć się bezpośrednio z bazą danych. W związku z tym przy każdej **zmianie** w notowaniach giełdowych **informacja o rekordzie jest rozgłaszana** do wszystkich połączonych klientów. Każdy **klient odczytuje rekord** i **aktualizuje** własną tabelę.

Przykłady można znaleźć w folderze Demos, w demonstracji SQLite/MultipleDatabase.

## Konfiguracja serwera DataSet

Utwórz nowy serwer protokołu Dataset i skonfiguruj go przy użyciu następujących właściwości

- **ApplyUpdates**: ustawić na **True**; za każdym razem, gdy nastąpi zmiana, zostanie ona rozesłana do klientów
- **AutoSynchronize**: ustawione na **True** powoduje, że za każdym razem, gdy nowy klient nawiązuje połączenie z serwerem, serwer wysyła wszystkie rekordy (metadane i dane), dzięki czemu klient otrzymuje najnowsze informacje z serwera.
- **UpdateMode**: ustaw na **upWhereAll** lub **upWhereChanged**. Różnica polega na tym, że pierwsza opcja wysyła wszystkie pola rekordu, a druga tylko pola zmienione podczas aktualizacji.

```
TsgcWebSocketServer *oServer = new TsgcWebSocketServer();
TsgcWSPServer_Dataset *oProtocolDataset = new TsgcWSPServer_Dataset();
oProtocolDataset->Server = oServer;
oProtocolDataset->Dataset = <...your dataset...>;
oProtocolDataset->ApplyUpdates = true;
oProtocolDataset->AutoSynchronize = true;
oProtocolDataset->NotifyUpdates = true;
oProtocolDataset->UpdateMode = upWhereAll;
oServer->Port = 80;
oServer->Active = true;
```

## Konfiguruj klienta Dataset

Utwórz nowy komponent klienta protokołu Dataset i skonfiguruj go przy użyciu następujących właściwości:

- **ApplyUpdates**: ustawienie na **True** powoduje, że każda zmiana będzie wysyłana do serwera.
- **AutoSubscribe**: gdy ustawione na **True**, przy każdym nowym połączeniu klienta z serwerem klient automatycznie subskrybuje aktualizacje, usunięcia i nowe rekordy.
- **UpdateMode**: ustaw na **upWhereAll** lub **upWhereChanged**. Różnica polega na tym, że pierwsza opcja wysyła wszystkie pola rekordu, a druga tylko pola zmienione podczas aktualizacji.



```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSPClient_Dataset *oProtocolDataset = new TsgcWSPClient_Dataset();
oProtocolDataset->Client = oClient;
oProtocolDataset->Dataset = <...your dataset...>;
oProtocolDataset->ApplyUpdates = true;
oProtocolDataset->AutoSubscribe = true;
oProtocolDataset->NotifyUpdates = true;
oProtocolDataset->UpdateMode = upWhereAll;
oClient->Host = "127.0.0.1";
oClient->Port = 80;
oClient->Active = true;
```

# Protokół Dataset | Powiadamianie o aktualizacjach

Ten tryb próbuje rozwiązać scenariusz, w którym **serwer** i **klienci** współdzielą **jedną bazę danych** (serwer i klienci są połączeni z tą samą fizyczną bazą danych), a klienci chcą być powiadamiani za każdym razem, gdy inny klient wprowadzi zmiany w zbiorze danych.

Przykład można znaleźć w folderze Demos, w wersji demonstracyjnej SQLite/SingleDatabase.

## Konfiguracja serwera DataSet

Utwórz nowy serwer protokołu Dataset i skonfiguruj go przy użyciu następujących właściwości

- **ApplyUpdates:** ustawić na **True**; za każdym razem, gdy nastąpi zmiana, zostanie ona rozesłana do klientów
- **AutoSynchronize:** ustaw na **False**, tutaj nie jest potrzebne ustawianie na True, ponieważ klient jest połączony z tą samą bazą danych co serwer.
- **UpdateMode:** ustawić na **upRefreshAll**.

```
TsgcWebSocketServer *oServer = new TsgcWebSocketServer();
TsgcWSPServer_Dataset *oProtocolDataset = new TsgcWSPServer_Dataset();
oProtocolDataset->Server = oServer;
oProtocolDataset->Dataset = <...your dataset...>;
oProtocolDataset->ApplyUpdates = true;
oProtocolDataset->AutoSynchronize = false;
oProtocolDataset->NotifyUpdates = true;
oProtocolDataset->UpdateMode = upRefreshAll;
oServer->Port = 80;
oServer->Active = true;
```

## Konfiguruj klienta Dataset

Utwórz nowy komponent klienta protokołu Dataset i skonfiguruj go przy użyciu następujących właściwości:

- **ApplyUpdates:** ustawienie na **True** powoduje, że każda zmiana będzie wysyłana do serwera.
- **AutoSubscribe:** gdy ustawione na **True**, przy każdym nowym połączeniu klienta z serwerem klient automatycznie subskrybuje aktualizacje, usunięcia i nowe rekordy.
- **UpdateMode:** należy ustawić na **upRefreshAll**.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSPClient_Dataset *oProtocolDataset = new TsgcWSPClient_Dataset();
oProtocolDataset->Client = oClient;
oProtocolDataset->Dataset = <...your dataset...>;
oProtocolDataset->ApplyUpdates = true;
oProtocolDataset->AutoSubscribe = true;
oProtocolDataset->NotifyUpdates = true;
```

```
oProtocolDataset->UpdateMode = upRefreshAll;  
oClient->Host = "127.0.0.1";  
oClient->Port = 80;  
oClient->Active = true;
```

# Protokół Files

Protokół ten umożliwia wysyłanie plików przy użyciu binarnego transportu WebSocket. Obsługuje duże pliki przy niskim zużyciu pamięci.

## Funkcje

- **Wzorzec publikowania/subskrypcji** zapewniający dystrybucję wiadomości jeden-do-wielu i rozdzielenie aplikacji.
- **Potwierdzenie** wysłanych wiadomości.
- Implementuje **QoS** (Quality of Service) dla dostarczania plików.
- Opcjonalnie można zażądać **Autoryzacji** dla odebranych plików.
- **Niskie zużycie** pamięci.

## Komponenty

[TsgcWSPServer\\_Files](#): Komponent VCL protokołu plików serwera.

[TsgcWSPClient\\_Files](#): Komponent VCL protokołu plików klienta.

## Klasy

[TsgcWSMessageFile](#): obiekt hermetyzujący informacje o pakiecie pliku.

## Najczęstsze zastosowania

- **Wyślij pliki**
  - [Jak wysyłać pliki do serwera](#)
  - [Jak wysyłać pliki do klientów](#)
- **Duże pliki**
  - [Jak wysyłać duże pliki](#)

# TsgcWSPServer\_Files

Jest to komponent protokołu plików po stronie serwera.

## Wprowadzenie

Jest to składnik protokołu serwera plików. Należy umieścić go w formularzu i wybrać składnik [TsgcWebSocketServer](#) za pomocą właściwości Server.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 3 metody publiczne.
- Zdarzenia — 20 zdarzeń do obsługi.

# TsgcWSPClient\_Files

Jest to komponent klienta protokołu plików.

## Wprowadzenie

Jest to komponent protokołu klienta dla plików. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketClient](#) za pomocą właściwości Client.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 4 publiczne metody.
- Zdarzenia — 19 zdarzeń do obsługi.

# TsgcWSMessageFile

Ten obiekt jest przekazywany jako parametr za każdym razem, gdy wywoływane jest zdarzenie protokołu plików.

## Właściwości

- **BufferSize:** domyślny rozmiar pakietu.
- **Channel:** jeśli zostanie podany, plik zostanie wysłany wyłącznie do klientów subskrybujących określony kanał.
- **Metoda:** metoda wewnętrzna.
- **FileId:** identyfikator pliku; jest unikalny dla wszystkich odebranych/wysłanych plików.
- **Data:** niestandardowe dane użytkownika. Użytkownik może tu ustawić dowolny tekst.
- **FileName:** nazwa pliku.
- **FilePosition:** pozycja w pliku w bajtach.
- **FileSize:** łączny rozmiar pliku w bajtach.
- **Id:** identyfikator pakietu; jest unikalny dla każdego pakietu.
- **QoS:** jakość usługi komunikatu.
- **Streaming:** do użytku wewnętrznego.
- **Text:** do użytku wewnętrznego.

# Pliki protokołu | Jak wysyłać pliki do serwera

Aby wysłać plik do serwera, wystarczy wywołać metodę **SendFile** protokołu Files i przekazać pełną ścieżkę **FileName** jako argument.

Plik odebrany przez serwer zostanie domyślnie zapisany w tym samym katalogu, w którym znajduje się plik wykonywalny serwera, lub w ścieżce ustawionej we właściwości Files.SaveDirectory.

```
// ... Create Server
TsgcWebSocketServer oServer = new TsgcWebSocketServer();
TsgcWSPServer_Files oServer_Files = new TsgcWSPServer_Files();
oServer_Files->Server = oServer;
oServer->Host = "127.0.0.1";
oServer->Port = 8080;

// ... Create Client
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
oClient->URL = "ws://127.0.0.1:8080";

// ... Create Protocol
TsgcWSPClient_Files oClient_Files = new TsgcWSPClient_Files();
oClient_Files->Client = oClient;

// ... Start Server
oServer->Active = true;

// ... Connect client and Send File
if oClient->Connect() then
    oClient_Files->SendFile("c:\Documents\yourfile.txt");
```



# Pliki protokołu | Jak wysyłać pliki do klientów

Aby wysłać plik do klienta, należy wywołać metodę **SendFile** protokołu Files i przekazać **Guid** połączenia oraz pełną **FileName** jako argument. Guid połączenia klienta można przechwycić w zdarzeniu OnConnect protokołu Server Protocol Files.

Plik odebrany przez klienta zostanie domyślnie zapisany w tym samym katalogu, w którym znajduje się plik wykonywalny klienta, lub w ścieżce ustawionej we właściwości Files.SaveDirectory.

```
// ... capture the guid of the client connection to send later the file
void OnConnectEvent(TsgcWSConnection *Connection)
{
    FGuid = Connection->Guid;
}
```

```
// ... Create Server
TsgcWebSocketServer oServer = new TsgcWebSocketServer();
TsgcWSPServer_Files oServer_Files = new TsgcWSPServer_Files();
oServer_Files->Server = oServer;
oServer_Files->OnConnect = OnConnectEvent;
oServer->Host = "127.0.0.1";
oServer->Port = 8080;
```

```
// ... Create Client
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
oClient->URL = "ws://127.0.0.1:8080";
```

```
// ... Create Protocol
TsgcWSPClient_Files oClient_Files = new TsgcWSPClient_Files();
oClient_Files->Client = oClient;
```

```
// ... Start Server
oServer->Active = true;
oClient->Connect();
```

```
// ... Send File to the client connected
oServer_Files->SendFile(FGuid, "c:\\Documents\\yourfile.txt");
```

# Protokół plików | Jak wysyłać duże pliki

Jeśli chcesz wysyłać duże pliki do serwera lub klienta, na przykład pliki o rozmiarze kilku gigabajtów, próba załadowania całego pliku może powodować problemy z pamięcią. Protokół Files umożliwia wysyłanie plików w mniejszych pakietach, które po odebraniu przez drugą stronę są składane w jeden plik. Wystarczy użyć parametru **Size** metody **SendFile**, aby ustawić rozmiar w bajtach każdego pojedynczego pakietu.

```
// ... Create Server
TsgcWebSocketServer oServer = new TsgcWebSocketServer();
TsgcWSPServer_Files oServer_Files = new TsgcWSPServer_Files();
oServer_Files->Server = oServer;
oServer->Host = "127.0.0.1";
oServer->Port = 8080;

// ... Create Client
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
oClient->URL = "ws://127.0.0.1:8080";

// ... Create Protocol
TsgcWSPClient_Files oClient_Files = new TsgcWSPClient_Files();
oClient_Files->Client = oClient;

// ... Start Server
oServer->Active = true;

// ... Connect client and Send File in packets of 100000 bytes
if oClient->Connect() then
    oClient_Files->SendFile("c:\Documents\yourfile.txt", 100000, qosLevel0, "");
```

# Obecność protokołu

Protokół Presence umożliwia sprawdzenie, kto jest subskrybentem kanału. Ułatwia to tworzenie aplikacji czatu i sprawdzanie, kto jest online, na przykład: użytkownicy gry, pokoje czatu, użytkownicy przeglądający ten sam dokument...

## Funkcje

- Domyślnie użytkownik jest **identyfikowany po nazwie**, jednak można to **dostosować** przekazując dodatkowe dane: adres e-mail, firma, Twitter...
- Zdarzenia do **autoryzacji**, czy można utworzyć **kanał**, czy **członek** jest dozwolony...
- Za każdym razem gdy nowy **członek dołącza** do kanału, wszyscy członkowie są **powiadomiani**.
- **Publikuj wiadomości** do wszystkich subskrybentów kanału.
- **Niskie zużycie** pamięci.

## Komponenty

[TsgcWSPServer\\_Presence](#): Komponent VCL protokołu serwera Presence.

[TsgcWSPClient\\_Presence](#): Komponent VCL protokołu klienta Presence.

## Klasy

[TsgcWSPresenceMessage](#): obiekt hermetyzujący informacje o pakiecie obecności.

# TsgcWSPServer\_Presence

Jest to składnik protokołu Presence po stronie serwera.

## Wprowadzenie

Jest to komponent protokołu obecności serwera. Należy upuścić ten komponent na formularz i wybrać komponent [TsgcWebSocketServer](#) za pomocą właściwości Server.

## Dokumentacja

- Właściwości — 6 właściwości dostępnych dla komponentu.
- Metody — 1 metoda publiczna.
- Zdarzenia — 17 zdarzeń do obsługi.

## TsgcWSPresenceMessage

Ten obiekt hermetyzuje wszystkie wewnętrzne wiadomości wymieniane przez protokół obecności serwera i klienta.

### TsgcWSPresenceMember

**ID:** wewnętrzny identyfikator

**Name:** nazwa członka, podana przez klienta.

**Info:** dodatkowe informacje o elemencie, dostarczone przez klienta.

**Data:** TObject, który może być używany do celów po stronie serwera.

### TsgcWSPresenceMemberList

**Member[i]:** element listy według indeksu

**Count:** liczba elementów listy

### TsgcWSPresenceChannel

**Name:** nazwa kanału podana przez klienta.

### TsgcWSPresenceMsg

**Text:** wiadomość tekstowa dostarczona przez klienta podczas wywołania metody Publish

### TsgcWSPresenceError

**Code:** wartość całkowita identyfikująca błąd

**Text:** opis błędu.

# TsgcWSPClient\_Presence

To jest komponent protokołu obecności klienta.

## Wprowadzenie

Jest to komponent protokołu obecności klienta. Należy umieścić ten komponent w formularzu i wybrać komponent [TsgcWebSocketClient](#) przy użyciu właściwości Client.

## Dokumentacja

- Właściwości — 7 właściwości dostępnych w komponencie.
- Metody — 5 metod publicznych.
- Zdarzenia — 16 zdarzeń do obsługi.

# Protocol Presence Javascript

Protokół Presence w JavaScript. sgcWebSockets używa plików **sgcWebSocket.js** i **presence.esegece.com.js**.

Poniżej znajdują się dostępne metody; zmienne {%host%} i {%port%} należy zastąpić odpowiednimi wartościami. Na przykład: jeśli serwer sgcWebSocket nasłuchuje na porcie 80 pod adresem [www.example.com](http://www.example.com), należy skonfigurować:

```
<script src="http://www.example.com:80/sgcWebSockets.js"></script>
<script src="http://www.example.com:80/presence.esegece.com.js"></script>
```

## Otwórz połączenie

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
    var socket = new sgcws_presence('ws://{%host%}:{%port%}');
</script>
```

## Nowy uczestnik po połączeniu

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
    var socket = new sgcws_presence('ws://{%host%}:{%port%}');
    socket.on('sgcsession', function(event)
    {
        socket.newmember(event.id, 'John', 'Additional Info');
    });
</script>
```

## Subskrybuj kanał Topic 1

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
    var socket = new sgcws_presence('ws://{%host%}:{%port%}');
    socket.subscribe('Topic 1');
</script>
```



## Anuluj subskrypcję kanału Topic 1

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
  var socket = new sgcws_presence('ws://{%host%}:{%port%}');
  socket.unsubscribe('Topic 1');
</script>
```

## Opublikuj wiadomość do kanału Topic 1

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
  var socket = new sgcws_presence('ws://{%host%}:{%port%}');
  socket.publish('Hello sgcWebSockets!', 'Topic 1');
</script>
```

## Odbieranie wiadomości

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
  var socket = new sgcws_presence('ws://{%host%}:{%port%}');
  socket.on('sgcpublishmsg', function(event)
  {
    console.log('#publishmsg: ' + event.message.text);
  });
</script>
```

## Pobieranie wszystkich połączonych członków

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
  var socket = new sgcws_presence('ws://{%host%}:{%port%}');
  socket.on('sgcgetmembers', function(event)
  {
    for (var i in event.members) {
      console.log(event.members[i].id + ' ' + event.members[i].name);
    }
  });
  socket.getmembers();
</script>
```

## Wyświetl alert, gdy członkowie subskrybują/anulują subskrypcję

```
<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
```

```

<script>
var socket = new sgcws_presence('ws://{%host%}:{%port%}');
socket.on('sgcnewmember', function(event)
{
    alert('#new member: ' + event.member.name);
}
);
socket.on('sgcremovemember', function(event)
{
    alert('#removed member: ' + event.member.name);
}
);
socket.on('sgcnewchannelmember', function(event)
{
    alert('#new member: ' + event.member.name + ' in channel: ' + event.channel.name);
}
);
socket.on('sgcremovechannelmember', function(event)
{
    alert('#remove member: ' + event.member.name + ' from channel: ' + event.channel.name);
}
);
</script>

```

## Wyświetl alert dla zdarzeń OnConnect, OnDisconnect i OnError

```

<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
var socket = new sgcws_presence('ws://{%host%}:{%port%}');
socket.on('open', function(event)
{
    alert('sgcWebSocket Open!');
});
socket.on('close', function(event)
{
    alert('sgcWebSocket Closed!');
});
socket.on('sgcerrormemberchannel', function(event)
{
    alert('#error member channel: ' + event.error.text);
}
);
socket.on('sgcerrormapublishmsg', function(event)
{
    alert('#error publish: ' + event.error.text);
}
);
</script>

```

## Zamknij połączenie

```

<script src="http://{%host%}:{%port%}/sgcWebSockets.js"></script>
<script src="http://{%host%}:{%port%}/presence.esegece.com.js"></script>
<script>
    socket.close();
</script>

```



# Protokół E2EE

Szyfrowanie end-to-end (E2EE) oznacza, że wiadomości są szyfrowane na urządzeniu nadawcy i mogą być odszyfrowane wyłącznie na urządzeniach odbiorców. Serwer przekazuje pakiety, ale nie może odczytać treści w postaci jawnej.

Ten temat wyjaśnia przepływ techniczny dla:

- **Wiadomości bezpośrednie (1:1)** między dwoma węzłami.
- **Wiadomości grupowe** z użyciem modelu klucza nadawcy z rotacją kluczy opartą na członkostwie.

## Kryptograficzne elementy składowe

- **Tożsamość / uzgadnianie kluczy:** Elliptic Curve Diffie-Hellman (ECDH) P-256.
- **Szyfrowanie symetryczne:** AES-256-GCM.
- **Wyprowadzanie kluczy:** HKDF-SHA-256 do wyprowadzania kluczy szyfrowania ze wspólnych sekretów.
- **Uwierzytelnianie wiadomości:** Tag uwierzytelniania AEAD (dostarczany przez AES-GCM).
- **Ochrona przed odtworzeniem:** nonce/IV dla każdej wiadomości oraz liczniki sekwencji nadawcy.

## Techniczny przepływ wiadomości bezpośrednich (1:1)

1. **Odkrywanie klucza publicznego**  
Każdy partner publikuje publiczny klucz tożsamości. Serwer może dystrybuować klucze publiczne, ale klucze prywatne nigdy nie opuszczają urządzenia klienta.
2. **Ustanowienie wspólnego sekretu**  
Nadawca i odbiorca wykonują operację ECDH (klucz prywatny + publiczny klucz partnera) i uzyskują ten sam wspólny sekret.
3. **Wyprowadzanie klucza sesji**  
HKDF-SHA-256 wyprowadza jeden lub więcej kluczy symetrycznych (klucz szyfrowania, opcjonalny klucz nagłówka) z wyjścia ECDH.
4. **Szyfrowanie wiadomości**  
Tekst jawny jest szyfrowany algorytmem AES-256-GCM przy użyciu unikalnej wartości nonce/IV. Wynik zawiera szyfrogram oraz znacznik uwierzytelniający.

## 5. Transport

Serwer przekazuje zaszyfrowane ładunki i metadane (np. identyfikator nadawcy, identyfikator klucza, licznik, znacznik czasu) bez dostępu do tekstu jawnego.

## 6. Deszyfrowanie po stronie odbiorcy

Odbiorca wyprowadza ten sam klucz sesji, weryfikuje znacznik uwierzytelnienia i odszyfrowuje dane. Jakiegokolwiek manipulacje powodują błąd uwierzytelnienia.

## Techniczny przepływ wiadomości grupowych (klucze nadawcy)

W przypadku grup oddzielne szyfrowanie każdej wiadomości dla każdego członka jest kosztowne. Po-  
wszechną optymalizacją jest projektowanie z użyciem **klucza nadawcy**:

- Każdy nadawca utrzymuje **stan klucza nadawcy** (Sender Key State) dla każdej grupy.
- Stan zawiera symetryczny **klucz łańcucha** i licznik wiadomości.
- Dla każdej wychodzącej wiadomości grupowej nadawca wyprowadza jednorazowy klucz wiadomości z klucza łańcuchowego i przesuwając łańcuch (hash ratchet).
- Klucz wiadomości szyfruje ładunek przy użyciu AES-256-GCM.

## Sposób dystrybucji klucza nadawcy

1. Gdy nadawca dołącza do grupy, tworzy nowy stan klucza nadawcy.
2. Stan klucza nadawcy jest współdzielony z każdym bieżącym członkiem przez istniejące szyfrowane sesje 1:1 (parowe E2EE).
3. Po dystrybucji normalne ładunki grupowe używają szybkiej ścieżki klucza nadawcy (jedno szyfrowanie na wiadomość).

## Zmiany członkostwa i rotacja klucza nadawcy

Aby zachować poufność przekazywania i wsteczną, klucze nadawcy grupy muszą być rotowane przy zdarzeniach zmiany członkostwa:

- **Użytkownik dołącza do grupy:** należy rotować klucze nadawcy, aby nowy członek nie mógł odszyfrować starszej historii, chyba że zostanie ona udostępniona wprost.
- **Użytkownik odchodzi lub jest usuwany:** natychmiast należy obrócić klucze nadawcy, aby usunięty członek nie mógł odszyfrować nowego ruchu.
- **Rotacja okresowa:** opcjonalna rotacja oparta na czasie lub liczbie wiadomości zmniejsza wpływ kompromitacji klucza.

Typowa sekwencja rotacji:

1. Utwórz nowy stan klucza nadawcy (nowy identyfikator klucza, nowy klucz łańcucha).
2. Należy udostępniać go wyłącznie aktualnie autoryzowanym członkom za pośrednictwem parami nawiązanych kanałów E2EE.
3. Rozpoczyna szyfrowanie nowych wiadomości grupowych przy użyciu nowego identyfikatora klucza.
4. Akceptuj stary identyfikator klucza tylko przez krótki okres przejściowy, a następnie wycofaj go.

## Właściwości zabezpieczeń

- **Poufność:** tylko klienci z ważnymi kluczami mogą odszyfrować dane.
- **Integralność i autentyczność:** weryfikacja AEAD wykrywa manipulacje i sfałszowane szyfrogramy.
- **Niewidoczność serwera:** serwery przekaźnikowe nie przechowują kluczy w postaci jawnej dla treści wiadomości.
- **Ochrona po opuszczeniu grupy:** rotowane klucze uniemożliwiają byłym członkom odczyt przyszłych wiadomości grupowych.

## Komponenty

[TsgcWSPServer\\_E2EE](#): Komponent protokołu serwera E2EE. Przekazuje zaszyfrowane wiadomości między klientami bez znajomości ich treści.

[TsgcWSPClient\\_E2EE](#): Komponent protokołu klienta E2EE. Zarządza wymianą kluczy, szyfrowaniem i deszyfrowaniem na urządzeniach peer.

# TsgcWSPServer\_E2EE

Komponent protokołu serwera E2EE kieruje zaszyfrowanymi wiadomościami bezpośrednimi i grupowymi między klientami, zachowując nieprzezroczystość ładunków.

## Wprowadzenie

Składnik protokołu serwera E2EE przekierowuje zaszyfrowane wiadomości bezpośrednie i grupowe między klientami, zachowując ładunki nieprzejrzystymi. Należy go umieścić na formularzu, przypisać [TsgcWebSocketHTTPServer](#) (lub inny składnik serwera) do właściwości **Server** i skonfigurować opcje E2EE. Implementacja znajduje się w jednostce **sgcWebSocket\_Protocol\_E2EE\_Server**.

## Konfiguracja

Aby skorzystać z komponentu:

- Przypisz właściwość **Server** do skonfigurowanej instancji **TsgcWebSocketHTTPServer**.
- Konfiguracja obsługi potwierdzeń odbywa się w sekcji **E2EE\_Options.Ack**.
- Wybierz sposób obsługi nowych kluczy publicznych za pomocą właściwości **E2EE\_Options.PublicKeys.ReceiveNewPublicKey**.
- Ustaw **Server.Active := True**, aby rozpocząć nasłuchiwanie.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 0 metod publicznych.
- Zdarzenia — 8 zdarzeń do obsługi.

# TsgcWSPClient\_E2EE

Komponent protokołu klienta E2EE dodaje szyfrowanie end-to-end do połączenia WebSocket, w tym zaszyfrowane wiadomości bezpośrednie i zaszyfrowane wiadomości grupowe.

## Wprowadzenie

Komponent protokołu klienta E2EE dodaje szyfrowanie end-to-end do połączenia WebSocket, w tym szyfrowane wiadomości bezpośrednie i szyfrowane wiadomości grupowe. Należy upuścić komponent na formularz, przypisać komponent [TsgcWebSocketClient](#) do właściwości **Client** i skonfigurować opcje E2EE przed nawiązaniem połączenia. Implementacja znajduje się w module **sgcWebSocket\_Protocol\_E2EE\_Client**.

## Konfiguracja

Aby skorzystać z komponentu:

- Przypisz właściwość **Client** do skonfigurowanej instancji **TsgcWebSocketClient**.
- Przypisz unikalny identyfikator użytkownika przy użyciu właściwości **E2EE\_Options.UserId**.
- Należy włączyć potwierdzenia wiadomości za pomocą **E2EE\_Options.Ack.RcvDirectMessage** i **E2EE\_Options.Ack.RcvGroupMessage**, gdy wymagane są wywołania zwrotne stanu dostarczenia.
- Ustaw **Client.Active := True**, aby otworzyć połączenie WebSocket.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 9 publicznych metod.
- Zdarzenia — 19 zdarzeń do obsługi.



# WebSocket APIs

Istnieje wiele implementacji opartych na WebSocket: finanse, publikowanie wiadomości, kolejki... sgcWebSockets implementuje najważniejsze API oparte na protokole WebSocket. Aby skorzystać z API, wystarczy dołączyć komponent API do klienta, a wszystkie wiadomości będą obsługiwane przez ten komponent (do klienta można dołączyć tylko jeden komponent API).

## API klienckie

|                                          |                                                                                                                                                                                                                                           |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Binance</b><br><b>Binance Futures</b> | jest międzynarodową, wielojęzyczną giełdą kryptowalut.<br>umożliwia połączenie z strumieniami WebSocket / REST rynku Binance Futures.                                                                                                     |
| <b>Coinbase</b>                          | Coinbase to giełda kryptowalut z siedzibą w Stanach Zjednoczonych. Umożliwia handel kryptowalutami takimi jak Bitcoin (BTC), Ethereum (ETH) i innymi kryptowalutami za USD, EUR i GBP. Obsługuje WebSocket API.                           |
| <b>SignalR</b>                           | to biblioteka dla deweloperów ASP.NET ułatwiająca tworzenie funkcjonalności sieciowych w aplikacjach ASP.NET. SignalR umożliwia komunikację w czasie rzeczywistym między klientami a serwerem za pomocą WebSocket lub innych transportów. |
| <b>SignalRCore</b>                       | ASP.NET Core SignalR to biblioteka open-source upraszczająca dodawanie funkcjonalności w czasie rzeczywistym do aplikacji internetowych.                                                                                                  |
| <b>SocketIO</b>                          | Jest to biblioteka JavaScript do aplikacji internetowych działających w czasie rzeczywistym. Umożliwia łatwą komunikację w czasie rzeczywistym między klientami internetowymi a serwerami.                                                |
| <b>Kraken</b><br><b>Kraken Futures</b>   | Kraken to amerykańska giełda kryptowalut.<br>umożliwia połączenie z danymi rynkowymi Kraken Futures WebSocket / REST.                                                                                                                     |
| <b>Pusher</b>                            | Pusher to łatwa i niezawodna platforma z elastyczną obsługą komunikatów pub/sub, listami subskrypcyjnymi i innymi...                                                                                                                      |
| <b>Bitfinex</b>                          | Bitfinex jest jedną z największych i najbardziej zaawansowanych platform do handlu kryptowalutami. Obsługuje handel kryptowalutami takimi jak Bitcoin, Ethereum, Ripple, EOS, Bitcoin Cash, Iota, NEO, Litecoin, Ethereum Classic.        |
| <b>Bitstamp</b>                          | Bitstamp jest jedną z najdłużej działających giełd kryptowalut na świecie, wspierającą ekosystem kryptowalut.                                                                                                                             |
| <b>Huobi</b>                             | Huobi jest międzynarodową, wielojęzyczną giełdą kryptowalut.                                                                                                                                                                              |
| <b>Cex</b>                               | CEX.IO jest giełdą kryptowalut i dawnym dostawcą usług cloud miningu Bitcoin.                                                                                                                                                             |
| <b>Cex Plus</b>                          | CEX.IO Exchange Plus to zaawansowana platforma handlu kryptowalutami oferująca głęboką płynność i niskie opłaty transakcyjne.                                                                                                             |
| <b>Bitmex</b>                            | Bitmex jest giełdą kryptowalut i platformą handlu instrumentami pochodnymi.                                                                                                                                                               |
| <b>3Commas</b>                           | To bot handlowy kryptowalut.                                                                                                                                                                                                              |
| <b>Kucoin</b>                            | Kucoin jest giełdą kryptowalut umożliwiającą kupowanie, sprzedawanie i przechowywanie kryptowalut. Obsługuje handel kryptowalutami takimi jak Bitcoin, Ethereum, Litecoin, Dogecoin (DOGE).                                               |
| <b>Kucoin Futures</b>                    | umożliwia połączenie z serwerami Kucoin Futures (WebSocket i REST).                                                                                                                                                                       |
| <b>OKX</b>                               | dawniej znana jako OKEx, jest jedną z największych giełd transakcji spot i instrumentów pochodnych.                                                                                                                                       |
| <b>XTB</b>                               | XTB to polska firma zajmująca się Handel FX i CFD z dostępem do ponad 2000 rynków finansowych.                                                                                                                                            |
| <b>Discord</b>                           | Discord jest jednym z najpopularniejszych narzędzi komunikacyjnych w grach online i streamingu.                                                                                                                                           |
| <b>Bybit</b>                             | Bybit jest giełdą kryptowalut i platformą handlową.                                                                                                                                                                                       |
| <b>OpenAI API</b>                        | OpenAI Realtime API umożliwia interakcje niskolatencyjne z obsługą wielu modalności, w tym transkrypcję w czasie rzeczywistym.                                                                                                            |

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <b>MEXC</b>         | Scentralizowana giełda kryptowalut i platforma handlowa; ten składnik implementuje SPOT A |
| <b>MEXC Futures</b> | Scentralizowana giełda kryptowalut i platforma transakcyjna; komponent implementuje API F |
| <b>Bitget</b>       | Giełda kryptowalut i platforma handlowa obsługująca rynki Spot i Futures.                 |
| <b>Gate.io</b>      | Giełda kryptowalut obsługująca handel spot i kontraktami futures z API WebSocket i REST.  |
| <b>Deribit</b>      | Giełda instrumentów pochodnych kryptowalut oferująca handel kontraktami futures i opcjami |
| <b>Crypto.com</b>   | Giełda kryptowalut obsługująca kanały rynkowe i użytkownika za pośrednictwem interfejsów  |
| <b>HTX</b>          | Międzynarodowa giełda kryptowalut (dawniej Huobi) z REST API do danych rynkowych i ha     |

WebSocket API można rejestrować w czasie **wykonania**; wystarczy wywołać metodę **RegisterAPI** i przekazać komponent API jako parametr.

## Inne interfejsy API klienta

|                     |                                                                                                                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Telegram</b>     | to oparta na chmurze usługa natychmiastowego przesyłania wiadomości i telefonii internetowej, umożliwia wymieniać zdjęcia, filmy, naklejki, pliki audio i pliki dowolnego typu. |
| <b>Whatsapp</b>     | Ogólnoświatowa, bezpłatna, wielopatformowa, scentralizowana usługa komunikacji błyskawicznej                                                                                    |
| <b>RCON</b>         | to protokół komunikacyjny oparty na TCP/IP, który umożliwia wydawanie poleceń konsolowych                                                                                       |
| <b>CryptoHopper</b> | Jest to bot do handlu kryptowalutami i menedżer portfela.                                                                                                                       |
| CryptoRobotics      | To robot do handlu kryptowalutami.                                                                                                                                              |

## API serwera

|                           |                                                                                                                                               |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RTCMultiConnection</b> | RTCMultiConnection to biblioteka JavaScript WebRTC do aplikacji peer-to-peer (udziało, udostępnianie plików, strumieniowanie mediów itp.).    |
| <b>WebPush</b>            | Protokół Web Push umożliwia aplikacjom internetowym wysyłanie powiadomień do użytkowników, których przeglądarka nie jest otwarta ani aktywna. |
| <b>WebAuthn</b>           | Serwer FIDO2/WebAuthn API do uwierzytelniania bezhasłowego z użyciem biometrii i kluczy prywatnych.                                           |

# API Binance

## Binance

Binance to międzynarodowa, wielojęzyczna giełda kryptowalut. Oferuje kilka API do uzyskiwania dostępu do danych Binance. Obsługiwane są następujące API:

1. **Strumienie WebSocket:** pozwalają subskrybować określone metody i otrzymywać dane w czasie rzeczywistym. Zdarzenia są wypychane do klientów przez serwer do subskrybentów. Używa protokołu WebSocket.
2. **UserData stream:** subskrybujący klienci otrzymują szczegóły konta. Wymaga klucza API do uwierzytelnienia i używa WebSocket jako protokołu.
3. **REST API:** Wymaga klucza API i sekretu do uwierzytelnienia i używa HTTPS jako protokołu.
  1. [Dane rynkowe](#)
  2. [Dane konta i handlu](#)
  3. [Portfel](#)
4. **Futures:** strumieniowe transmisje danych rynkowych WebSocket Futures są obsługiwane przez [interfejs API klienta Binance Futures](#).

Klient obsługuje również **Binance.us**; obsługiwane są następujące API:

1. **Strumienie WebSocket:** pozwalają subskrybować określone metody i otrzymywać dane w czasie rzeczywistym. Zdarzenia są wypychane do klientów przez serwer do subskrybentów. Używa protokołu WebSocket.
2. **UserData stream:** subskrybujący klienci otrzymują szczegóły konta. Wymaga klucza API do uwierzytelnienia i używa WebSocket jako protokołu.
3. **REST API:** klienci mogą żądać danych rynkowych i konta od serwera. Wymaga klucza API i sekretu do uwierzytelnienia; protokołem jest HTTPS.

## Właściwości

API Binance ma 2 typy metod: publiczne i prywatne. Do metod publicznych można uzyskać dostęp bez uwierzytelnienia, na przykład: pobieranie cen tickerów. Niektóre są prywatne i dotyczą danych użytkownika; te metody wymagają użycia kluczy API Binance.

- **ApiKey:** nowy klucz API można wygenerować na koncie Binance — wystarczy skopiować wartość do tej właściwości.
- **ApiSecret:** ApiSecret jest wymagany tylko dla REST\_API; websocket API wymaga ApiKey jedynie dla niektórych metod.
- **TestNet:** jeśli włączone, połączy się z kontem demonstracyjnym Binance (domyślnie false).
  - **HTTPLogOptions:** przechowuje dziennik żądań HTTP w pliku tekstowym
    - **Enabled:** jeśli włączone, zapisuje wszystkie żądania HTTP interfejsu WebSocket API.
    - **FileName:** pełna ścieżka do pliku, w którym będą przechowywane dzienniki
    - **REST:** przechowuje dziennik żądań REST API w pliku tekstowym
      - **Enabled:** jeśli włączone, zapisuje wszystkie żądania HTTP interfejsu REST API.
      - **FileName:** pełna ścieżka do pliku, w którym będą przechowywane dzienniki.
- **UserStream:** jeśli włączone, klient będzie otrzymywać powiadomienia o aktualizacjach konta, zleceń lub salda (domyślnie true).
- **BinanceUS:** jeśli jest włączone, połączenie zostanie nawiązane z serwerami Binance.us (zamiast domyślnych serwerów Binance.com).

- **ListenKeyOnDisconnect:** ta właściwość określa, co należy zrobić po rozłączeniu klienta z serwerami Binance przy aktywnym kluczu ListenKey.
  - **blkodDeleteListenKey:** Usuwa aktywny klucz ListenKey, wysyłając żądanie HTTP do serwerów Binance (opcja domyślna).
  - **blkodClearListenKey:** Nie usuwa klucza ListenKey z serwerów Binance i jedynie czyści wartość pola.
  - **blkodDoNothing:** Nie wykonuje żadnej akcji, więc przy następnym połączeniu z Binance zostanie podjęta próba użycia tego samego klucza ListenKey.
- **UseCombinedStreams:** jeśli włączone, strumienie będą łączone w następujący sposób: {"stream":"<streamName>","data":<rawPayload>} (domyślnie wyłączone)

## Najczęstsze zastosowania

- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować kanał WebSocket](#)
- **REST API**
  - [Jak pobierać dane rynkowe](#)
  - [Jak korzystać z prywatnego REST API](#)
  - [Jak handlować na rynku spot](#)
  - [Czas prywatnych żądań](#)
  - [Wypłata](#)

## WebSocket Stream API

Bazowy endpoint to wss://stream.binance.com:9443; klient może subskrybować zdarzenia i anulować subskrypcje po nawiązaniu pomyślnego połączenia.

Obsługiwane są następujące metody subskrypcji i anulowania subskrypcji.

|                  |                  |                                                                                                                                                                                                                                                                                                            |
|------------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AggregateTrades  | Symbol           | przesyłanie zagregowanych informacji o transakcjach dla jednego zlecenia taker                                                                                                                                                                                                                             |
| Transakcje       | Symbol           | przesyłaj surowe informacje o transakcjach; każda transakcja ma unikalnego kupującego i sprzedającego                                                                                                                                                                                                      |
| KLine            | Symbol, interwał | wypycha aktualizacje bieżących wykresów klines/świecowych co sekundę, minutę, godzinę...                                                                                                                                                                                                                   |
| MiniTicker       | Symbol           | Statystyki mini-tickera w 24-godzinny oknie kroczącym. NIE są to statystyki dnia UTC, lecz 24-godzinne okno kroczące dla poprzednich 24 godzin.                                                                                                                                                            |
| AllMiniTickers   |                  | Statystyki mini-tickera w 24-godzinny oknie kroczącym dla wszystkich symboli, których wartości uległy zmianie, zebrane w tablicy. Nie są to statystyki dnia UTC, lecz 24-godzinne okno kroczące obejmujące poprzednie 24 godziny. W tablicy znajdują się wyłącznie tickery, których wartości się zmieniły. |
| Ticker           | Symbol           | Statystyki tickera dla kroczącego okna 24-godzinnego dla pojedynczego symbolu. NIE są to statystyki dnia UTC, lecz kroczące okno 24-godzinne dla poprzednich 24 godzin.                                                                                                                                    |
| AllMarketTickers |                  | Statystyki tickera dla kroczącego okna 24-godzinnego dla wszystkich symboli, które zmieniły się w tablicy. NIE są to statystyki dnia UTC, lecz kroczące okno 24-godzinne                                                                                                                                   |

|                  |                   |                                                                                                                                    |
|------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------|
|                  |                   | dla poprzednich 24 godzin. Uwaga: w tablicy będą obecne tylko tickery, które uległy zmianie.                                       |
| BookTicker       | Symbol            | Przekazuje w czasie rzeczywistym aktualizacje najlepszej ceny lub wolumenu po stronie bid lub ask dla wskazanego symbolu.          |
| AllBookTickers   |                   | Przesyła w czasie rzeczywistym wszelkie aktualizacje ceny lub ilości najlepszej oferty kupna lub sprzedaży dla wszystkich symboli. |
| PartialBookDepth | Symbol, Głębokość | Najlepsze <levels> ofert kupna i sprzedaży, przesyłane co sekundę. Prawidłowe wartości <levels> to 5, 10 lub 20.                   |
| DiffDepth        | Symbol            | Aktualizacje głębokości ceny i ilości w księdze zleceń, używane do lokalnego zarządzania księgą zleceń.                            |

Po pomyślnej subskrypcji / anulowaniu subskrypcji klient otrzymuje wiadomość o tym fakcie, gdzie id jest wynikiem metody Subscribed / Unsubscribed.

```
{
  "result": null,
  "id": 1
}
```

## User Data Stream API

Wymaga prawidłowego ApiKey uzyskanego z konta Binance; ApiKey musi być ustawiony we właściwości Binance.ApiKey komponentu.

Następujące dane są wypychane do klienta za każdym razem, gdy nastąpi zmiana. Nie jest konieczna subskrypcja żadnej metody — odbywa się ona automatycznie po ustawieniu prawidłowego ApiKey.

|                       |                                                                                                                                                |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Aktualizacja konta    | Stan konta jest aktualizowany za pomocą zdarzenia outboundAccountInfo.                                                                         |
|                       | Aktualizacja salda następuje w następujących przypadkach:                                                                                      |
| Aktualizacja salda    | <ul style="list-style-type: none"> <li>• Wpłaty lub wypłaty z konta</li> <li>• Transfer środków między kontami (np. Spot do Margin)</li> </ul> |
| Aktualizacja zlecenia | Zlecenia są aktualizowane zdarzeniem executionReport.                                                                                          |

## REST API

Bazowy punkt końcowy to: <https://api.binance.com>. Wszystkie punkty końcowe zwracają obiekt lub tablicę JSON. Dane są zwracane w porządku rosnącym: od najstarszych do najnowszych.

Dostęp do opcji REST API przy użyciu właściwości `REST_API.BinanceOptions`.

## Publiczne punkty końcowe API

Te punkty końcowe są dostępne bez żadnej autoryzacji.

### Ogólne punkty końcowe

|                        |                                                               |
|------------------------|---------------------------------------------------------------|
| Ping                   | Testuje łączność z Rest API.                                  |
| GetServerTime          | Sprawdź połączenie z Rest API i pobierz bieżący czas serwera. |
| GetExchangeInformation | Bieżące reguły handlowe giełdy i informacje o symbolach       |

### Punkty końcowe danych rynkowych

|                     |                  |                                                                                                                                                                     |
|---------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetOrderBook        | Symbol           | Pobierz arkusz zleceń.                                                                                                                                              |
| GetTrades           | Symbol           | Pobierz ostatnie transakcje                                                                                                                                         |
| GetHistoricalTrades | Symbol           | Pobierz starsze transakcje.                                                                                                                                         |
| GetAggregateTrades  | Symbol           | Pobierz skompresowane, zagregowane transakcje. Transakcje realizowane w tym samym czasie, z tego samego zlecenia i po tej samej cenie będą miały zagregowaną ilość. |
| GetKLines           | Symbol, interwał | Świece Kline dla instrumentu. Świece są jednoznacznie identyfikowane przez czas otwarcia.                                                                           |
| GetAveragePrice     | Symbol           | Bieżąca średnia cena dla symbolu.                                                                                                                                   |
| Get24hrTicker       | Symbol           | Statystyki zmian cen w 24-godzinny oknie kroczącym. Należy zachować ostrożność podczas wywoływania bez podania symbolu.                                             |
| GetPriceTicker      | Symbol           | Ostatnia cena dla danego symbolu.                                                                                                                                   |
| GetPriceTickers     | Symbols          | Ostatnia cena dla tablicy symboli. Przykład: ["BTCUSDT","BNBUSDT"]                                                                                                  |
| GetBookTicker       | Symbol           | Najlepsza cena/ilość w arkuszu zleceń dla jednego lub wielu symboli.                                                                                                |

|                         |                             |                                                                                                                                           |
|-------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| GetUIKLines             | Symbol, interwał            | Słupki Kline/świecowe dla danego symbolu. Odpowiedź jest podobna do GetKLines, zoptymalizowana pod kątem prezentacji wykresów świecowych. |
| GetRollingWindow-Ticker | Symbol, Symbols, WindowSize | Statystyki zmiany ceny w kroczącej oknie czasowym. Uwaga: domyślna wartość WindowSize to 1d, jeśli nie zostanie podana.                   |
| GetTradingDayTicker     | Symbol, Symbols, Type       | Statystyki zmian ceny dla dnia handlowego.                                                                                                |

## Prywatne punkty końcowe API

Wymaga podania APIKey i APISecret w celu autoryzacji przez serwer.

### Punkty końcowe danych konta

|                              |                                                   |                                                                                                                                                  |
|------------------------------|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| NewOrder                     | Symbol, strona, typ                               | Wyślij nowe zlecenie.                                                                                                                            |
| PlaceMarketOrder             | Side, Symbol, Quantity                            | Składa nowe zlecenie rynkowe                                                                                                                     |
| PlaceMarketQuote-Order       | Side, Symbol, QuoteOrderQty                       | Składa nowe zlecenie rynkowe z kwotowaniem                                                                                                       |
| PlaceLimitOrder              | Side, Symbol, Quantity, LimitPrice                | Składa nowe zlecenie Limit                                                                                                                       |
| PlaceStopOrder               | Side, Symbol, Quantity, StopPrice, LimitPrice     | Składa nowe zlecenie stop                                                                                                                        |
| PlaceStopTrailingOrder       | Side, Symbol, Quantity, TrailingDelta, LimitPrice | Składa nowe zlecenie Stop Trailing                                                                                                               |
| PlaceTakeProfitOrder         | Side, Symbol, Quantity, StopPrice, LimitPrice     | Składa nowe zlecenie Take Profit                                                                                                                 |
| PlaceTakeProfitTrailingOrder | Side, Symbol, Quantity, TrailingDelta, LimitPrice | Składa nowe zlecenie Take Profit Trailing                                                                                                        |
| PlaceLimitMakerOrder         | Side, Symbol, Quantity                            | Składa nowe zlecenie z limitem ceny rynkowej                                                                                                     |
| TestNewOrder                 | Symbol, strona, typ                               | Testuje tworzenie nowego zlecenia oraz poprawność podpisu/recvWindow. Tworzy i weryfikuje nowe zlecenie, ale nie wysyła go do silnika dopasowań. |
| QueryOrder                   | Symbol                                            | Sprawdź status zlecenia.                                                                                                                         |
| CancelOrder                  | Symbol                                            | Anuluje aktywne zlecenie. Należy podać OrderId lub OrigClientOrderId.                                                                            |
| CancelAllOpenOrders          | Symbol (opcjonalnie)                              |                                                                                                                                                  |
| GetOpenOrders                |                                                   | Pobierz wszystkie otwarte zlecenia dla symbolu. Zachować ostrożność przy dostępie bez podania symbolu.                                           |
| GetAllOrders                 | Symbol                                            | Pobierz wszystkie zlecenia konta: aktywne, anulowane lub zrealizowane.                                                                           |

|                        |                                                                                                                                   |                                                                                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| NewOCO                 | Symbol, Side, Quantity, Price, Stop-Price                                                                                         | Wyślij nowe zlecenie OCO                                                                                                                                |
| CancelOCO              | Symbol                                                                                                                            | Anuluj całą listę zleceń                                                                                                                                |
| QueryOCO               | Symbol                                                                                                                            | Pobiera określone zlecenie OCO na podstawie podanych opcjonalnych parametrów.                                                                           |
| GetAlloCO              |                                                                                                                                   | Pobiera wszystkie zlecenia OCO na podstawie podanych opcjonalnych parametrów                                                                            |
| GetOpenOCO             |                                                                                                                                   | Pobierz wszystkie otwarte OCO.                                                                                                                          |
| GetAccountInformation  |                                                                                                                                   | Pobierz informacje o bieżącym koncie.                                                                                                                   |
| GetAccountTradeList    | Symbol                                                                                                                            | Pobierz transakcje dla określonego konta i symbolu.                                                                                                     |
| CancelReplaceOrder     | Symbol, Side, Type, CancelReplace-Mode                                                                                            | Anuluje istniejące zlecenie i składa nowe zlecenie na tym samym instrumencie.                                                                           |
| NewOrderListOCO        | Symbol, Side, Quantity, AboveType, BelowType                                                                                      | Dodaj nowe zlecenie listy OCO.                                                                                                                          |
| NewOrderListOTO        | Symbol, WorkingType, WorkingSide, WorkingQuantity, WorkingPrice, PendingType, PendingSide, PendingQuantity                        | Złóż nowe zlecenie OTO (One-Triggers-the-Other).                                                                                                        |
| NewOrderListOTO-CO     | Symbol, WorkingType, WorkingSide, WorkingQuantity, WorkingPrice, PendingSide, PendingAboveType, PendingBelowType, PendingQuantity | Złóż nowe zlecenie OTOCO (One-Triggers-a-One-Cancels-the-Other).                                                                                        |
| NewSOROrder            | Symbol, Side, Type, Quantity                                                                                                      | Składa zlecenie przy użyciu Smart Order Routing (SOR).                                                                                                  |
| TestSOROrder           | Symbol, Side, Type, Quantity                                                                                                      | Testuje nowe zlecenie przy użyciu inteligentnego routingu zleceń (SOR). Tworzy i weryfikuje nowe zlecenie, lecz nie wysyła go do silnika dopasowywania. |
| GetOrderRateLimitUsage |                                                                                                                                   | Wyświetla bieżące wykorzystanie liczby zleceń użytkownika dla wszystkich interwałów.                                                                    |
| GetPreventedMatches    | Symbol                                                                                                                            | Wyświetla listę zleceń, które wygasły z powodu STP (Self Trade Prevention).                                                                             |
| GetAllocations         | Symbol                                                                                                                            | Pobiera alokacje wynikające z rozmieszczenia zleceń SOR.                                                                                                |
| GetAccountCommission   | Symbol                                                                                                                            | Pobierz bieżące stawki prowizji konta.                                                                                                                  |

## Convert EndPoints

|                         |                    |                                                                                                            |
|-------------------------|--------------------|------------------------------------------------------------------------------------------------------------|
| GetAllConvertPairs      | FromAsset, ToAsset | Zapytaj o wszystkie pary tokenów podlegające konwersji oraz odpowiadające im górne i dolne limity tokenów. |
| GetConvertAssetInfo     |                    | Zapytanie o informacje dotyczące precyzji obsługiwanych aktywów                                            |
| SendConvertQuoteRequest | FromAsset, ToAsset | Żądaj wyceny dla wybranych par tokenów                                                                     |
| AcceptConvertQuote      | QuoteId            | Zaakceptuj oferowaną wycenę na podstawie identyfikatora wyceny.                                            |



|                           |                                          |                                                                                                                                                                                                    |
|---------------------------|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetConvertOrderStatus     | OrderId lub QuoteId                      | Zapytaj o status zlecenia na podstawie jego identyfikatora.<br><br>Umożliwia użytkownikom złożenie zlecenia z limitem ceny.                                                                        |
| PlaceConvertLimitOrder    | BaseAsset, QuoteAsset, Side, Limit-Price | baseAsset lub quoteAsset można określić za pomocą punktu końcowego exchangeInfo.<br>Cena limitowa jest zdefiniowana od baseAsset do quoteAsset.<br>Używany jest albo baseAmount, albo quoteAmount. |
| CancelConvertLimitOrder   | OrderId                                  | Umożliwia użytkownikom anulowanie zlecenia z limitem ceny                                                                                                                                          |
| GetConvertLimitOpenOrders |                                          | Umożliwia użytkownikom zapytanie o wszystkie istniejące zlecenia z limitem ceny.                                                                                                                   |
| GetConvertTradeHistory    | StartTime, EndTime                       | Maksymalny interwał między startTime a endTime wynosi 30 dni.                                                                                                                                      |

## Wallet EndPoints

(\*endpointy portfela działają wyłącznie z serwerem produkcyjnym, nie z wersją demonstracyjną)

|                                    |                                                                                                                                                                                                                                                                                        |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetWalletSystemStatus              | Pobierz status systemu.                                                                                                                                                                                                                                                                |
| GetWalletAllCoinsInformation       | Pobierz informacje o monetach (dostępnych do wpłat i wypłat) dla użytkownika.<br>Typ: "SPOT", "MARGIN", "FUTURES"                                                                                                                                                                      |
| GetWalletDailyAccountSnapshot      | <ul style="list-style-type: none"> <li>Okres zapytania musi być krótszy niż 30 dni</li> <li>Obsługiwane są wyłącznie zapytania w ciągu ostatniego miesiąca</li> <li>Jeśli parametry startTime i endTime nie zostaną podane, domyślnie zwracane są rekordy z ostatnich 7 dni</li> </ul> |
| SetWalletDisableFastWithdrawSwitch | To żądanie wyłączy przełącznik szybkiego wypłacania na Twoim koncie.<br>Należy włączyć opcję "trade" dla klucza API, który wysyła żądanie do tego punktu końcowego.<br>To żądanie włącza przełącznik szybkiej wypłaty na koncie.                                                       |
| SetWalletEnableFastWithdrawSwitch  | Należy włączyć opcję "trade" dla klucza API, który wywołuje ten punkt końcowy.<br>Gdy przełącznik szybkiej wypłaty jest włączony, transfer środków na konto Binance odbywa się natychmiastowo. Nie ma transakcji on-chain, identyfikatora transakcji ani opłaty za wypłatę.            |
| WalletWithdraw                     | Prześlij żądanie wypłaty.                                                                                                                                                                                                                                                              |
| GetWalletDepositHistory            | Pobierz historię wpłat.                                                                                                                                                                                                                                                                |
| GetWalletWithdrawHistory           | Pobierz historię wypłat.                                                                                                                                                                                                                                                               |
| GetWalletDepositAddress            | Pobierz adres depozytu wraz z siecią.                                                                                                                                                                                                                                                  |
| GetWalletAccountStatus             | Pobierz szczegóły stanu konta.                                                                                                                                                                                                                                                         |

|                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetWalletAccountAPITrading-Status          | Pobiera szczegóły statusu handlu API konta.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| GetWalletDustLog                           | Zwraca tylko ostatnie 100 rekordów<br>Zwraca tylko rekordy po 2020/12/01                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| GetWalletAssetsConvertedBNB                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| WalletDustTransfer                         | Konwertuj aktywa dust na BNB.<br><br>Należy włączyć uprawnienia Spot & Margin Trading dla klucza API, który korzysta z tego endpointu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| GetWalletAssetDividendRecord               | Zapytanie o rekordy dywidend z aktywów.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| GetWalletAssetDetail                       | Pobierz szczegóły zasobów obsługiwanych na Binance.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| GetWalletTradeFee                          | Pobierz opłatę transakcyjną<br><br>Należy włączyć opcję Permits Universal Transfer dla klucza API, który żąda tego punktu końcowego.<br>MAIN_UMFUTURE Przelew z konta Spot na konto USDⓈ-M Futures<br><br>ENUM typu: <ul style="list-style-type: none"> <li>MAIN_CMFUTURE Przelew z konta Spot na konto COIN-M Futures</li> <li>MAIN_MARGIN Przelew z konta Spot na konto Margin (cross)</li> <li>UMFUTURE_MAIN Przelew z konta USDⓈ-M Futures na konto Spot</li> <li>UMFUTURE_MARGIN Przelew z konta USDⓈ-M Futures na konto Margin (cross)</li> <li>CMFUTURE_MAIN Przelew z konta COIN-M Futures na konto Spot</li> <li>CMFUTURE_MARGIN Przelew z konta COIN-M Futures na konto Margin(cross)</li> <li>MARGIN_MAIN Przelew z konta Margin (cross) na konto Spot</li> <li>MARGIN_UMFUTURE Przelew z konta Margin (cross) na USDⓈ-M Futures</li> <li>MARGIN_CMFUTURE Przelew z konta Margin (cross) na COIN-M Futures</li> <li>ISOLATEDMARGIN_MARGIN Przelew z izolowanego konta margin na konto Margin(cross)</li> <li>MARGIN_ISOLATEDMARGIN Przelew z konta Margin(cross) na izolowane konto margin</li> <li>ISOLATEDMARGIN_ISOLATEDMARGIN Przelew z izolowanego konta margin na izolowane konto margin</li> <li>MAIN_FUNDING Przelew z konta Spot na konto Funding</li> <li>FUNDING_MAIN Przelew z konta Funding na konto Spot</li> <li>FUNDING_UMFUTURE Przelew z konta Funding na konto UMFUTURE</li> <li>UMFUTURE_FUNDING Przelew z konta UMFUTURE na konto Funding</li> <li>MARGIN_FUNDING Przelew z konta MARGIN na konto Funding</li> <li>FUNDING_MARGIN Przelew z konta Funding na konto Margin</li> <li>FUNDING_CMFUTURE Przelew z konta Funding na konto CMFUTURE</li> <li>CMFUTURE_FUNDING Przelew z konta CMFUTURE na konto Funding</li> </ul> |
| WalletUserUniversalTransfer                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| GetWalletQueryUserUniversalTransferHistory | <ul style="list-style-type: none"> <li>fromSymbol musi być przesłane, gdy typ to ISOLATEDMARGIN_MARGIN lub ISOLATEDMARGIN_ISOLATEDMARGIN</li> <li>toSymbol musi być przesłane, gdy typ to MARGIN_ISOLATEDMARGIN lub ISOLATEDMARGIN_ISOLATEDMARGIN</li> <li>Obsługa zapytań wyłącznie z ostatnich 6 miesięcy</li> <li>Jeśli startTime i endTime nie zostaną przesłane, domyślnie zwracane są rekordy z ostatnich 7 dni</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| GetWalletFundingWallet                     | Obsługuje aktualnie zapytania dotyczących następujących aktywów biznesowych: Binance Pay, Binance Card, Binance Gift Card, Stock Token                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| GetWalletUserAsset                         | Pobierz zasoby użytkownika, tylko dla danych dodatnich.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## Zdarzenia

Wiadomości Binance są odbierane w komponencie TsgcWebSocketClient; można używać następujących zdarzeń:

### OnConnect

Po pomyślnym połączeniu z serwerem Binance.

### OnDisconnect

Po rozłączeniu z serwerem Binance

### OnMessage

Wiadomości wysyłane przez serwer do klienta są obsługiwane w tym zdarzeniu.

### OnError

W przypadku wystąpienia błędu protokołu wywoływane jest to zdarzenie.

### OnException

Jeżeli wystąpi nieobsługiwany wyjątek, zostanie wywołane to zdarzenie.

Dodatkowo w komponencie Binance API istnieje specyficzne zdarzenie o nazwie **OnBinanceHTTPException**, wywoływane za każdym razem, gdy wystąpi błąd podczas wywołania żądania HTTP (REST API lub strumień użytkownika WebSocket).

(\*) Ze względu na zmiany na serwerach Binance, wersje Indy starsze niż Rad Studio 10.1 nie będą mogły łączyć się z serwerami testowymi. Problem ten nie dotyczy wersji Enterprise ani przypadku, gdy wersja Indy została zaktualizowana do najnowszej.

# Binance | Połącz z WebSocket API

Aby połączyć się z API WebSocket Binance, wystarczy utworzyć nowego klienta API Binance i podłączyć go do TsgcWebSocketClient.

Poniżej znajduje się przykład:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(NULL);
oBinance->Client = oClient;
oClient->Active = true;
```

# Binance | Subskrybuj kanał WebSocket

Binance oferuje wiele kanałów, na które można subskrybować w celu otrzymywania aktualizacji danych rynkowych, zleceń w czasie rzeczywistym... Poniżej przedstawiono przykład subskrypcji Ticker:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(NULL);
oBinance->Client = oClient;
oBinance->SubscribeTicker("bnbbtc");

void OnMessage(TsgcWSConnection *Connection, const string aText)
{
    // here you will receive the ticker updates
}
```

## Binance | Get Market Data

Binance udostępnia publiczne dane rynkowe przez punkty końcowe REST. Wywołanie jednego z tych punktów końcowych zwraca migawkę żądanych danych rynkowych.

Punkty końcowe danych rynkowych nie wymagają uwierzytelniania i są ogólnodostępne dla wszystkich użytkowników.

**Przykład:** aby uzyskać migawkę tickera BNBBTC, należy wykonać następujące wywołanie:

```
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(this);  
ShowMessage(oBinance->REST_API->GetPriceTicker("BNBBTC"));
```

# Binance | Prywatne REST API

Binance REST API oferuje publiczne i prywatne punkty końcowe. Prywatne punkty końcowe wymagają podpisywania wiadomości w celu zwiększenia bezpieczeństwa transakcji.

Najpierw należy zalogować się na konto Binance i utworzyć nowy interfejs API; zostaną wówczas udostępnione następujące wartości:

- ApiKey
- ApiSecret

Te pola muszą być skonfigurowane we właściwości Binance komponentu klienckiego Binance API.

Po skonfigurowaniu można przystąpić do wysyłania prywatnych żądań do Binance Pro REST API.

\*Żądania prywatne wymagają synchronizacji czasu lokalnego z serwerem. W przeciwnym razie żądania zostaną odrzucone przez serwer Binance. Więcej informacji na ten temat zawiera artykuł [Binance Private Requests Time](#).

```
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(this);  
oBinance->ApiKey = "<your api key>";  
oBinance->ApiSecret = "<your api secret>";  
ShowMessage(oBinance->REST_API->GetAccountInformation());
```

# Binance | Handel spot

Binance umożliwia handel spot przy użyciu swojego API REST.

## Konfiguracja

Najpierw należy utworzyć **klucz API** na koncie Binance i przyznać uprawnienia do handlu Spot.

Po wykonaniu tych czynności można rozpocząć handel spot.

Najpierw **ustaw ApiKey i ApiSecret** w komponencie klienta Binance; zostaną one użyte do podpisywania żądań wysyłanych do serwera Binance.

## Złóż zlecenie

Aby złożyć nowe zlecenie, należy wywołać metodę **REST\_API.NewOrder** komponentu klienta Binance.

W zależności od rodzaju zlecenia (rynkowe, limitowane...) API wymaga podania mniejszej lub większej liczby pól.

## Pola obowiązkowe

- **Symbol:** symbol identyfikatora produktu, przykład: BNBBTC
- **Side:** BUY lub SELL
- **type:** typ zlecenia
  - LIMIT
  - MARKET
  - STOP\_LOSS
  - STOP\_LOSS\_LIMIT
  - TAKE\_PROFIT
  - TAKE\_PROFIT\_LIMIT
  - LIMIT\_MAKER



## Dodatkowe obowiązkowe pola w zależności od typu

- **LIMIT:** timeInForce, quantity, price
- **MARKET:** quantity lub quoteOrderQty
- **STOP\_LOSS / TAKE\_PROFIT:** ilość, stopPrice
- **STOP\_LOSS\_LIMIT / TAKE\_PROFIT\_LIMIT:** timeInForce, quantity, price, stopPrice
- **LIMIT\_MAKER:** quantity, price

Podczas składania zlecenia możliwe są 2 scenariusze:

1. **Powodzenie:** funkcja NewOrder zwraca wiadomość wysłaną przez serwer Binance.
2. **Błąd:** wyjątek jest zwracany w zdarzeniu OnBinanceHTTPException.

### Złóż zlecenie rynkowe 1 BNBBTC

```
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(this);
oBinance->Binance->ApiKey = "<api key>";
oBinance->Binance->ApiSecret = "<api secret>";
ShowMessage(oBinance->REST_API->NewOrder("BNBBTC", "BUY", "MARKET", "", 1));
```

-

### Złóż zlecenie Limit 1 BNBBTC po cenie 0.009260

```
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(this);
oBinance->Binance->ApiKey = "<api key>";
oBinance->Binance->ApiSecret = "<api secret>";
ShowMessage(oBinance->REST_API->NewOrder("BNBBTC", "BUY", "LIMIT", "GTC", 1, 0, 0.009260));
```

-

## Binance | Czas prywatnych żądań

Podczas wykonywania prywatnego żądania do Binance wiadomość jest podpisywana w celu zwiększenia bezpieczeństwa żądań. Wiadomość zawiera lokalny czas i jest wysyłana w podpisanej wiadomości; jeśli czas lokalny różni się o więcej niż 5 sekund od czasu serwerów Binance, żądanie zostanie odrzucone. Dlatego ważne jest, aby zweryfikować, czy czas lokalny jest zsynchronizowany; można to zrobić za pomocą metody synchronizacji czasu właściwej dla systemu operacyjnego.

Logika jest następująca:

```
if (timestamp < (serverTime + 1000) && (serverTime - timestamp) <= recvWindow) {  
    // process request  
} else {  
    // reject request  
}
```

Zaleca się używanie małego `recvWindow` wynoszącego 5000 lub mniej! Maksymalna wartość nie może przekraczać 60000 milisekund.

Można sprawdzić czas serwera Binance, wywołując metodę **GetServerTime**, która zwróci aktualny czas serwera Binance.

Wartość domyślna **RecvWindow** wynosi **5000**; można ją zwiększyć przy użyciu właściwości **REST\_API.BinanceOptions.RecvWindow**.

# Binance | Wypłata

Binance umożliwia korzystanie z Wallet API do przesyłania żądania wypłaty; wymagane są wyłącznie następujące parametry:

- Moneta
- Adres
- Kwota

```
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(this);  
oBinance->Binance->ApiKey = "<your api key>";  
oBinance->Binance->ApiSecret = "<your api secret>";  
ShowMessage(oBinance->REST_API->WalletWithdraw("BTC", "7213fea8e94b4a5593d507237e5a555b", 0.25));
```

# API Binance Futures

## Binance

Binance to międzynarodowa wielojęzyczna giełda kryptowalut. Oferuje szereg API umożliwiających dostęp do danych Binance. Ten komponent umożliwia pobieranie strumieni rynkowych WebSocket dla kontraktów Binance Futures.

<https://binance-docs.github.io/apidocs/futures/en>

<https://binance-docs.github.io/apidocs/delivery/en>

## Kontrakty futures

Binance API posiada 2 typy metod: publiczne i prywatne. Metody publiczne są dostępne bez uwierzytelniania, np. pobieranie cen tickerów. Niektóre metody są prywatne i związane z danymi użytkownika; te metody wymagają użycia kluczy API Binance.

- **ApiKey:** nowy klucz API można wygenerować na koncie Binance — wystarczy skopiować wartość do tej właściwości.
- **ApiSecret:** ApiSecret jest wymagany tylko dla REST\_API; websocket API wymaga ApiKey jedynie dla niektórych metod.
- **TestNet:** jeśli włączone, połączy się z kontem demonstracyjnym Binance (domyślnie false).
  - **HTTPLogOptions:** przechowuje dziennik żądań HTTP w pliku tekstowym
    - **Enabled:** jeśli włączone, zapisuje wszystkie żądania HTTP interfejsu WebSocket API.
    - **FileName:** pełna ścieżka do pliku, w którym będą przechowywane dzienniki
    - **REST:** przechowuje dziennik żądań REST API w pliku tekstowym
      - **Enabled:** jeśli włączone, zapisuje wszystkie żądania HTTP interfejsu REST API.
      - **FileName:** pełna ścieżka do pliku, w którym będą przechowywane dzienniki.
- **UserStream:** jeśli włączone, klient będzie otrzymywał powiadomienia o aktualizacjach konta, zleceń lub salda (domyślnie true).
- **ListenKeyOnDisconnect:** ta właściwość określa, co zrobić w przypadku rozłączenia klienta z serwerami Binance przy aktywnym ListenKey.
  - **blkodDeleteListenKey:** Usuwa aktywny ListenKey poprzez żądanie HTTP do serwerów Binance (wartość domyślna).
  - **blkodClearListenKey:** Nie usuwa ListenKey z serwerów Binance i jedynie czyści wartość pola.
  - **blkodDoNothing:** nie wykonuje żadnych działań, więc przy następnym połączeniu z Binance zostanie podjęta próba użycia tego samego ListenKey.
- **UseCombinedStreams:** jeśli włączone, strumienie będą łączone w następujący sposób:  
{ "stream": "<streamName>", "data": "<rawPayload>" } (domyślnie wyłączone)

Klient może połączyć się z kontraktami terminowymi Binance **USDT** lub **COIN**. Aby wybrać typ kontraktu do handlu, należy ustawić właściwość **FuturesContracts**:

- **bfcUSDT:** łączy się z API kontraktów futures USD-M.
- **bfcCOIN:** łączy się z COIN-M Futures API.

Klient może łączyć się z kontami Binance produkcyjnymi lub demonstracyjnymi. Jeśli właściwość **Test-Net** jest włączona, połączenie nastąpi z kontem demonstracyjnym; w przeciwnym razie klient połączy się z serwerami produkcyjnymi Binance.

## WebSocket Stream API

Klient może subskrybować zdarzenia i anulować subskrypcję po nawiązaniu połączenia.

Obsługiwane są następujące metody subskrypcji i anulowania subskrypcji.

|                      |                      |                                                                                                                                                                                                                                                                    |
|----------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AggregateTrades      | Symbol               | Agregowane strumienie transakcji przesyłają informacje o transakcjach agregowane dla jednego zlecenia taker co 100 milisekund.                                                                                                                                     |
| MarkPrice            | Symbol, Update-Speed | Cena mark i stopa finansowania dla pojedynczego symbolu, przesyłane co 3 sekundy lub co sekundę.                                                                                                                                                                   |
| AllMarkPrice         | UpdateSpeed          | Cena markowa i stopa finansowania dla wszystkich symboli wysyłana co 3 sekundy lub co sekundę.                                                                                                                                                                     |
| KLine                | Symbol, interwał     | Strumień Kline/Candlestick przesyła aktualizacje bieżących danych świecowych co 250 milisekund (jeśli istnieją).                                                                                                                                                   |
| MiniTicker           | Symbol               | 24-godzinne kroczące statystyki mini-tickera dla pojedynczego symbolu. Nie są to statystyki dnia UTC, lecz 24-godzinne okno kroczące od czasu żądania do 24 godzin wcześniej.                                                                                      |
| AllMiniTicker        |                      | 24-godzinne kroczące statystyki mini-tickera dla wszystkich symboli. Nie są to statystyki dnia UTC, lecz 24-godzinne okno kroczące od czasu requestTime do 24 godzin wcześniej. Należy pamiętać, że w tablicy będą obecne wyłącznie tickery, które uległy zmianie. |
| Ticker               | Symbol               | Statystyki tickera z 24-godzinnego okna kroczącego dla pojedynczego symbolu. Nie są to statystyki dnia UTC, lecz 24-godzinne okno kroczące od requestTime do 24 godzin wstecz.                                                                                     |
| AllMarketTickers     |                      | 24-godzinne kroczące statystyki tickerów dla wszystkich symboli. Nie są to statystyki dnia UTC, lecz 24-godzinne okno kroczące od requestTime do 24 godzin wstecz. Należy pamiętać, że w tablicy znajdują się wyłącznie tickery, których wartości uległy zmianie.  |
| BookTicker           | Symbol               | Przekazuje w czasie rzeczywistym aktualizacje najlepszej ceny lub wolumenu po stronie bid lub ask dla wskazanego symbolu.                                                                                                                                          |
| AllBookTickers       |                      | Przesyła w czasie rzeczywistym wszelkie aktualizacje ceny lub ilości najlepszej oferty kupna lub sprzedaży dla wszystkich symboli.                                                                                                                                 |
| LiquidationOrders    | Symbol               | Strumienie zleceń likwidacyjnych przesyłają informacje o wymuszonych zleceniach likwidacyjnych dla określonego symbolu                                                                                                                                             |
| AllLiquidationOrders |                      | Strumienie All Liquidation Order przesyłają informacje o zleceniach przymusowej likwidacji dla wszystkich symboli na rynku.                                                                                                                                        |
| PartialBookDepth     | Symbol, Głębokość    | Najlepsze oferty kupna i sprzedaży; prawidłowe wartości to 5, 10 lub 20.                                                                                                                                                                                           |
| DiffDepth            | Symbol               | Oferty kupna i sprzedaży, przesyłane co 250 milisekund, 500 milisekund, 100 milisekund lub w czasie rzeczywistym (jeśli są dostępne)                                                                                                                               |

Po pomyślnej subskrypcji lub rezygnacji z subskrypcji klient otrzymuje o tym wiadomość, w której id jest wynikiem metody Subscribed lub Unsubscribed.

```
{
  "result": null,
  "id": 1
}
```

## User Data Stream API

Wymaga prawidłowego ApiKey uzyskanego z konta Binance; ApiKey musi być ustawiony we właściwości Binance.ApiKey komponentu.

Następujące dane są wypychane do klienta za każdym razem, gdy nastąpi zmiana. Nie jest konieczna subskrypcja żadnej metody — odbywa się ona automatycznie po ustawieniu prawidłowego ApiKey.

|                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Wezwanie do uzupełnienia depozytu zabezpieczającego | <p>Gdy wskaźnik ryzyka pozycji użytkownika jest zbyt wysoki, ten strumień zostanie wypchnięty. Ta wiadomość służy wyłącznie jako informacja o ryzyku i nie jest zalecana jako podstawa strategii inwestycyjnych. Na wysoce zmiennym rynku istnieje możliwość, że pozycja użytkownika zostanie zlikwidowana w tym samym czasie, gdy zostanie wypchnięty ten strumień.</p> <p>Aktualizacja salda następuje w następujących przypadkach:</p> |
| Aktualizacja salda i pozycji                        | <ul style="list-style-type: none"><li>• Zdarzenie jest wysyłane po aktualizacji salda lub pozycji.</li><li>• Gdy "FUNDING FEE" zmienia saldo użytkownika.</li><li>•</li></ul>                                                                                                                                                                                                                                                             |
| Aktualizacja zlecenia                               | Gdy zostanie utworzone nowe zlecenie, zostanie wypchnięte zdarzenie zmiany statusu zlecenia.                                                                                                                                                                                                                                                                                                                                              |

## REST API

Wszystkie punkty końcowe zwracają obiekt lub tablicę JSON. Dane są zwracane w kolejności rosnącej: najstarsze pierwsze, najnowsze ostatnie.

## Publiczne punkty końcowe API

Te punkty końcowe są dostępne bez żadnej autoryzacji.

### Ogólne EndPoints

|                        |                                                               |
|------------------------|---------------------------------------------------------------|
| Ping                   | Testuje łączność z Rest API.                                  |
| GetServerTime          | Sprawdź połączenie z Rest API i pobierz bieżący czas serwera. |
| GetExchangeInformation | Bieżące reguły handlowe giełdy i informacje o symbolach       |

## Punkty końcowe danych rynkowych

|                           |                              |                                                                                                                                                                     |
|---------------------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetOrderBook              | Symbol                       | Pobierz arkusz zleceń.                                                                                                                                              |
| GetTrades                 | Symbol                       | Pobierz ostatnie transakcje                                                                                                                                         |
| GetHistoricalTrades       | Symbol                       | Pobierz starsze transakcje.                                                                                                                                         |
| GetAggregateTrades        | Symbol                       | Pobierz skompresowane, zagregowane transakcje. Transakcje realizowane w tym samym czasie, z tego samego zlecenia i po tej samej cenie będą miały zagregowaną ilość. |
| GetKLines                 | Symbol, interwał             | Świece Kline dla instrumentu. Świece są jednoznacznie identyfikowane przez czas otwarcia.                                                                           |
| Get24hrTicker             | Symbol                       | Statystyki zmian cen w 24-godzinym oknie kroczącym. Należy zachować ostrożność podczas wywoływania bez podania symbolu.                                             |
| GetPriceTicker            | Symbol                       | Ostatnia cena dla symbolu lub symboli.                                                                                                                              |
| GetBookTicker             | Symbol                       | Najlepsza cena/ilość w arkuszu zleceń dla jednego lub wielu symboli.                                                                                                |
| GetMarkPrice              | Symbol                       | Cena Mark i Stopa Finansowania                                                                                                                                      |
| GetFundingRateHistory     | Symbol                       |                                                                                                                                                                     |
| GetOpenInterest           | Symbol                       | Pobierz bieżące otwarte zainteresowanie dla określonego symbolu.                                                                                                    |
| GetOpenInterestStatistics | Symbol, Period               |                                                                                                                                                                     |
| GetTopTraderAccountRatio  | Symbol, Period               |                                                                                                                                                                     |
| GetTopTraderPositionRatio | Symbol, Period               |                                                                                                                                                                     |
| GetGlobalAccountRatio     | Symbol, Period               |                                                                                                                                                                     |
| GetTakerVolume            | Symbol, Period               |                                                                                                                                                                     |
| GetContinuousKLines       | Pair, ContractType, Interval | Dane świecowe dla określonego typu kontraktu.                                                                                                                       |
| GetIndexPriceKLines       | Pair, Interval               | Świece Kline/candlestick dla ceny indeksu pary.                                                                                                                     |
| GetMarkPriceKLines        | Symbol, interwał             | Świece Kline dla ceny mark danego symbolu.                                                                                                                          |
| GetPremiumIndexKLines     | Symbol, interwał             | Słupki kline indeksu premii dla symbolu.                                                                                                                            |
| GetFundingInfo            |                              | Pobierz informacje o stawce finansowania dla wszystkich symboli.                                                                                                    |
| GetPriceTickerView2       | Symbol                       | Ostatnia cena dla jednego lub wielu symboli (V2).                                                                                                                   |
| GetIndexInfo              | Symbol                       | Pobierz informacje o indeksie.                                                                                                                                      |
| GetAssetIndex             | Symbol                       | Pobierz indeks aktywów dla trybu wieloaktywowego.                                                                                                                   |

|                  |                            |                             |
|------------------|----------------------------|-----------------------------|
| GetConstituents  | Symbol                     | Pobierz składowe indeksu.   |
| GetDeliveryPrice | Para                       | Pobierz cenę dostawy.       |
| GetBasis         | Pair, ContractType, Period | Pobieranie danych bazowych. |

## Private API EndPoints

Wymaga podania APIKey i APISecret w celu autoryzacji przez serwer.

### Punkty końcowe konta i transakcji

|                              |                                                         |                                                                                                        |
|------------------------------|---------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| ChangePositionMode           | DualPosition                                            | Zmień tryb pozycji użytkownika (tryb Hedge lub tryb jednokierunkowy) dla KAŻDEGO symbolu               |
| GetCurrentPosition-Mode      |                                                         | Pobierz tryb pozycji użytkownika (tryb zabezpieczony lub jednokierunkowy) dla KAŻDEGO symbolu          |
| NewOrder                     | Symbol, Side, PositionSide, Type                        | Wyślij nowe zlecenie.                                                                                  |
| PlaceMarketOrder             | Side, Symbol, Quantity                                  |                                                                                                        |
| PlaceLimitOrder              | Side, Symbol, Quantity, LimitPrice                      |                                                                                                        |
| PlaceStopOrder               | Side, Symbol, Quantity, StopPrice, LimitPrice           |                                                                                                        |
| PlaceTrailingStopOrder       | Side, Symbol, Quantity, aActivationPrice, aCallbackRate |                                                                                                        |
| QueryOrder                   | Symbol                                                  | Sprawdź status zlecenia.                                                                               |
| CancelOrder                  | Symbol                                                  | Anuluj aktywne zlecenie. Należy podać OrderId lub OrigClientOrderId.                                   |
| CancelAllOpenOrders          | Symbol                                                  |                                                                                                        |
| AutoCancelAllOpenOrders      | Symbol, CountdownTimer                                  | Anuluj wszystkie otwarte zlecenia określonego symbolu na koniec określonego odliczania.                |
| QueryCurrentOpenOrder        | Symbol                                                  |                                                                                                        |
| GetOpenOrders                | Symbol                                                  | Pobierz wszystkie otwarte zlecenia dla symbolu. Zachować ostrożność przy dostępie bez podania symbolu. |
| GetAllOrders                 | Symbol                                                  | Pobierz wszystkie zlecenia konta: aktywne, anulowane lub zrealizowane.                                 |
| GetAccountBalance            |                                                         |                                                                                                        |
| GetAccountInformation        |                                                         | Pobierz informacje o bieżącym koncie.                                                                  |
| ChangeInitialLeverage        | Symbol, Leverage                                        | Zmień początkową dźwignię użytkownika dla określonego rynku symbolu.                                   |
| ChangeMarginType             | Symbol, MarginType                                      |                                                                                                        |
| ModifyIsolatedPositionMargin | Symbol, Amount, Type                                    |                                                                                                        |



|                                |                                  |                                                                                                                                                  |
|--------------------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| GetPositionMarginChangeHistory | Symbol                           |                                                                                                                                                  |
| GetPositionInformation         | Symbol                           |                                                                                                                                                  |
| GetAccountTradeList            | Symbol                           |                                                                                                                                                  |
| GetIncomeHistory               | Symbol                           |                                                                                                                                                  |
| GetNotionalLeverageBracket     | Symbol                           |                                                                                                                                                  |
| TestNewOrder                   | Symbol, Side, PositionSide, Type | Testuje tworzenie nowego zlecenia oraz poprawność podpisu/recvWindow. Tworzy i weryfikuje nowe zlecenie, ale nie wysyła go do silnika dopasowań. |
| ModifyOrder                    | Symbol                           | Modyfikowanie istniejącego zlecenia.                                                                                                             |
| NewBatchOrders                 | BatchOrders                      | Złóż wiele zleceń jednocześnie.                                                                                                                  |
| ModifyBatchOrders              | BatchOrders                      | Modyfikuj wiele zleceń jednocześnie.                                                                                                             |
| CancelBatchOrders              | Symbol                           | Anuluje wiele zleceń.                                                                                                                            |
| GetOrderAmendment              | Symbol                           | Pobiera historię modyfikacji zlecenia.                                                                                                           |
| CountdownCancelAll             | Symbol, CountdownTime            | Anuluj wszystkie otwarte zlecenia określonego symbolu na koniec określonego odliczania.                                                          |
| GetForceOrders                 | Symbol                           | Pobierz zlecenia przymusowej likwidacji użytkownika.                                                                                             |
| GetADLQuantile                 | Symbol                           | Pobiera kwantylowe szacowanie ADL dla pozycji.                                                                                                   |
| GetAccountBalanceV3            |                                  | Pobierz saldo konta futures (V3).                                                                                                                |
| GetAccountInformationV3        |                                  | Pobierz bieżące informacje o koncie (V3).                                                                                                        |
| GetPositionInformationV3       | Symbol                           | Pobierz bieżące informacje o pozycji (V3).                                                                                                       |
| GetCommissionRate              | Symbol                           | Pobierz stawkę prowizji użytkownika.                                                                                                             |
| GetAccountConfig               |                                  | Pobierz bieżącą konfigurację konta.                                                                                                              |
| GetSymbolConfig                | Symbol                           | Pobierz konfigurację symbolu.                                                                                                                    |
| GetOrderRateLimit              |                                  | Pobierz limit szybkości zleceń użytkownika.                                                                                                      |
| GetApiTradingStatus            | Symbol                           | Pobierz wskaźniki reguł ilościowych dotyczących handlu przez API.                                                                                |
| ChangeMultiAssetsMode          | MultiAssetsMargin                | Zmiana trybu wielu aktywów użytkownika. Tryb wielu aktywów: true; Tryb jednego aktywa: false.                                                    |
| GetMultiAssetsMode             |                                  | Pobiera bieżący tryb wielu aktywów użytkownika.                                                                                                  |
| SetFeeBurn                     | FeeBurn                          | Zmień status spalania opłat BNB użytkownika.                                                                                                     |
| GetFeeBurn                     |                                  | Pobierz status spalania opłat BNB użytkownika.                                                                                                   |
| CreateListenKey                |                                  | Uruchom nowy strumień danych użytkownika. Strumień zostanie zamknięty po 60 minutach, jeśli nie zostanie wysłany sygnał keepalive.               |
| KeepAliveListenKey             |                                  | Utrzymuje aktywność strumienia danych użytkownika, zapobiegając przekroczeniu limitu czasu.                                                      |
| CloseListenKey                 |                                  | Zamknij strumień danych użytkownika.                                                                                                             |

## Zdarzenia

Wiadomości Binance Futures są odbierane w komponencie TsgcWebSocketClient; można używać następujących zdarzeń:

### **OnConnect**

Po pomyślnym połączeniu z serwerem Binance.

### **OnDisconnect**

Po rozłączeniu z serwerem Binance

### **OnMessage**

Wiadomości wysyłane przez serwer do klienta są obsługiwane w tym zdarzeniu.

### **OnError**

W przypadku wystąpienia błędu protokołu wywoływane jest to zdarzenie.

### **OnException**

Jeżeli wystąpi nieobsługiwany wyjątek, zostanie wywołane to zdarzenie.

Dodatkowo w komponencie Binance API istnieje specyficzne zdarzenie o nazwie **OnBinanceHTTPException**, wywoływane za każdym razem, gdy wystąpi błąd podczas wywołania żądania HTTP (REST API lub strumień użytkownika WebSocket).

**(\*) W związku ze zmianami na serwerach Binance wersje Indy wcześniejsze niż Rad Studio 10.1 nie będą mogły łączyć się z serwerami testowymi. Problem ten nie dotyczy wersji Enterprise ani zaktualizowanych wersji Indy.**

# API Binance Futures | Handel

Binance umożliwia handel kontraktami futures za pośrednictwem swojego REST API.

## Konfiguracja

Najpierw należy utworzyć **klucz API** na koncie Binance i dodać uprawnienia do handlu kontraktami Futures.

Po wykonaniu tych czynności można rozpocząć handel kontraktami futures.

Najpierw należy wybrać, czy transakcje mają być realizowane za pomocą kontraktów futures **USDT** czy **COIN**. Dostępna jest właściwość `FuturesContracts`, w której można określić rodzaj kontraktu futures.

Następnie **ustaw klucz ApiKey oraz tajny klucz ApiSecret** w komponencie klienta Binance Futures; zostaną one użyte do podpisywania żądań wysyłanych do serwera Binance.

## Złóż zlecenie

Aby złożyć nowe zlecenie, należy wywołać metodę **REST\_API.NewOrder** komponentu klienta Binance Futures.

W zależności od typu zlecenia (rynkowe, limit...) API wymaga mniejszej lub większej liczby pól.

## Pola obowiązkowe

- **Symbol:** symbol identyfikatora produktu, przykład: BTCUSD\_210326
- **Side:** BUY lub SELL
- **type:** typ zlecenia
  - LIMIT
  - MARKET
  - STOP
  - TAKE\_PROFIT
  - STOP\_MARKET
  - TAKE\_PROFIT\_MARKET
  - TRAILING\_STOP\_MARKET

## **Dodatkowe obowiązkowe pola w zależności od typu**

- **LIMIT:** timeInForce, quantity, price
- **MARKET:** ilość
- **STOP/TAKE\_PROFIT:** ilość, cena, cena stop
- **STOP\_MARKET/TAKE\_PROFIT\_MARKET:** stopPrice
- **TRAILING\_STOP\_MARKET:** callbackRate

Podczas składania zlecenia możliwe są 2 scenariusze:

1. **Powodzenie:** funkcja NewOrder zwraca wiadomość wysłaną przez serwer Binance.
2. **Błąd:** wyjątek jest zwracany w zdarzeniu OnBinanceHTTPException.

# API SocketIO

## SocketIO

Socket.IO to biblioteka JavaScript służąca do tworzenia aplikacji internetowych działających w czasie rzeczywistym. Umożliwia dwukierunkową komunikację między klientami webowymi a serwerami w czasie rzeczywistym. Składa się z dwóch części: biblioteki po stronie klienta działającej w przeglądarce oraz biblioteki po stronie serwera dla Node.js. Oba komponenty mają niemal identyczne API. Podobnie jak Node.js, jest sterowany zdarzeniami.

## Typy wiadomości

**0:** otwarte (Wysyłane przez serwer po otwarciu nowego transportu (ponowne sprawdzenie))

**1:** zamknięcie (żądanie zamknięcia tego transportu, bez zamykania samego połączenia).

**2:** ping (wysyłany przez klienta; serwer powinien odpowiedzieć pakietem pong zawierającym te same dane)

przykład

klient wysyła: 2probe

server sends: 3probe

**3:** pong (wysyłany przez serwer w odpowiedzi na pakiety ping.)

**4:** wiadomość tekstowa (właściwa wiadomość; klient i serwer powinny wywołać swoje wywołania zwrotne z danymi).

przykład:

```
42/chat,["join","{room:1}"]
```

4 to typ pakietu wiadomości w protokole engine.io

2 to typ EVENT w protokole socket.io

/chat to dane przetwarzane przez socket.io

socket.io wyzwoli zdarzenie "join"

przekazuje dane "room: 1". Przestrzeń nazw można pominąć tylko wtedy, gdy wynosi /.

**5:** uaktualnienie (Przed przełączeniem transportu przez engine.io sprawdza on, czy serwer i klient mogą się ze sobą komunikować za pośrednictwem tego transportu. Jeśli test zakończy się sukcesem, klient wysyła pakiet uaktualnienia, żądając od serwera opróżnienia pamięci podręcznej na starym transporcie i przełączenia na nowy.)

**6:** noop (Pakiet noop. Używany przede wszystkim do wymuszania cyklu odpytywania, gdy odebrane zostanie przychodzące połączenie WebSocket).

## Właściwości

**API:** określa wersję SocketIO:

**ioAPI0:** obsługuje serwery socket.io 0.\* (wybrane domyślnie)

**ioAPI1:** obsługuje serwery socket.io w wersji 1.\*

**ioAPI2:** obsługuje serwery socket.io 2.\*

**ioAPI3:** obsługuje serwery socket.io 3.\*

**ioAPI4:** obsługuje serwery socket.io w wersji 4.\*

**Base64:** jeśli włączony, wiadomości binarne są odbierane w formacie base64.

**HandShakeCustomURL:** umożliwia dostosowanie adresu URL w celu pobrania sesji socket.io.

**HandShakeTimestamp:** należy włączyć tylko wtedy, gdy chcesz wysyłać znacznik czasu jako parametr przy żądaniu nowej sesji (włącz tę właściwość, jeśli próbujesz uzyskać dostęp do serwera Python gevent-socketio).

**HandShakeAuthToken:** jeśli serwer wymaga tokena do uwierzytelnienia, należy tu ustawić token uwierzytelniający.

**Namespace:** umożliwia ustawienie przestrzeni nazw podczas łączenia z serwerem.

**Polling:** wyłączenie tej właściwości powoduje, że klient łączy się bezpośrednio z serwerem przy użyciu WebSocket jako transportu.

**Parametry:** umożliwia ustawienie parametrów połączenia.

**EncodeParameters:** jeżeli jest włączone, parametry są kodowane.

## Metody

Aby wysyłać wiadomości do serwera socket.io, należy użyć metody `WriteData` (zgodnie z sekcją Typy wiadomości).

1. Wywołaj metodę „add user” z jednym parametrem, używając wartości John jako nazwy użytkownika.

```
writeData("42[\"add user\", \"John\"]");
```

## Zdarzenia

### OnHTTPRequest

Przed nawiązaniem nowego połączenia WebSocket serwer socket.io wymaga, aby klient otworzył nowe połączenie HTTP w celu uzyskania nowego identyfikatora sesji. W niektórych przypadkach serwer socket.io wymaga uwierzytelnienia za pomocą nagłówków HTTP. Za pomocą tego zdarzenia można dodawać niestandardowe nagłówki HTTP, takie jak uwierzytelnianie Basic lub token Bearer.

### OnAfterConnect

To zdarzenie jest wywoływane po pomyślnym nawiązaniu połączenia socket.io i umożliwia klientowi wysyłanie wiadomości do serwera. Tutaj można na przykład subskrybować przestrzeń nazw.

### OnHTTPConnectionSSL

Gdy serwer WebSocket wymaga bezpiecznych połączeń, przy próbie połączenia klienta z serwerem może pojawić się następujący komunikat o błędzie:

**Error connecting with SSL. error:XXXXXXXX:SSL routines:ssl3\_read\_bytes:tlsv1 alert protocol version**

Błąd ten oznacza, że klient próbuje nawiązać połączenie przy użyciu wersji TLS, która nie jest obsługiwana przez serwer.

Aby rozwiązać ten błąd, należy obsłużyć OnSSLAfterCreateHandler komponentu klienta WebSocket i ustawić nowszą wersję TLS.

Na przykład: tutaj ustawiamy TLS 1.2 jako wersję protokołu.

```
void OnHTTPConnectionSSL(TObject *Sender; TIdSSLIOHandlerSocketBase *aSSLHandler)
{
    static_cast<tidsslhandlersocketopenssl*>(aSSLHandler)->SSLOptions->Method = sslvTLSv1_2;
}
```



# API Coinbase

Coinbase

## Obsługiwane API

- [WebSockets API](#): połączenie z publicznym serwerem WebSocket zapewniające aktualizacje danych rynkowych w czasie rzeczywistym.
- [REST API](#): REST API oferuje punkty końcowe do zarządzania kontem i zleceniami oraz dostęp do publicznych danych rynkowych.

## Najczęstsze zastosowania

- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować kanał WebSocket](#)
- **REST API**
  - [Jak uzyskać dane rynkowe](#)
  - [Jak korzystać z prywatnego REST API](#)
  - [Jak składać zlecenia](#)
  - [Jak korzystać z konta SandBox](#)
  - [Czas prywatnych żądań](#)

## WebSockets API

Kanał WebSocket jest publicznie dostępny i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym dla zleceń i transakcji. W środowisku produkcyjnym obsługiwane są dwa punkty końcowe:

- **Dane rynkowe** to feed dostarczający aktualizacje dla zleceń i transakcji. Większość kanałów jest teraz dostępna bez uwierzytelniania.
- **Dane zamówień użytkownika** dostarczają aktualizacji dotyczących zamówień użytkownika.

Można subskrybować następujące kanały:

|                           |                                            |                                                                                    |
|---------------------------|--------------------------------------------|------------------------------------------------------------------------------------|
| <b>SubscribeHeartBeat</b> |                                            | Ping serwera w czasie rzeczywistym w celu utrzymania wszystkich połączeń aktywnych |
| <b>SubscribeStatus</b>    | <b>aProductId</b> : identyfikator produktu | Wysyła wszystkie produkty i waluty w ustalonym interwale                           |

|                                       |                                           |                                                                                               |
|---------------------------------------|-------------------------------------------|-----------------------------------------------------------------------------------------------|
| <b>SubscribeCandles</b>               | <b>aProductId:</b> identyfikator produktu | Aktualizacje świec produktu w czasie rzeczywistym                                             |
| <b>SubscribeTicker</b>                | <b>aProductId:</b> identyfikator produktu | Aktualizacje cen w czasie rzeczywistym po każdym dopasowaniu transakcji                       |
| <b>SubscribeTickerBatch</b>           | <b>aProductId:</b> identyfikator produktu | Aktualizacje cen w czasie rzeczywistym co 5000 milisekund                                     |
| <b>SubscribeLevel2</b>                | <b>aProductId:</b> identyfikator produktu | Wszystkie aktualizacje i najłatwiejszy sposób na utrzymanie migawki księgi zleceń             |
| <b>SubscribeMarket-Trades</b>         | <b>aProductId:</b> identyfikator produktu | Aktualizacje w czasie rzeczywistym przy każdej transakcji rynkowej                            |
| <b>SubscribeUser</b>                  | <b>aProductId:</b> identyfikator produktu | Wysyła tylko wiadomości zawierające uwierzytelnionego użytkownika                             |
| <b>SubscribeFuturesBalanceSummary</b> |                                           | Aktualizacje w czasie rzeczywistym za każdym razem, gdy zmienia się saldo futures użytkownika |

Kanały User i FuturesBalanceSummary wymagają uwierzytelnienia; należy najpierw uzyskać klucze API na koncie Coinbase, a następnie ustawić wartości we właściwości Coinbase komponentu:

- ApiKey
- ApiSecret

Uwierzytelnienie przyniesie kilka korzyści:

1. Wiadomości, w których jesteś jedną ze stron, są rozwinięte i zawierają więcej użytecznych pól
2. Będziesz otrzymywać prywatne wiadomości, takie jak informacje o cyklu życia składanych zleceń stop

## REST API

### Prywatne punkty końcowe

Prywatne punkty końcowe są dostępne do zarządzania zleceniami i zarządzania kontem.

Przed możliwością podpisywania żądań należy utworzyć klucz API za pośrednictwem strony internetowej Coinbase Pro. Klucz API będzie powiązany z konkretnym profilem. Po jego utworzeniu użytkownik otrzymuje 3 informacje, które należy zapamiętać:

- Klucz
- Sekret
- Hasło dostępu

Klucz i Secret zostaną losowo wygenerowane i udostępnione przez Coinbase Pro; Passphrase zostanie podane przez użytkownika w celu dodatkowego zabezpieczenia dostępu do API. Coinbase Pro przechowuje solony skrót hasła w celu weryfikacji, ale nie może go odzyskać, jeśli zostanie zapomniane.

Można ograniczyć funkcjonalność kluczy API. Przed utworzeniem klucza należy wybrać, jakie uprawnienia ma on posiadać. Dostępne uprawnienia to:

- View — nadaje kluczowi uprawnienia do odczytu. Obejmuje wszystkie punkty końcowe GET.
- Transfer — umożliwia kluczowi API transfer waluty w imieniu konta, w tym wpłaty i wypłaty. Włączaj ostrożnie — transfery za pomocą klucza API POMIJAJĄ uwierzytelnianie dwuskładnikowe.
- Trade — umożliwia składanie zleceń oraz pobieranie danych handlowych. Obejmuje żądania POST /orders i kilka punktów końcowych GET.

## Konta

|                     |                                        |                                                                                                                                            |
|---------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ListAccounts</b> |                                        | Pobiera listę kont handlowych z profilu klucza API.                                                                                        |
| <b>GetAccount</b>   | <b>aAccountId:</b> identyfikator konta | Informacje o jednym koncie. Użyj tego punktu końcowego, gdy znany jest account_id. Klucz API musi należeć do tego samego profilu co konto. |

## Zamówienia

|                         |                                                                                                    |                                                                                                             |
|-------------------------|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>PlaceNewOrder</b>    | <b>aOrder:</b> klasa zawierająca wszystkie możliwe pola zlecenia                                   | Składa nowe zlecenie. Należy używać tylko wtedy, gdy wymagany jest dostęp do zaawansowanych opcji zlecenia. |
|                         | <b>aSide:</b> kup lub sprzedaj                                                                     |                                                                                                             |
|                         | <b>aProductId:</b> identyfikator produktu                                                          |                                                                                                             |
|                         | <b>aQuoteSize:</b> Kwota drugiego składnika aktywów w parze handlowej.                             |                                                                                                             |
| <b>PlaceMarketOrder</b> | <b>aBaseSize:</b> Ilość pierwszego składnika aktywów w parze handlowej                             | Składa nowe zlecenie rynkowe.                                                                               |
|                         | <b>aClient_oid:</b> Identyfikator zlecenia wybrany przez użytkownika w celu identyfikacji zlecenia |                                                                                                             |
| <b>PlaceLimitOrder</b>  | <b>aSide:</b> kup lub sprzedaj                                                                     | Składa nowe zlecenie z limitem.                                                                             |

|                         |                                                                                                   |                                                                                                             |
|-------------------------|---------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
|                         | <b>aProductId:</b> identyfikator produktu                                                         |                                                                                                             |
|                         | <b>aQuoteSize:</b> Ilość drugiego aktywa w parze handlowej.                                       |                                                                                                             |
|                         | <b>aBaseSize:</b> Ilość pierwszego aktywa w parze handlowej                                       |                                                                                                             |
|                         | <b>aLimitPrice:</b> limit cenowy                                                                  |                                                                                                             |
|                         | <b>Client_oid:</b> Identyfikator zlecenia wybrany przez użytkownika w celu identyfikacji zlecenia |                                                                                                             |
|                         | <b>aSide:</b> kup lub sprzedaj                                                                    |                                                                                                             |
|                         | <b>ProductId:</b> identyfikator produktu                                                          |                                                                                                             |
|                         | <b>aBaseSize:</b> Ilość pierwszego aktywa w parze handlowej                                       |                                                                                                             |
|                         | <b>StopPrice:</b> cena stop                                                                       | Składa nowe zlecenie stop                                                                                   |
|                         | <b>aLimitPrice:</b> limit ceny                                                                    |                                                                                                             |
| <b>PlaceStopOrder</b>   | <b>aStopDirection:</b> strata lub wejście                                                         |                                                                                                             |
|                         | <b>Client_oid:</b> Identyfikator zlecenia wybrany przez użytkownika w celu identyfikacji zlecenia |                                                                                                             |
| <b>CancelOrder</b>      | <b>aOrderId:</b> identyfikator zlecenia                                                           | Anuluje wcześniej złożone zlecenie. Zlecenie musi należeć do profilu, do którego przypisany jest klucz API. |
|                         | <b>aOrderId:</b> identyfikator zlecenia                                                           |                                                                                                             |
| <b>EditOrder</b>        | <b>aPrice:</b> cena                                                                               | Edytuj zlecenie z określoną nową wielkością lub nową ceną                                                   |
|                         | <b>aSize:</b> Ilość                                                                               |                                                                                                             |
| <b>EditOrderPreview</b> | <b>aOrderId:</b> identyfikator zlecenia                                                           |                                                                                                             |
|                         | <b>aPrice:</b> cena                                                                               | Podgląd żądania edycji zlecenia z nową określoną wielkością lub ceną.                                       |
|                         | <b>aSize:</b> Ilość                                                                               |                                                                                                             |
| <b>ListOrders</b>       |                                                                                                   | Pobiera listę zleceń filtrowaną według opcjonalnych parametrów zapytania (product_id, order_status itp.).   |
| <b>GetOrder</b>         | <b>aOrderId:</b> identyfikator zlecenia                                                           | Pobierz pojedyncze zlecenie na podstawie jego identyfikatora.                                               |
| <b>PreviewOrder</b>     |                                                                                                   | Podgląd zlecenia.                                                                                           |
| <b>ClosePosition</b>    | <b>aOrderId:</b> identyfikator zlecenia                                                           | Składa zlecenie zamknięcia otwartych pozycji dla określonego product_id.                                    |

**aProductId:** identyfikator produktu

**aSize:** ilość

## Dane rynkowe

|                                |                                                                                                                                                                                                             |                                                                                                                                    |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetPublicProducts</b>       |                                                                                                                                                                                                             | Pobierz listę dostępnych par walutowych do handlu.                                                                                 |
| <b>GetPublicProduct</b>        | <b>aProductId:</b> identyfikator produktu                                                                                                                                                                   | Pobierz informacje o pojedynczym produkcie według identyfikatora produktu.                                                         |
| <b>GetPublicProductBook</b>    | <b>aProductId:</b> identyfikator produktu                                                                                                                                                                   | Pobierz listę ofert kupna/sprzedaży dla pojedynczego produktu. Poziom szczegółowości można dostosować za pomocą parametru limit.   |
| <b>GetPublicProductCandles</b> | <b>aProductId:</b> identyfikator produktu<br><b>aStart:</b> początek przedziału czasu.<br><b>aEnd:</b> koniec przedziału czasu<br><b>aGranularity:</b> Przedział czasowy reprezentowany przez każdą świecę. | Pobierz kursy dla pojedynczego produktu według jego identyfikatora, pogrupowane w przedziały czasowe.                              |
| <b>GetTrades</b>               | <b>aProductId:</b> identyfikator produktu                                                                                                                                                                   | Pobierz informacje migawkowe według identyfikatora produktu o ostatnich transakcjach (tick) oraz najlepszej cenie kupna/sprzedaży. |
| <b>GetTime</b>                 |                                                                                                                                                                                                             | Pobierz bieżący czas z Coinbase Advanced API.                                                                                      |

## Realizacje

|                            |                                                                                   |
|----------------------------|-----------------------------------------------------------------------------------|
| <b>GetFillsByOrderId</b>   | Pobierz listę realizacji przefiltrowanych według identyfikatora zamówienia        |
| <b>GetFillsByProductId</b> | Pobierz listę realizacji transakcji przefiltrowaną według identyfikatora produktu |
| <b>GetFillsByTradeId</b>   | Pobierz listę realizacji przefiltrowaną według identyfikatora transakcji          |

## Konwertuj

|                           |                                             |
|---------------------------|---------------------------------------------|
| <b>CreateConvertQuote</b> | Tworzenie oferty konwersji między walutami. |
| <b>CommitConvertTrade</b> | Zatwierdź transakcję konwersji.             |
| <b>GetConvertTrade</b>    | Pobierz szczegóły transakcji konwersji.     |

## Opłaty

|                              |                                            |
|------------------------------|--------------------------------------------|
| <b>GetTransactionSummary</b> | Pobierz podsumowanie opłat transakcyjnych. |
|------------------------------|--------------------------------------------|

## Produkty (uwierzytelnione)

|                          |                                           |                                      |
|--------------------------|-------------------------------------------|--------------------------------------|
| <b>ListProducts</b>      |                                           | Wylistuj dostępne produkty.          |
| <b>GetProduct</b>        | <b>aProductId:</b> identyfikator produktu | Pobierz konkretny produkt.           |
| <b>GetProductBook</b>    | <b>aProductId:</b> identyfikator produktu | Pobiera arkusz zleceń produktu.      |
| <b>GetProductCandles</b> | <b>aProductId:</b> identyfikator produktu | Pobierz świece OHLCV dla produktu.   |
| <b>GetMarketTrades</b>   | <b>aProductId:</b> identyfikator produktu | Pobierz ostatnie transakcje rynkowe. |
| <b>GetBestBidAsk</b>     |                                           | Pobierz najlepsze ceny bid/ask.      |

## Portfele

|                              |                                          |
|------------------------------|------------------------------------------|
| <b>ListPortfolios</b>        | Wyświetl listę wszystkich portfeli.      |
| <b>CreatePortfolio</b>       | Utwórz nowe portfolio.                   |
| <b>DeletePortfolio</b>       | Usuń portfolio.                          |
| <b>GetPortfolioBreakdown</b> | Pobierz szczegóły podziału portfela.     |
| <b>MovePortfolioFunds</b>    | Przeniesienie środków między portfelami. |

## Wieczyste

|                                      |                                                            |
|--------------------------------------|------------------------------------------------------------|
| <b>GetPerpetualsPortfolioSummary</b> | Pobierz podsumowanie portfela kontraktów wieczystych.      |
| <b>ListPerpetualsPositions</b>       | Lista otwartych pozycji perpetual.                         |
| <b>GetPerpetualsPosition</b>         | Pobiera określoną pozycję na rynku kontraktów wieczystych. |

## Coinbase | Połączenie z API WebSocket

Aby połączyć się z Coinbase WebSocket API, wystarczy utworzyć nowego klienta API Coinbase i podłączyć go do TsgcWebSocketClient. Poniżej znajduje się przykład:

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase();
oCoinbase->Client = oClient;
oClient->Active = true;
```

# Coinbase | Subskrybuj kanał WebSocket

Coinbase oferuje różne kanały, do których można subskrybować, aby otrzymywać aktualizacje danych rynkowych w czasie rzeczywistym, informacje o zleceniach itd. Poniżej przykład subskrypcji kanału Ticker:

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase();
oCoinbase->Client = oClient;
oCoinbase->SubscribeTicker("ETH-USD");

void OnCoinbaseMessage(TObject *Sender, string aType, string aRawMessage)
{
    // here you will receive the ticker updates
}
```



## Coinbase Pro | Pobieranie danych rynkowych

Coinbase udostępnia publiczne dane rynkowe przez punkty końcowe REST. Wywołanie jednego z tych punktów końcowych zwraca migawkę żądanych danych rynkowych.

Punkty końcowe danych rynkowych nie wymagają uwierzytelniania i są ogólnodostępne dla wszystkich użytkowników.

**Przykład:** aby uzyskać migawkę księgi BTC-USD, należy wykonać następujące wywołanie

```
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase(this);  
ShowMessage(oCoinbase->REST_API->GetPublicProductBook("BTC-USD"));
```

# Coinbase Pro | Prywatne REST API

Coinbase REST API udostępnia publiczne i prywatne punkty końcowe. Prywatne punkty końcowe wymagają podpisywania wiadomości w celu zwiększenia bezpieczeństwa transakcji.

Najpierw należy zalogować się na konto Coinbase i utworzyć nowe API; uzyska się wtedy następujące wartości:

- ApiKey
- ApiSecret

Pola te należy skonfigurować we właściwości Coinbase komponentu klienta Coinbase API.

Po skonfigurowaniu można rozpocząć wysyłanie prywatnych żądań do REST API Coinbase

```
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase(this);  
oCoinbase->Coinbase->ApiKey = "<your api key>";  
oCoinbase->Coinbase->ApiSecret = "<your api secret>";  
ShowMessage(oCoinbase->REST_API->ListAccounts);
```

## Coinbase Pro | Czas żądań prywatnych

Podczas wykonywania prywatnego żądania do Coinbase wiadomość jest podpisywana w celu zwiększenia bezpieczeństwa żądań. Wiadomość zawiera lokalny czas i jest wysyłana w ramach podpisanej wiadomości. Jeśli różnica między czasem lokalnym a czasem serwerów Coinbase wynosi więcej niż 30 sekund, żądanie zostanie odrzucone. Dlatego ważne jest, aby sprawdzić, czy czas lokalny jest zsynchronizowany, co można wykonać za pomocą metody synchronizacji czasu dostępnej dla danego systemu operacyjnego.

Można sprawdzić czas serwera Coinbase Pro, wywołując metodę **GetTime**, która zwróci czas serwera Coinbase Pro

# Coinbase Pro | Składanie zleceń

Aby składać nowe zlecenia na Coinbase, wymagane są klucze API umożliwiające dostęp do prywatnych danych. Należy zapoznać się z następującym artykułem: [Jak korzystać z prywatnego REST API](#).

Po skonfigurowaniu kluczy API można rozpocząć składanie zleceń

## Zlecenie rynkowe

Złóż nowe zlecenie rynkowe, kup 0,002 kontraktów BTC-USD

```
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase(this);
oCoinbase->Coinbase->ApiKey = "your api key";
oCoinbase->Coinbase->ApiSecret = "your api secret";
oCoinbase->Coinbase->ApiPassphrase = "your passphrase";
ShowMessage(oCoinbase->REST_API>PlaceMarketOrder(coisBuy, "BTC-USD", 0.002, 0));
```

## Zlecenie z limitem ceny

Złóż nowe zlecenie Limit — kup 0,002 kontraktów BTC-USD po cenie limitowanej 10000

```
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase(this);
oCoinbase->Coinbase->ApiKey = "your api key";
oCoinbase->Coinbase->ApiSecret = "your api secret";
oCoinbase->Coinbase->ApiPassphrase = "your passphrase";
ShowMessage(oCoinbase->REST_API>PlaceLimitOrder(coisBuy, "BTC-USD", 0.002, 0, 10000));
```

## Konto piaskownicy Coinbase Pro

Coinbase umożliwia korzystanie z konta SandBox, na którym można handlować bez prawdziwych środków. Konto to wymaga utworzenia kluczy API innych niż dla konta produkcyjnego.

Aby korzystać z konta SandBox, wystarczy ustawić właściwość **Coinbase.SandBox** na **true** przed wysłaniem jakiegokolwiek żądania do REST API.

```
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase(this);  
oCoinbase->Coinbase->ApiKey = "your api key";  
oCoinbase->Coinbase->ApiSecret = "your api secret";  
oCoinbase->Coinbase->SandBox = true;  
ShowMessage(oCoinbase->REST_API>ListAccounts);
```

# API SignalRCore

## SignalRCore

ASP.NET Core SignalR to biblioteka open-source upraszczająca dodawanie funkcji czasu rzeczywistego do aplikacji internetowych. Funkcje czasu rzeczywistego umożliwiają serwerowi natychmiastowe przesyłanie treści do klientów.

Dobre kandydatury dla SignalR:

- Aplikacje wymagające częstych aktualizacji z serwera. Przykłady to aplikacje do gier, portale społecznościowe, głosowania, aukcje, map i GPS.
- Pulpity nawigacyjne i aplikacje monitorujące. Przykłady obejmują firmowe pulpity nawigacyjne, natychmiastowe aktualizacje sprzedaży lub alerty podrózne.
- Aplikacje do współpracy. Przykładami aplikacji do współpracy są aplikacje z tablicą whiteboard i oprogramowanie do spotkań zespołowych.
- Aplikacje wymagające powiadomień. Sieci społecznościowe, poczta e-mail, czat, gry, alerty podrózne i wiele innych aplikacji korzysta z powiadomień.

Komponent `sgcWebSockets` SignalRCore używa `WebSocket` jako transportu do połączenia z serwerem SignalRCore. Jeżeli ten transport nie jest obsługiwany, zostanie zgłoszony błąd.

## Koncentratory

SignalRCore używa hubów do komunikacji między klientami a serwerami. SignalRCore udostępnia 2 protokoły hubowe: protokół tekstowy oparty na JSON oraz protokół binarny oparty na `MessagePack`. Komponent `sgcWebSockets` implementuje wyłącznie protokół tekstowy JSON do komunikacji z serwerami SignalRCore.

Aby skonfigurować Hub, z którego będzie korzystał klient, należy ustawić nazwę Huba we właściwości **SignalRCore/Hub** przed nawiązaniem połączenia klienta z serwerem.

## Połączenie

Gdy klient otwiera nowe połączenie z serwerem, wysyła komunikat żądania zawierający protokół formatu i wersję. `sgcWebSockets` zawsze wysyła protokół formatu jako JSON. Serwer odpowie błędem, jeśli protokół nie jest przez niego obsługiwany; błąd ten można obsłużyć za pomocą zdarzenia **OnSignalRCoreError**. W przypadku nawiązania połączenia zostanie wywołane zdarzenie **OnSignalRCoreConnect**.

Gdy klient łączy się z serwerem SignalRCore, może wysłać `ConnectionId` identyfikujący klienta między sesjami, dzięki czemu po rozłączeniu klient może ponownie połączyć się z serwerem, podając ten sam wcześniejszy identyfikator połączenia. Aby uzyskać nowy identyfikator połączenia, należy połączyć się normalnie z serwerem; `ConnectionId` można poznać za pomocą zdarzenia **OnBeforeConnectEvent**. Aby ponownie połączyć się z serwerem i przekazać wcześniejszy identyfikator połączenia, należy użyć metody **ReConnect** i przekazać **ConnectionId** jako parametr.

## Protokół SignalRCore

Protokół SignalR to protokół dwukierunkowego RPC oparty na dowolnym transporcie obsługującym komunikaty. Obie strony połączenia mogą wywoływać procedury drugiej strony, a procedury mogą zwracać zero lub więcej wyników albo błąd. Przykład: klient może zażądać metody od serwera, a serwer może zażądać metody od klienta. Między serwerem a klientami wymieniane są następujące komunikaty:

- **HandshakeRequest:** wysyłane przez klienta do serwera w celu uzgodnienia formatu wiadomości.
- **HandshakeResponse:** serwer wysyła do klienta potwierdzenie poprzedniej wiadomości HandshakeRequest. Zawiera błąd, jeśli uzgadnianie połączenia nie powiodło się.
- **Close:** wywoływane przez klienta lub serwer po zamknięciu połączenia. Zawiera błąd, jeśli połączenie zostało zamknięte z powodu błędu.
- **Wywołanie:** klient lub serwer wysyła wiadomość do innego węzła w celu wywołania metody z argumentami lub bez.
- **StreamInvocation:** klient lub serwer wysyła wiadomość do innego uczestnika w celu wywołania metody strumieniowej z argumentami lub bez. Odpowiedź zostanie podzielona na różne elementy.
- **StreamItem:** jest odpowiedzią na poprzednie wywołanie StreamInvocation.
- **Completion:** oznacza zakończenie poprzedniego wywołania lub StreamInvocation. Może zawierać wynik, jeśli proces zakończył się pomyślnie, lub błąd w przypadku wystąpienia problemu.
- **CancelInvocation:** anuluj poprzednie żądanie StreamInvocation.
- **Ping:** wiadomość służąca do sprawdzenia, czy połączenie jest nadal aktywne.

## Kodowanie SignalRCore

SignalRCore umożliwia stosowanie następujących kodowań:

- **JSON:** aktualnie jedyne obsługiwane kodowanie.
- **MessagePack**

Aktualnie obsługiwany jest wyłącznie format JSON, chociaż MessagePack może być używany przez kodowanie wysyłanych wiadomości za pomocą zewnętrznej biblioteki messagepack. Więcej informacji znajdziesz w sekcji MessagePack poniżej.

Konfiguracja protokołu kodowania jest zdefiniowana we właściwości SignalRCore.Protocol. Domyślną wartością jest **srcpJSON**.

## Autoryzacja

Uwierzytelnianie można włączyć, aby powiązać użytkownika z każdym połączeniem i filtrować, którzy użytkownicy mogą uzyskiwać dostęp do zasobów. Uwierzytelnianie jest realizowane przy użyciu tokenów Bearer: klient dostarcza token dostępu, a serwer weryfikuje ten token i używa go do identyfikacji użytkownika.

W standardowych API sieci Web tokeny bearer są wysyłane w nagłówku HTTP, ale w przypadku WebSockets token jest przekazywany jako parametr ciągu zapytania.

Obsługiwane są następujące metody:

### **srcRequestToken**

Jeśli uwierzytelnianie jest włączone, przepływ wygląda następująco:

1. Najpierw próbuje uzyskać ważny token z serwera. Otwiera połączenie HTTP z `Authentication.RequestToken.URL` i wykonuje żądanie POST z danymi użytkownika i hasłem.
2. Jeśli poprzedni krok zakończy się sukcesem, zwracany jest token. W przeciwnym razie zwracany jest błąd.
3. Jeśli token zostanie zwrócony, otwierane jest nowe połączenie HTTP w celu negocjacji nowego połączenia. W tym miejscu token jest przekazywany jako nagłówek HTTP.
4. Jeśli poprzedni krok się powiedzie, otwierane jest połączenie WebSocket i token jest przekazywany jako parametr zapytania.

- **Authentication.Enabled:** jeśli aktywna, autoryzacja będzie stosowana przed nawiązaniem połączenia WebSocket.
- **Authentication.Username:** nazwa użytkownika przekazywana serwerowi w celu uwierzytelnienia.
- **Authentication.Password:** tajne hasło podawane serwerowi w celu uwierzytelnienia.
- **Authentication.RequestToken.PostFieldUsername:** nazwa pola do przekazania nazwy użytkownika (zależy od konfiguracji; należy sprawdzić stronę JavaScript HTTP, aby zobaczyć używaną nazwę).
- **Authentication.RequestToken.PostFieldPassword:** nazwa pola, w którym przesyłane jest hasło (zależy od konfiguracji; należy sprawdzić stronę HTTP JavaScript, aby zobaczyć, która nazwa jest używana).
- **Authentication.RequestToken.URL:** adres URL, pod którym żądany jest token.
- **Authentication.RequestToken.QueryFieldToken:** nazwa parametru ciągu zapytania używanego przy połączeniu WebSocket.

### **srcSetToken**

Token jest przekazywany bezpośrednio do serwera SignalRCore (ponieważ token został uzyskany z innego serwera).

- **Authentication.Enabled:** jeśli aktywne, autoryzacja będzie stosowana przed nawiązaniem połączenia WebSocket.



- **Authentication.SetToken.Token:** uzyskana wartość tokenu.

Token dostępu może być wysyłany jako parametr zapytania (opcja domyślna) lub jako nagłówek HTTP w postaci tokenu Bearer. Właściwość `Authentication.TokenParam` służy do konfiguracji tego zachowania.

- **srctQuery:** token `access_token` jest przekazywany w adresie URL zapytania połączenia WebSocket.
- **srctHeader:** `access_token` jest przekazywany jako nagłówek HTTP w postaci tokenu Bearer.

## srcaBasic

Ta opcja korzysta z uwierzytelniania **Basic Authentication** i wymaga skonfigurowania komponentu `SignalRCore` oraz [TsgcWebSocketClient](#).

**Przykład:** jeśli serwer wymaga uwierzytelniania podstawowego, a nazwa użytkownika to „user” i hasło to „secret”, należy skonfigurować składniki w sposób przedstawiony poniżej.

```
// websocket client
TsgcWebSocketClient* WSClient = new TsgcWebSocketClient();
WSClient->Authentication->Enabled = true;
WSClient->Authentication->Basic->Enabled = true;
WSClient->Authentication->URL->Enabled = false;
WSClient->Authentication->Session->Enabled = false;
WSClient->Authentication->Token->Enabled = false;
WSClient->Authentication->User = "user";
WSClient->Authentication->Password = "secret";
// signalrcore
TsgcWSAPI_SignalRCore* Signal = new TsgcWSAPI_SignalRCore();
Signal->SignalRCore->Authentication->Enabled = true;
Signal->SignalRCore->Authentication->Authentication = srcaBasic;
Signal->SignalRCore->Authentication->Username = "user";
Signal->SignalRCore->Authentication->Password = "secret";
Signal->Client = WSClient;
```

## Komunikacja między klientem a serwerem

Istnieją trzy rodzaje interakcji między serwerem a klientami:

### Wywołania

Wywołujący wysyła wiadomość do wywoływanego i oczekuje wiadomości informującej o zakończeniu wywołania oraz opcjonalnie wyniku wywołania

**Przykład:** klient wywołuje metodę `SendMessage` i przekazuje jako parametry nazwę użytkownika i treść

wiadomości. Wysyła identyfikator wywołania, aby otrzymać komunikat z wynikiem od serwera.

```
SignalRCore->Invoke("SendMessage", ARRAYOFCONST(("John", "Hello All.")), "id-000001");

void OnSignalRCoreCompletion(TObject *Sender, TSignalRCore_Completion *Completion)
{
    if (Completion->Error != "")
    {
        ShowMessage("Something goes wrong.");
    }
    else
    {
        ShowMessage("Invocation Successful!");
    }
}
```

## Wywołania nieblokujące

Wywołujący wysyła wiadomość do wywołanego i nie oczekuje żadnych dalszych wiadomości dla tego wywołania. Wywołania mogą być wysyłane bez wartości identyfikatora wywołania. Oznacza to, że wywołanie jest „nieblokujące”.

**Przykład:** klient wywołuje metodę SendMessage i przekazuje jako parametry nazwę użytkownika i treść wiadomości tekstowej. Klient nie oczekuje żadnej odpowiedzi od serwera na temat wyniku wywołania.

```
SignalRCore->Invoke("SendMessage", ARRAYOFCONST(("John", "Hello All.")));
```

## Wywołania strumieniowe

Wywołujący wysyła wiadomość do Wywoływanego i oczekuje jednego lub więcej wyników zwróconych przez Wywoływanego, a następnie wiadomości wskazującej koniec wywołania.

**Przykład:** klient wywołuje metodę Counter i żąda 10 liczb z interwałem 500 milisekund.

```
SignalRCore->InvokeStream("Counter", [10, 500], "id-000002");

void OnSignalRCoreStreamItem(TObject *Sender, TSignalRCore_StreamItem *StreamItem, bool &Cancel)
{
    DoLog("#stream item: " + StreamItem->Item);
}

void OnSignalRCoreCompletion(TObject *Sender, TSignalRCore_Completion *Completion)
{
    if (Completion->Error != "")
    {
        ShowMessage("Something goes wrong.");
    }
    else
    {
        ShowMessage("Invocation Successful!");
    }
}
```

## Wywołania

Aby wykonać pojedyncze wywołanie, Caller stosuje następujący podstawowy przepływ:

```
void Invoke(const string aTarget, const Array of Const aArguments, const string aInvocationId)
void InvokeStream(const string aTarget, const Array of Const aArguments, const String aInvocationId)
```

Należy przydzielić unikalną wartość Invocation ID (dowolny ciąg, wybrany przez wywołującego) reprezentującą wywołanie. Następnie wywołać metodę Invoke lub InvokeStream, przekazując Target, Arguments i InvocationId (brak InvocationId spowoduje brak wyniku zakończenia).

Jeśli wywołanie jest oznaczone jako nieblokujące (patrz "Wywołania nieblokujące" poniżej), należy tu zakończyć i natychmiast zwrócić sterowanie do aplikacji. Należy obsłużyć wiadomość StreamItem lub Completion z pasującym identyfikatorem wywołania.

```
SignalRCore->InvokeStream("Counter", [10, 500], "id-000002");

void OnSignalRCoreStreamItem(TObject *Sender, TSignalRCore_StreamItem *StreamItem, bool &Cancel)
{
    if (StreamItem->InvocationId == "id-000002")
    {
        DoLog("#stream item: " + StreamItem->Item);
    }
}

void OnSignalRCoreCompletion(TObject *Sender, TSignalRCore_Completion *Completion)
{
    if (Completion->InvocationId == "id-000002")
    {
        if (Completion->Error != "")
        {
            ShowMessage("Something goes wrong.");
        }
        else
        {
            ShowMessage("Invocation Successful!");
        }
    }
}
```

Można wykonać pojedyncze wywołanie i oczekiwać na jego zakończenie.

```
bool InvokeAndWait(const String aTarget, System::TVarRec *aArguments, string aInvocationId,
    TSignalRCore_Completion &Completion, const int aTimeout = 10000);
bool InvokeStreamAndWait(const String aTarget, System::TVarRec *aArguments, string aInvocationId,
    TSignalRCore_Completion &Completion, const int aTimeout = 10000);
```

Przypisz unikalną wartość identyfikatora wywołania (dowolny ciąg znaków wybrany przez wywołującego) do reprezentowania wywołania. Wywołaj metodę InvokeAndWait lub InvokeStreamAndWait z parametrami Target (cel wywołania), Arguments (argumenty) i InvocationId. Program będzie czekał do momentu wywołania zdarzenia zakończenia lub przekroczenia limitu czasu.

```
{
    TSignalRCore_Completion oCompletion;
    if (SignalRCore->InvokeStreamAndWait("Counter", ARRAYOFCONST((10, 500)), "id-000002", oCompletion))
```

```

{
    DoLog("#invoke stream ok: " + oCompletion->Result);
}
else
{
    DoLog("#invocke stream error: " + oCompletion->Error);
}
}

void OnSignalRCoreStreamItem(TObject *Sender, TSignalRCore_StreamItem *StreamItem, bool &Cancel)
{
    if (StreamItem->InvocationId == "id-000002")
    {
        DoLog("#stream item: " + StreamItem->Item);
    }
}

```

## Anuluj wywołanie

Jeśli klient chce przestać odbierać wiadomości StreamItem przed wysłaniem przez serwer wiadomości Completion, może wysłać wiadomość CancelInvocation z tym samym InvocationId, który był używany w wiadomości StreamInvocation inicjującej strumień.

```

void OnSignalRCoreStreamItem(TObject *Sender, TSignalRCore_StreamItem *StreamItem, bool &Cancel)
{
    if (StreamItem->InvocationId == "id-000002")
    {
        Cancel = true;
    }
}

```

## Wyniki klienta

Wywołanie jest uznawane za zakończone dopiero po odebraniu komunikatu Completion. Jeśli klient odbierze wywołanie z serwera, zostanie wywołane zdarzenie OnSignalRCoreInvocation.

```

void OnSignalRCoreInvocation(TObject *Sender, TSignalRCore_Invocation *Invocation)
{
    if (Invocation->Target == "SendMessage")
    {
        ... your code here ...
    }
}

// Once invocation is completed, call Completion method to inform server invocation is finished.
// If result is successful, then call CompletionResult method:
SignalRCore->CompletionResult("id-000002", "ok");

// If not, then call CompletionError method:
SignalRCore->CompletionError("id-000002", "Error processing invocation.");

```

## Zamknij połączenie

Wysyłane przez klienta po zamknięciu połączenia. Zawiera powód błędu, jeśli połączenie zostało zamknięte z powodu błędu.

```
SignalRCore->Close("Unexpected message").
```

```
// If the server close connection by any reason, OnSignalRCoreClose event will be called.
```

```
void OnSignalRCoreClose(TObject *Sender, TSignalRCore_Close *Close)
```

```
{  
    DoLog("#closed: " + Close->Error);  
}
```

## Ping

Protokół SignalR Hub obsługuje komunikaty „Keep Alive” służące do utrzymania aktywności połączenia transportowego. Komunikaty te zapewniają:

Że serwery proxy nie zamykają połączenia w czasie bezczynności (gdy przesyłanych jest niewiele wiadomości). Jeżeli połączenie zostanie przerwane bez poprawnego zakończenia, aplikacja jest o tym informowana możliwie najszybciej.

Zachowanie keep-alive osiąga się przez wywołanie metody Ping lub włączenie HeartBeat na kliencie WebSocket. Jeżeli serwer wyśle ping do klienta, klient automatycznie wyśle odpowiedź i zostanie wywołane zdarzenie OnSignalRCoreKeepAlive.

```
void OnSignalRCoreKeepAlive(TObject *Sender)
```

```
{  
    DoLog("#keepalive");  
}
```

## MessagePack

W kodowaniu MsgPack protokołu SignalR każda wiadomość jest reprezentowana jako pojedyncza tablica MsgPack zawierająca elementy odpowiadające właściwościom danej wiadomości protokołu hub. Elementy tablicy mogą być wartościami prymitywnymi, tablicami (np. argumenty metody) lub obiektami (np. wartość argumentu). Pierwszym elementem tablicy jest typ wiadomości.

Aby dowiedzieć się, jak kodować wysyłane wiadomości, należy zapoznać się z [dokumentacją MessagePack](#).

Za każdym razem, gdy odebrana zostaje nowa wiadomość, jest ona przekazywana w zdarzeniu OnSignalRCoreMessagePack. Wiadomość jest dostępna poprzez odczyt parametru strumienia Data. Parametr JSON jest domyślnie pusty; jeśli wiadomość MessagePack zostanie przekonwertowana na JSON, komponent przetworzy wiadomość JSON tak, jakby kodowanie używało JSON (a zatem zdarzenia OnSignalRCoreCompletion, OnSignalRCoreInvocation itp. zostaną wywołane).

# API SignalR

## SignalR

Komponent SignalR używa WebSocket jako transportu do łączenia się z serwerem SignalR; jeśli ten transport nie jest obsługiwany, zostanie zgłoszony błąd.

Komponent klienta SignalR ma właściwość `SignalR`, w której można ustawić następujące dane:

- **Hubs:** zawiera listę hubów, do których klient subskrybuje.
- **ProtocolVersion:** wersja protokołu używana przez klienta; obsługiwane są wersje protokołu od 1.2 do 1.5
- **UserAgent:** agent użytkownika używany do połączenia z serwerem SignalR.

Klient obsługuje wysyłanie danych tekstowych lub binarnych.

## Wiadomości hubów

Hubs API umożliwia wywoływanie metod serwera przez klienta oraz metod klienta przez serwer. Protokół używany do trwałych połączeń nie jest wystarczająco rozbudowany, aby wyrażać semantykę RPC (zdalnego wywołania procedur). Nie oznacza to jednak, że protokół używany dla połączeń z hubami jest całkowicie inny niż protokół dla połączeń trwałych. Protokół połączeń z hubami jest w dużej mierze rozszerzeniem protokołu dla połączeń trwałych.

Gdy klient wywołuje metodę serwera, nie wysyła już swobodnego ciągu znaków jak w przypadku połączeń trwałych. Zamiast tego wysyła ciąg JSON zawierający wszystkie informacje niezbędne do wywołania metody. Poniżej znajduje się przykładowa wiadomość, którą klient wysyłałby w celu wywołania metody serwera:

```
WriteData("{\"H\":\"chathub\",\"M\":\"Send\",\"A\":[\"CBuilder Client\",\"Test message\"],\"I\":0}");
```

Ładunek posiada następujące właściwości:

I — identyfikator wywołania — umożliwia dopasowanie odpowiedzi do żądań

H — nazwa huba

M — nazwa metody

A – argumenty (tablica; może być pusta, jeśli metoda nie ma żadnych parametrów)

Jeśli argument łańcuchowy zawiera **cudzysłówy**, należy zastąpić " przez \"

Przykład: jeśli argument to {"test":1}, należy wysłać argument jako {"test\\":1}

```
WriteData({'H':"chathub","M":"Send","A":["{"test\\":1}"],"I":0});
```

## Autoryzacja

Uwierzytelnianie można włączyć, aby powiązać użytkownika z każdym połączeniem i filtrować, którzy użytkownicy mogą uzyskiwać dostęp do zasobów. Uwierzytelnianie jest realizowane przy użyciu tokenów Bearer: klient dostarcza token dostępu, a serwer weryfikuje ten token i używa go do identyfikacji użytkownika.

Obecnie obsługiwane są wyłącznie tokeny Bearer:

Tutaj token jest przekazywany bezpośrednio do serwera Signal (ponieważ token został uzyskany z innego serwera).

- **Authentication.Enabled:** jeśli aktywne, autoryzacja będzie stosowana przed nawiązaniem połączenia WebSocket.
- **Authentication.Authentication:** Obsługiwane są 2 typy uwierzytelniania: token Bearer lub pliki cookie. Oba wymagają zewnętrznego sposobu uzyskania wymaganych wartości.
  - **BearerToken:** uzyskana wartość tokenu.
  - **Cookie:** ustaw wartość wymaganego pliku cookie.

```
TsgcWSAPI_Signal *oSignalR = new TsgcWSAPI_Signal(NULL);  
oSignalR->SignalR->Enabled = true;  
oSignalR->SignalR->Authentication = srcBearerToken;  
oSignalR->SignalR->BearerToken->Token = "token here";
```

Komponent posiada następujące zdarzenia:

## OnSignalRConnect

Zdarzenie to jest wywoływane, gdy klient pomyślnie połączy się z serwerem.

## OnSignalRDisconnect

Zdarzenie jest wywoływane po rozłączeniu klienta z serwerem.

## OnSignalRError

Zdarzenie jest wywoływane w przypadku wystąpienia błędu w połączeniu WebSocket.

## OnSignalRMessage

Protokół używany do połączenia stałego jest dość prosty. Wiadomości wysyłane do serwera są zwykłymi ciągami tekstowymi bez wymaganego określonego formatu. Wiadomości wysyłane do klienta mają bardziej uporządkowaną strukturę. Właściwości dostępne w wiadomości są następujące:

C – identyfikator wiadomości, obecny we wszystkich wiadomościach innych niż KeepAlive.

M – tablica zawierająca rzeczywiste dane.

```
{"C":"d-9B7A6976-B,2|C,2","M":["Welcome!"]}
```

## OnSignalRBinary

Zdarzenie to jest wywoływane po otrzymaniu danych binarnych z serwera.

## OnSignalRResult

Po wywołaniu metody serwera, serwer zwraca potwierdzenie zakończenia wywołania, wysyłając do klienta identyfikator wywołania oraz, jeśli metoda zwróciła wartość, tę wartość, lub, jeśli wywołanie metody nie powiodło się, błąd.

Poniżej przedstawiono przykładowe wyniki wywołania metody serwera:

```
{"I":"0"}
```

Metoda serwera void z identyfikatorem wywołania „0” zakończyła się pomyślnie.



```
{"I":"0", "R":42}
```

Metoda serwera zwracająca liczbę, której identyfikator wywołania wynosił "0", zakończyła się pomyślnie i zwróciła wartość 42.

```
{"I":"0", "E":"Error occurred"}
```

## **OnSignalRKeepAlive**

To zdarzenie jest wywoływane po odebraniu wiadomości KeepAlive z serwera.

# API Kraken

[Kraken](#)

## Przegląd

WebSockets API oferuje aktualizacje danych rynkowych w czasie rzeczywistym. WebSockets to protokół dwukierunkowy zapewniający najszybszy przekaz danych w czasie rzeczywistym, umożliwiając budowanie aplikacji czasu rzeczywistego. Przedstawione poniżej typy wiadomości publicznych nie wymagają uwierzytelniania. Wiadomości z danymi prywatnymi można subskrybować na oddzielnym uwierzytelnionym punkcie końcowym.

Kraken oferuje również REST API z publicznymi danymi rynkowymi i prywatnymi danymi użytkownika (wymagającymi uwierzytelniania).

## Konfiguracja

Prywatne API wymaga utworzenia klucza API na koncie Kraken.

Kraken umożliwia korzystanie ze środowiska testowego w protokole WebSocket; należy włączyć właściwość Beta z właściwości Kraken, aby skorzystać z tej funkcji beta.

## Obsługiwane API

- [WebSockets Public API](#): łączy się z publicznym serwerem WebSocket.
- [Prywatny API WebSockets](#): łączy się z prywatnym serwerem WebSocket i wymaga klucza API oraz sekretu API do uwierzytelnienia na serwerze.
- [Publiczne REST API](#): łączy się z publicznym serwerem REST.
- [REST Private API](#): łączy się z publicznym serwerem REST i wymaga klucza API oraz sekretu API do uwierzytelnienia na serwerze.

## Przykłady Kraken

### Jak połączyć się z publicznym serwerem WebSocket

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Kraken *oKraken = new TsgcWSAPI_Kraken(NULL);
oKraken->Client = oClient;
oClient->Active = true;
```

## Jak połączyć się z prywatnym serwerem WebSocket

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Kraken *oKraken = new TsgcWSAPI_Kraken(NULL);
oKraken->Kraken->ApiKey = "your api key";
oKraken->Kraken->ApiSecret = "your api secret";
oKraken->Client = oClient;
oClient->Active = true;
```

## Jak pobrać ticker z REST API

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Kraken *oKraken = new TsgcWSAPI_Kraken(NULL);
oKraken->Client = oClient;
ShowMessage(oKraken->GetTicker(ARRAYOFCONST(("XBTUSD"))));
```

## Metody REST API

### Publiczne punkty końcowe

|                        |                                  |
|------------------------|----------------------------------|
| <b>GetSystemStatus</b> | Pobierz aktualny status systemu. |
|------------------------|----------------------------------|

### Prywatne punkty końcowe

|                               |                                                                            |
|-------------------------------|----------------------------------------------------------------------------|
| <b>GetExtendedBalance</b>     | Pobierz rozszerzone informacje o saldzie.                                  |
| <b>AmendOrder</b>             | Zmiana istniejącego zlecenia.                                              |
| <b>CancelAllOrders</b>        | Anuluj wszystkie otwarte zlecenia.                                         |
| <b>CancelAllOrdersAfter</b>   | Martwy przełącznik — anuluje wszystkie zamówienia po upływie limitu czasu. |
| <b>EditOrder</b>              | Edytuj istniejące zlecenie.                                                |
| <b>AddOrderBatch</b>          | Wsadowe dodawanie wielu zleceń.                                            |
| <b>CancelOrderBatch</b>       | Anuluj wiele zleceń jednocześnie.                                          |
| <b>GetWithdrawalMethods</b>   | Pobierz dostępne metody wypłat.                                            |
| <b>GetWithdrawalAddresses</b> | Pobierz adresy do wypłat.                                                  |

## Jak uzyskać saldo konta przez REST API

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Kraken *oKraken = new TsgcWSAPI_Kraken(NULL);
oKraken->Kraken->ApiKey = "your api key";
oKraken->Kraken->ApiSecret = "your api secret";
oKraken->Client = oClient;
ShowMessage(oKraken->GetAccountBalance());
```

# API Kraken | Publiczne API WebSocket

## Połączenie

URL: `wss://ws.kraken.com` (v1) lub `wss://ws.kraken.com/v2` (v2, zalecane)

Komponent obsługuje teraz WebSocket API v2 poprzez właściwość **Version** (domyślnie: 2). Należy ustawić `Kraken.Version := 1`, aby korzystać ze starszego punktu końcowego v1.

Po otwarciu gniazda można subskrybować publiczny kanał, wysyłając wiadomość z żądaniem subskrypcji.

## Uwagi ogólne

- Wszystkie wiadomości wysyłane i odbierane przez WebSockets są kodowane w formacie JSON
- Wszystkie pola zmiennoprzecinkowe (w tym znaczniki czasu) są ujęte w cudzysłów w celu zachowania precyzji.
- Format każdej pary handlowej to A/B, gdzie A i B są kodami ISO 4217-A3 dla standaryzowanych aktywów lub popularnym unikatowym symbolem w przypadku braku standaryzacji.
- Znaczników czasu nie należy traktować jako unikalnych ani używać jako aliasów identyfikatorów transakcji. Ponadto ziarnistość znaczników czasu nie odzwierciedla częstotliwości transakcji.

## Obsługiwane pary

ADA/CAD, ADA/ETH, ADA/EUR, ADA/USD, ADA/XBT, ATOM/CAD, ATOM/ETH, ATOM/EUR, ATOM/USD, ATOM/XBT, BCH/EUR, BCH/USD, BCH/XBT, DASH/EUR, DASH/USD, DASH/XBT, EOS/ETH, EOS/EUR, EOS/USD, EOS/XBT, GNO/ETH, GNO/EUR, GNO/USD, GNO/XBT, QTUM/CAD, QTUM/ETH, QTUM/EUR, QTUM/USD, QTUM/XBT, USDT/USD, ETC/ETH, ETC/XBT, ETC/EUR, ETC/USD, ETH/XBT, ETH/CAD, ETH/EUR, ETH/GBP, ETH/JPY, ETH/USD, LTC/XBT, LTC/EUR, LTC/USD, MLN/ETH, MLN/XBT, REP/ETH, REP/XBT, REP/EUR, REP/USD, STR/EUR, STR/USD, XBT/CAD, XBT/EUR, XBT/GBP, XBT/JPY, XBT/USD, BTC/CAD, BTC/EUR, BTC/GBP, BTC/JPY, BTC/USD, XDG/XBT, XLM/XBT, DOGE/XBT, STR/XBT, XLM/EUR, XLM/USD, XMR/XBT, XMR/EUR, XMR/USD, XRP/XBT, XRP/CAD, XRP/EUR, XRP/JPY, XRP/USD, ZEC/XBT, ZEC/EUR, ZEC/JPY, ZEC/USD, XTZ/CAD, XTZ/ETH, XTZ/EUR, XTZ/USD, XTZ/XBT

## Metody

### Ping

Klient może wysłać ping do serwera, aby sprawdzić, czy połączenie jest aktywne; serwer odpowiada pongiem.

Jest to ping na poziomie aplikacji, w odróżnieniu od domyślnego pinga w standardzie WebSockets, który jest inicjowany przez serwer

## Ticker

Informacje o kursach obejmują najlepsze ceny kupna i sprzedaży, wolumen z ostatnich 24 godzin, cenę ostatniej transakcji, ważoną wolumenem średnią cenę itp. dla danej pary walutowej. Wiadomość kursowa jest publikowana za każdym razem, gdy odbywa się transakcja lub grupa transakcji.

Subskrybuj ticker, wywołując metodę `SubscribeTicker`:

```
SubscribeTicker(["XBT/USD"]);
```

Jeśli subskrypcja zakończy się sukcesem, zostanie wywołane zdarzenie **OnKrakenSubscribed**:

```
void OnKrakenSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription, string ChannelName,
    int ReqID)
{
    DoLog("#subscribed: " + Subscription + " " + Pair + " " + ChannelName);
}
```

Anulowanie subskrypcji przez wywołanie metody `UnSubscribeTicker`:

```
UnSubscribeTicker(ARRAYOFCONST(["XBT/USD"]));
```

Jeśli anulowanie subskrypcji zakończy się powodzeniem, zostanie wywołane zdarzenie `OnKrakenUnSubscribed`:

```
void OnKrakenUnSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#unsubscribed: " + Subscription + " " + Pair);
}
```

Jeśli podczas próby subskrypcji / anulowania subskrypcji wystąpi błąd, zostanie wywołane zdarzenie `OnKrakenSubscriptionError`.

```
void OnKrakenSubscriptionError(TObject *Sender, string ErrorMessage, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#subscription error: " + ErrorMessage);
}
```

Aktualizacje tickera będą zgłaszane w zdarzeniu `OnKrakenData`.

```
[
  0,
  {
    "a": [
      "5525.40000",
      1,
      "1.000"
    ],
    "b": [
      "5525.10000",
      1,
      "1.000"
    ],
    "c": [
      "5525.10000",
      "0.00398963"
    ]
  }
]
```

```

    ],
    "v": [
        "2634.11501494",
        "3591.17907851"
    ],
    "p": [
        "5631.44067",
        "5653.78939"
    ],
    "t": [
        11493,
        16267
    ],
    "l": [
        "5505.00000",
        "5505.00000"
    ],
    "h": [
        "5783.00000",
        "5783.00000"
    ],
    "o": [
        "5760.70000",
        "5763.40000"
    ]
  ],
  "ticker",
  "XBT/USD"
]

```

## OHLC

Po zasubskrybowaniu OHLC wysyłana jest migawka ostatniej prawidłowej świecy (niezależnie od czasu zakończenia), a następnie aktualizacje bieżącej świecy. Na przykład, jeśli subskrypcja dotyczy świecy 1-minutowej i przez 5 minut nie było żadnych transakcji, opublikowana zostanie migawka ostatniej świecy 1-minutowej sprzed 5 minut. Parametr `endtime` pozwala ustalić, czy jest to stara świeca.

Aby subskrybować OHLC, należy wywołać metodę `SubscribeOHLC`, przekazując parę i interwał.

```
SubscribeOHLC(ARRAYOFCONST(("XBT/USD")), kin1min);
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie `OnKrakenSubscribed`:

```

void OnKrakenSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription, string ChannelName,
    int ReqID)
{
    DoLog("#subscribed: " + Subscription + " " + Pair + " " + ChannelName);
}

```

Anulowanie subskrypcji przez wywołanie metody `UnSubscribeOHLC`:

```
UnSubscribeOHLC(ARRAYOFCONST(("XBT/USD")), kin1min);
```

Jeśli anulowanie subskrypcji zakończy się powodzeniem, zostanie wywołane zdarzenie `OnKrakenUnSubscribed`:

```

void OnKrakenUnSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    int ReqID)
{

```

```

    DoLog("#unsubscribed: " + Subscription + " " + Pair);
}

```

Jeśli podczas próby subskrypcji / anulowania subskrypcji wystąpi błąd, zostanie wywołane zdarzenie OnKrakenSubscriptionError.

```

void OnKrakenSubscriptionError(TObject *Sender, string ErrorMessage, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#subscription error: " + ErrorMessage);
}

```

Aktualizacje OHLC będą powiadamiane w zdarzeniu OnKrakenData.

```

[
    42,
    [
        "1542057314.748456",
        "1542057360.435743",
        "3586.70000",
        "3586.70000",
        "3586.60000",
        "3586.60000",
        "3586.68894",
        "0.03373000",
        2
    ],
    "ohlcv-5",
    "XBT/USD"
]

```

## Transakcja

Kanał handlowy dla pary walutowej.

Subskrybuj kanał transakcji, wywołując metodę SubscribeTrade.

```

SubscribeTrade(ARRAYOFCONST((L"XBT/USD")));

```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie OnKrakenSubscribed:

```

void OnKrakenSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    string ChannelName, int ReqID)
{
    DoLog("#subscribed: " + Subscription + " " + Pair + " " + ChannelName);
}

```

Anuluj subskrypcję wywołując metodę UnSubscribeTrade:

```

UnSubscribeTrade(ARRAYOFCONST((L"XBT/USD")));

```

Jeśli anulowanie subskrypcji zakończy się powodzeniem, zostanie wywołane zdarzenie OnKrakenUnSubscribed:

```

void OnKrakenUnSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    int ReqID)
{

```

```

    DoLog("#unsubscribed: " + Subscription + " " + Pair);
}

```

Jeśli podczas próby subskrypcji / anulowania subskrypcji wystąpi błąd, zostanie wywołane zdarzenie OnKrakenSubscriptionError.

```

void OnKrakenSubscriptionError(TObject *Sender, string ErrorMessage, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#subscription error: " + ErrorMessage);
}

```

Aktualizacje transakcji będą powiadamiane w zdarzeniu OnKrakenData.

```

[
    0,
    [
        [
            "5541.20000",
            "0.15850568",
            "1534614057.321597",
            "s",
            "l",
            ""
        ],
        [
            "6060.00000",
            "0.02455000",
            "1534614057.324998",
            "b",
            "l",
            ""
        ]
    ],
    "trade",
    "XBT/USD"
]

```

## Książka

Poziomy arkusza zleceń. Po subskrypcji publikowana jest migawka przy określonej głębokości; po niej publikowane są aktualizacje poziomów.

Subskrybuj książkę zleceń, wywołując metodę SubscribeBook; należy przekazać parę i głębokość.

```

SubscribeBook(ARRAYOFCNST((L"XBT/USD")), kde10);

```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie OnKrakenSubscribed:

```

void OnKrakenSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription, string ChannelName,
    int ReqID)
{
    DoLog("#subscribed: " + Subscription + " " + Pair + " " + ChannelName);
}

```

Anulowanie subskrypcji przez wywołanie metody UnSubscribeBook:

```

UnSubscribeBook(ARRAYOFCNST((L"XBT/USD")), kde10);

```



Jeśli anulowanie subskrypcji zakończy się powodzeniem, zostanie wywołane zdarzenie OnKrakenUnSubscribed:

```
void OnKrakenUnSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#unsubscribed: " + Subscription + " " + Pair);
}
```

Jeśli podczas próby subskrypcji / anulowania subskrypcji wystąpi błąd, zostanie wywołane zdarzenie OnKrakenSubscriptionError.

```
void OnKrakenSubscriptionError(TObject *Sender, string ErrorMessage, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#subscription error: " + ErrorMessage);
}
```

Aktualizacje księgi będą powiadamiane w zdarzeniu OnKrakenData.

```
[
    0,
    {
        "as": [
            [
                "5541.30000",
                "2.50700000",
                "1534614248.123678"
            ],
            [
                "5541.80000",
                "0.33000000",
                "1534614098.345543"
            ],
            [
                "5542.70000",
                "0.64700000",
                "1534614244.654432"
            ]
        ],
        "bs": [
            [
                "5541.20000",
                "1.52900000",
                "1534614248.765567"
            ],
            [
                "5539.90000",
                "0.30000000",
                "1534614241.769870"
            ],
            [
                "5539.50000",
                "5.00000000",
                "1534613831.243486"
            ]
        ]
    },
    "book-100",
    "XBT/USD"
]
```

## Spread

Kanał spreadów pokazujący najlepszą cenę kupna i sprzedaży dla subskrybowanej pary aktywów. Wiadomość zawiera również wolumeny po stronie kupna i sprzedaży.

Subskrybuj kanał Spread, wywołując metodę `SubscribeSpread`.

```
SubscribeSpread(ARRAYOFCONST(("XBT/USD")));
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie `OnKrakenSubscribed`:

```
void OnKrakenSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription, string ChannelName,
    int ReqID)
{
    DoLog("#subscribed: " + Subscription + " " + Pair + " " + ChannelName);
}
```

Anulowanie subskrypcji przez wywołanie metody `UnSubscribeSpread`:

```
UnSubscribeSpread(ARRAYOFCONST(("XBT/USD")));
```

Jeśli anulowanie subskrypcji zakończy się powodzeniem, zostanie wywołane zdarzenie `OnKrakenUnSubscribed`:

```
void OnKrakenUnSubscribed(TObject *Sender, int ChannelId, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#unsubscribed: " + Subscription + " " + Pair);
}
```

Jeśli podczas próby subskrypcji / anulowania subskrypcji wystąpi błąd, zostanie wywołane zdarzenie `OnKrakenSubscriptionError`.

```
void OnKrakenSubscriptionError(TObject *Sender, string ErrorMessage, string Pair, string Subscription,
    int ReqID)
{
    DoLog("#subscription error: " + ErrorMessage);
}
```

Aktualizacje spreadu będą powiadamiane w zdarzeniu `OnKrakenData`.

```
[
    0,
    [
        "5698.40000",
        "5700.00000",
        "1542057299.545897",
        "1.01234567",
        "0.98765432"
    ],
    "spread",
    "XBT/USD"
]
```

## Inne metody

Można subskrybować i anulować subskrypcję wszystkich kanałów jedną metodą:

```
SubscribeAll(ARRAYOFCONST((L"XBT/USD")));
```

```
UnSubscribeAll(ARRAYOFCONST((L"XBT/USD")));
```

Wartość interwału OHLC wynosi 1, jeśli subskrybowane są wszystkie kanały.

## Zdarzenia

**OnConnect:** wywoływane po połączeniu klienta websocket z serwerem.

**OnKrakenConnect:** wywoływane po pomyślnym połączeniu WebSocket i gdy serwer wyśle status systemu.

**OnKrakenSystemStatus:** wywoływane przy zmianie statusu systemu.

**OnKrakenSubscribed:** wywoływane po pomyślnej subskrypcji kanału.

**OnKrakenUnSubscribed:** wywoływane po pomyślnym anulowaniu subskrypcji kanału.

**OnKrakenSubscriptionError:** wywoływane, gdy wystąpił błąd podczas próby subskrypcji lub anulowania subskrypcji.

**OnKrakenData:** wywoływane za każdym razem, gdy subskrypcja kanału otrzymuje aktualizację.

# API Kraken | Prywatny API WebSockets

## Połączenie

URL: `wss://ws-auth.kraken.com` (v1) lub `wss://ws-auth.kraken.com/v2` (v2, zalecane)

Komponent obsługuje teraz WebSocket API v2 za pośrednictwem właściwości **Version** (domyślnie: 2).

Po otwarciu gniazda można subskrybować kanały danych prywatnych, wysyłając uwierzytelnione żądanie subskrypcji.

## Uwierzytelnianie

Klient API musi zażądać tokenu uwierzytelniającego za pośrednictwem następującego punktu końcowego REST API „GetWebSocketsToken”, aby połączyć się z prywatnymi punktami końcowymi WebSockets. Token powinien zostać użyty w ciągu 15 minut od jego utworzenia. Token nie wygasa po nawiązaniu połączenia z prywatną wiadomością API WebSockets (openOrders lub ownTrades).

Aby uzyskać token WebSockets, należy ustawić klucz API oraz sekret API w komponencie Kraken Options — klucz API jest dostarczany przez Kraken w ramach konta użytkownika.

```
Kraken->ApiKey = "api key";  
Kraken->ApiSecret = "api secret";
```

## Metody

### OwnTrades

Pobierz listę własnych transakcji. Przy pierwszej subskrypcji otrzymywana jest lista ostatnich 50 transakcji.

```
SubscribeOwnTrades();
```

Później można anulować subskrypcję OwnTrades, wywołując metodę UnSubscribeOwnTrades

```
UnSubscribeOwnTrades();
```

Przykładowa odpowiedź serwera

```
[  
  [  
    {  
      "TDLH43-DVQXD-2KHVYY": {  
        "cost": "1000000.00000",  
        "fee": "600.00000",  
        "margin": "0.00000",  
        "ordertxid": "TDLH43-DVQXD-2KHVYY",  
        "ordertype": "limit",  
        "pair": "XBT/EUR",
```

```

        "postxid": "OGTT3Y-C6I3P-XRI6HX",
        "price": "100000.00000",
        "time": "1560520332.914664",
        "type": "buy",
        "vol": "1000000000.00000000"
    }
}
],
"ownTrades"
]

```

## Otwarte zlecenia

Kanał pokazujący wszystkie otwarte zlecenia należące do uwierzytelnionego klucza API użytkownika. Początkowa migawka zawiera listę wszystkich otwartych zleceń, a następnie wszelkie aktualizacje listy otwartych zleceń są wysyłane na bieżąco. W przypadku aktualizacji statusu, takich jak „zamknięte”, ładunek zawiera pola orderid i status.

```
SubscribeOpenOrders();
```

Można później anulować subskrypcję OpenOrders, wywołując metodę UnSubscribeOpenOrders.

```
UnSubscribeOpenOrders();
```

## Przykładowa odpowiedź serwera

```

[
  [
    {
      "OGTT3Y-C6I3P-XRI6HX": {
        "cost": "0.00000",
        "descr": {
          "close": "",
          "leverage": "0:1",
          "order": "sell 0.00001000 XBT/EUR @ limit 9.00000 with 0:1 leverage",
          "ordertype": "limit",
          "pair": "XBT/EUR",
          "price": "9.00000",
          "price2": "0.00000",
          "type": "sell"
        },
        "expiretm": "0.000000",
        "fee": "0.00000",
        "limitprice": "9.00000",
        "misc": "",
        "oflags": "fcib",
        "opentm": "0.000000",
        "price": "9.00000",
        "refid": "OKIVMP-5GVZN-Z2D2UA",
        "starttm": "0.000000",
        "status": "open",
        "stopprice": "0.000000",
        "userref": 0,
        "vol": "0.00001000",
        "vol_exec": "0.00000000"
      }
    }
  ],
  "openOrders"
]

```

## Dodaj zlecenie

### Wyślij nowe zlecenie do Kraken

```
TsgcWSKrakenOrder *oKrakenOrder = new TsgcWSKrakenOrder();
oKrakenOrder->Pair = "XBT/USD";
oKrakenOrder->_Type = kosBuy;
oKrakenOrder->OrderType = kotMarket;
oKrakenOrder->Volume = 1;
AddOrder(oKrakenOrder);
```

### Lista parametrów zlecenia

```
pair = asset pair
type = type of order (buy/sell)
ordertype = order type:
    market
    limit (price = limit price)
    stop-loss (price = stop loss price)
    take-profit (price = take profit price)
    stop-loss-profit (price = stop loss price, price2 = take profit price)
    stop-loss-profit-limit (price = stop loss price, price2 = take profit price)
    stop-loss-limit (price = stop loss trigger price, price2 = triggered limit price)
    take-profit-limit (price = take profit trigger price, price2 = triggered limit price)
    trailing-stop (price = trailing stop offset)
    trailing-stop-limit (price = trailing stop offset, price2 = triggered limit offset)
    stop-loss-and-limit (price = stop loss price, price2 = limit price)
    settle-position
price = price (optional. dependent upon ordertype)
price2 = secondary price (optional. dependent upon ordertype)
volume = order volume in lots
leverage = amount of leverage desired (optional. default = none)
oflags = comma delimited list of order flags (optional):
    viqc = volume in quote currency (not available for leveraged orders)
    fcib = prefer fee in base currency
    fciq = prefer fee in quote currency
    nompp = no market price protection
    post = post only order (available when ordertype = limit)
starttm = scheduled start time (optional):
    0 = now (default)
    +<n> = schedule start time <n> seconds from now
    <n> = unix timestamp of start time
expiretm = expiration time (optional):
    0 = no expiration (default)
    +<n> = expire <n> seconds from now
    <n> = unix timestamp of expiration time
userref = user reference id. 32-bit signed number. (optional)
validate = validate inputs only. do not submit order (optional)
optional closing order to add to system when order gets filled:
    close[ordertype] = order type
    close[price] = price
    close[price2] = secondary price
```

### Przykładowa odpowiedź serwera

```
{
  "descr": "buy 0.01770000 XBTUSD @ limit 4000",
  "event": "addOrderStatus",
  "status": "ok",
  "txid": "ONPNXH-KMKMU-F4MR5V"
}
```

## Anuluj zlecenie

Anuluj zlecenie

```
CancelOrder("Order Id");
```

Przykładowa odpowiedź serwera

```
{  
  "event": "cancelOrderStatus",  
  "status": "ok"  
}
```

# API Kraken | Publiczne REST API

## Połączenie

URL: <https://api.kraken.com>

Publiczne API Kraken nie wymaga żadnego uwierzytelniania.

## Konfiguracja

Jedyna konfiguracja dotyczy włączenia dziennika dla żądań HTTP REST. Należy włączyć HTTPLogOptions, jeśli zachowanie wszystkich żądań/odpowiedzi HTTP w pliku tekstowym jest pożądane.

## Zdarzenia

**OnKrakenHTTPException:** zdarzenie wywoływane w przypadku wystąpienia wyjątku podczas wykonywania żądania HTTP z REST API.

## Metody

### GetServerTime

Metoda pomocnicza służąca do przybliżonego wyznaczenia różnicy czasu między serwerem a klientem. Zwraca czas w formacie Unix.

```
{"error":[],"result":{"unixtime":1586705546,"rfc1123":"Sun, 12 Apr 20 15:32:26 +0000"}}
```

### GetAssets

Zwraca informacje o zasobach

```
{"error":[],"result":{"ADA":{"aclass":"currency","altname":"ADA","decimals":8,"display_decimals":6}}}}
```

### GetAssetPairs

Zwraca informacje o parze aktywów

```
Kraken->REST_API->GetAssetPairs(ARRAYOFCONST((L"XBTUSD")));
```

### GetTicker

Zwraca informacje o tickerze



```
Kraken->REST_API->GetTicker(ARRAYOFCONST((L"XBTUSD")));
```

## **GetOHLC**

Zwraca dane Open-High-Low-Close.

```
Kraken->REST_API->GetOHLC("XBTUSD");
```

## **GetOrderBook**

Zwraca parę nazw tablicy i głębokość rynku.

```
Kraken->REST_API->GetOrderBook("XBTUSD");
```

## **GetTrades**

Zwraca ostatnie dane transakcji dla pary.

```
Kraken->REST_API->GetTrades("XBTUSD");
```

## **GetSpread**

Zwraca ostatnie dane spreadu dla pary.

```
Kraken->REST_API->GetSpread("XBTUSD");
```

# API Kraken | Prywatne REST API

## Połączenie

URL: <https://api.kraken.com>

## Uwierzytelnianie

Prywatne REST API wymaga klucza API i sekretu API; wartości te są dostarczane przez Kraken na koncie użytkownika.

```
Kraken->ApiKey = "api key";  
Kraken->ApiSecret = "api secret";
```

## Metody

### GetAccountBalance

Zwraca saldo konta.

```
Kraken->REST_API->GetAccountBalance();
```

### GetTradeBalance

Zwraca informacje o transakcjach.

```
Kraken->REST_API->GetTradeBalance();
```

### GetOpenOrders

Zwraca listę otwartych zleceń.

```
Kraken->REST_API->GetOpenOrders();
```

### GetClosedOrders

Zwraca listę zamkniętych zleceń.

```
Kraken->REST_API->GetClosedOrders();
```

### QueryOrders

Zapytanie o informacje dotyczące zlecenia.

```
Kraken->REST_API->QueryOrders("1234");
```

## **GetTradesHistory**

Zwraca tablicę informacji o transakcjach.

```
Kraken->REST_API->GetTradesHistory();
```

## **QueryTrades**

Zapytanie o informacje dotyczące transakcji.

```
Kraken->REST_API->QueryTrades("1234");
```

## **GetOpenPositions**

Zwraca informacje o pozycji.

```
Kraken->REST_API->GetOpenPositions("1234");
```

## **GetLedgers**

Zwraca tablicę asocjacyjną z informacjami o księgach.

```
Kraken->REST_API->GetLedgers();
```

## **QueryLedgers**

Zwraca tablicę asocjacyjną z informacjami o księgach.

```
Kraken->REST_API->QueryLedgers("1234");
```

## **GetTradeVolume**

Zwraca informacje o wolumenie handlu.

```
Kraken->REST_API->GetTradeVolume();
```

## **AddExport**

Dodaje nowy eksport raportu.

```
Kraken->REST_API->AddExport("Report All Trades");
```

## **ExportStatus**

Pobierz status raportów

Kraken->REST\_API->ExportStatus();

## RetrieveExport

Pobierz raport według identyfikatora raportu.

Kraken->REST\_API->RetrieveExport("GOCO");

## RemoveExport

Usuwanie raportu według identyfikatora raportu.

Kraken->REST\_API->RemoveExport("GOCO");

## Dodaj zlecenie

Dodaje nowe zlecenie

```
pair = asset pair
type = type of order (buy/sell)
ordertype = order type:
    market
    limit (price = limit price)
    stop-loss (price = stop loss price)
    take-profit (price = take profit price)
    stop-loss-profit (price = stop loss price, price2 = take profit price)
    stop-loss-profit-limit (price = stop loss price, price2 = take profit price)
    stop-loss-limit (price = stop loss trigger price, price2 = triggered limit price)
    take-profit-limit (price = take profit trigger price, price2 = triggered limit price)
    trailing-stop (price = trailing stop offset)
    trailing-stop-limit (price = trailing stop offset, price2 = triggered limit offset)
    stop-loss-and-limit (price = stop loss price, price2 = limit price)
    settle-position
price = price (optional. dependent upon ordertype)
price2 = secondary price (optional. dependent upon ordertype)
volume = order volume in lots
leverage = amount of leverage desired (optional. default = none)
oflags = comma delimited list of order flags (optional):
    viqc = volume in quote currency (not available for leveraged orders)
    fcib = prefer fee in base currency
    fciq = prefer fee in quote currency
    nompp = no market price protection
    post = post only order (available when ordertype = limit)
starttm = scheduled start time (optional):
    0 = now (default)
    +n = schedule start time n seconds from now
    n = unix timestamp of start time
expiretm = expiration time (optional):
    0 = no expiration (default)
    +n = expire n seconds from now
    n = unix timestamp of expiration time
userref = user reference id. 32-bit signed number. (optional)
validate = validate inputs only. do not submit order (optional)
optional closing order to add to system when order gets filled:
    close[ordertype] = order type
    close[price] = price
    close[price2] = secondary price
```

```
TsgcHTTPKrakenOrder *oKrakenOrder = new TsgcHTTPKrakenOrder();  
oKrakenOrder->Pair = "XBT/USD";  
oKrakenOrder->_Type = koshBuy;  
oKrakenOrder->OrderType = kothMarket;  
oKrakenOrder->Volume = 1;  
Kraken->REST_API->AddOrder(oKrakenOrder);
```

## **CancelOrder**

Anuluje otwarte zlecenie na podstawie identyfikatora

```
Kraken->REST_API->CancelOrder("1234");
```

# API Kraken Futures

## Kraken Futures

### Przegląd

**REST API** umożliwia bezpieczny dostęp do metod konta Kraken Futures. Przykłady metod REST API:

- żądanie bieżących lub historycznych informacji o cenach
- sprawdzenie salda konta i PnL
- Twoje parametry depozytu zabezpieczającego i szacowane progi likwidacji
- składanie lub anulowanie zleceń (pojedynczo lub w partiach)
- wyświetl swoje otwarte zlecenia
- otwarte pozycje lub historia transakcji
- złóż żądanie wypłaty zasobu cyfrowego

Metody te są nazywane „punktami końcowymi” i zostały opisane w sekcji REST API.

**Websocket API** umożliwia bezpieczne ustanowienie kanału komunikacyjnego z platformą Kraken Futures w celu odbierania informacji w czasie rzeczywistym. Pozwala to na nasłuchiwanie aktualizacji zamiast ciągłego wysyłania żądań. Kanały te nazywane są subskrypcjami.

Niektóre punkty końcowe umożliwiają wykonywanie wrażliwych zadań, takich jak inicjowanie wypłaty aktywów cyfrowych. Aby uzyskać bezpieczny dostęp do tych punktów końcowych, API wykorzystuje techniki szyfrowania opracowane przez Agencję Bezpieczeństwa Narodowego.

### Konfiguracja

Aby korzystać z API, należy wygenerować parę unikalnych **kluczy API** (jeśli wymagany jest dostęp do prywatnych API):

1. Zaloguj się na swoje konto **Kraken Futures**.
2. Kliknij swoją nazwę w prawym górnym rogu.
3. Wybierz „Ustawienia” z menu rozwijanego.
4. Wybierz kartę „Utwórz klucz” w panelu API.
5. Naciśnij przycisk "Utwórz klucz".
6. Wyświetl swoje klucze publiczny i prywatny i zapisz je w bezpiecznym miejscu.

Skopiuj klucze publiczny i prywatny do właściwości **KrakenOptions** komponentu.

**KrakenOptions.ApiKey**

**KrakenOptions.ApiSecret**

## Obsługiwane API

- [WebSockets Public API](#): łączy się z publicznym serwerem WebSocket.
- [Prywatne API WebSockets](#): łączy się z prywatnym serwerem WebSocket i wymaga klucza API oraz sekretu API do uwierzytelnienia wobec serwera.
- [REST Public API](#): łączy się z publicznym serwerem REST.
- [REST Private API](#): łączy się z publicznym serwerem REST i wymaga klucza API oraz sekretu API do uwierzytelnienia wobec serwera.

## Metody REST API

### Prywatne punkty końcowe

|                                    |                                                                 |
|------------------------------------|-----------------------------------------------------------------|
| <b>BatchOrder</b>                  | Przesyłanie zleceń wsadowych (składanie/anulowanie/edytowanie). |
| <b>GetOrderStatus</b>              | Pobierz status określonych zleceń.                              |
| <b>GetPNL-Currency-Preferences</b> | Pobierz preferencje waluty PNL.                                 |
| <b>SetPNL-Currency-Preference</b>  | Ustawia preferencję waluty PNL dla danego symbolu.              |
| <b>GetLeverageSettings</b>         | Pobierz preferencje dźwigni.                                    |
| <b>SetLeverageSettings</b>         | Ustawienie dźwigni finansowej dla symbolu.                      |

# API Kraken Futures | WebSockets Public API

## Połączenie

URL: wss://futures.kraken.com/ws/v1

Po otwarciu gniazda można subskrybować publiczny kanał, wysyłając wiadomość z żądaniem subskrypcji.

## Metody

### Ticker

Ten punkt końcowy zwraca aktualne dane rynkowe dla wszystkich aktualnie notowanych kontraktów Futures i indeksów. Uwierzytelnianie nie jest wymagane.

Subskrybuj ticker, wywołując metodę **SubscribeTicker**:

```
SubscribeTicker(ARRAYOFCONST(("PI_XBTUSD")));
```

Jeśli subskrypcja zakończy się powodzeniem, zostanie wywołane zdarzenie **OnKrakenFuturesSubscribed**:

```
void OnKrakenFuturesSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#subscribed: " + Feed + " " + ProductId);
}
```

Anulowanie subskrypcji przez wywołanie metody **UnSubscribeTicker**:

```
UnSubscribeTicker(ARRAYOFCONST(("PI_XBTUSD")));
```

Jeśli anulowanie subskrypcji powiedzie się, zostanie wywołane zdarzenie **OnKrakenFuturesUnSubscribed**:

```
void OnKrakenFuturesUnSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#unsubscribed: " + Feed + " " + ProductId);
}
```

W przypadku błędu podczas próby subskrypcji lub anulowania subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesError**.

```
void OnKrakenFuturesError(TObject *Sender, string Error)
{
}
```



```
    DoLog("#error: " + Error);  
}
```

Aktualizacje tickera będą powiadamiane w zdarzeniu **OnKrakenData**.

```
{    "result": "success",
```

```
"tickers": [
```

```
{
```

```
"tag": "perpetual",
```

```
"pair": "XBT:USD",
```

```
"symbol": "pi_xbtusd",
```

```
"markPrice": 9520.2,
```

```
"bid": 9520,
```

```
"bidSize": 30950,
```

```
"ask": 9520.5,
```

```
"askSize": 3779,
```

```
"vol24h": 68238712,
```

```
"openInterest": 29308193,
```

```
"open24h": 10137,
```

"last": 9521,

"lastTime": "2020-06-03T08:14:26.624Z",

"lastSize": 1,

"suspended": false,

"fundingRate": 4.943012455e-9,

"fundingRatePrediction": 4.414499215e-9

}

{

"tag": "quarter",

"pair": "XBT:USD",

"symbol": "fi\_xbtusd\_200925",

"markPrice": 9659.8,

"bid": 9659.5,

"bidSize": 6480,

"ask": 9660,

"askSize": 17100,

"vol24h": 4562580,

"openInterest": 3573325,

"open24h": 10370.5,

"last": 9660,

"lastTime": "2020-06-03T08:10:37.800Z",

"lastSize": 5000,

"suspended": false

},

{

"symbol": "in\_xbtusd",

"last": 9519,

"lastTime": "2020-06-03T08:14:49.000Z"

}

```
],
```

```
"serverTime": "2020-06-03T08:14:49.865Z"
```

```
}
```

## Transakcja

Kanał transakcji zwraca informacje o zrealizowanych transakcjach

Subskrybuj kanał transakcji, wywołując metodę **SubscribeTrade**.

```
SubscribeTrade(ARRAYOFCONST(("PI_XBTUSD")));
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesSubscribed**:

```
void OnKrakenFuturesSubscribed(TObject *Sender, string Feed, string ProductId,)  
{  
    DoLog("#subscribed: " + Feed + " " + ProductId);  
}
```

Anuluj subskrypcję wywołując metodę **UnSubscribeTrade**:

```
UnSubscribeTrade(ARRAYOFCONST(("PI_XBTUSD")));
```

Jeśli anulowanie subskrypcji powiedzie się, zostanie wywołane zdarzenie **OnKrakenFuturesUnSubscribed**:

```
void OnKrakenFuturesUnSubscribed(TObject *Sender, string Feed, string ProductId)  
{  
    DoLog("#unsubscribed: " + Feed + " " + ProductId);  
}
```

W przypadku błędu podczas próby subskrypcji lub anulowania subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesError**.

```
void OnKrakenFuturesError(TObject *Sender, string Error)  
{  
    DoLog("#error: " + Error);  
}
```

Aktualizacje transakcji będą powiadamiane w zdarzeniu **OnKrakenData**.

```
{  "feed": "trade_snapshot",
```

"product\_id": "PI\_XBTUSD",

"trades": [

{

"feed": "trade",

"product\_id": "PI\_XBTUSD",

"uid": "caa9c653-420b-4c24-a9f1-462a054d86f1",

"side": "sell",

"type": "fill",

"seq": 655508,

"time": 1612269657781,

"qty": 440,

"price": 34893

},

{

```
"feed": "trade",

"product_id": "PI_XBTUSD",

"uid": "45ee9737-1877-4682-bc68-e4ef818ef88a",

"side": "sell",

"type": "fill",

"seq": 655507,

"time": 1612269656839,

"qty": 9643,

"price": 34891

}

]

}
```

## Książka

Ten strumień danych zwraca informacje o arkuszu zleceń.

Subskrybuj Arkusz zleceń, wywołując metodę `SubscribeBook` i podając `Symbol`.

```
SubscribeBook(ARRAYOFCONST(("PI_XBTUSD")));
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesSubscribed**:

```
void OnKrakenFuturesSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#subscribed: " + Feed + " " + ProductId);
}
```

Anuluj subskrypcję, wywołując metodę **UnSubscribeBook**:

```
UnSubscribeBook(ARRAYOFCONST(("PI_XBTUSD")));
```

Jeśli anulowanie subskrypcji powiedzie się, zostanie wywołane zdarzenie **OnKrakenFuturesUnSubscribed**:

```
void OnKrakenFuturesUnSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#unsubscribed: " + Feed + " " + ProductId);
}
```

W przypadku błędu podczas próby subskrypcji lub anulowania subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesError**.

```
void OnKrakenFuturesError(TObject *Sender, string Error)
{
    DoLog("#error: " + Error);
}
```

Aktualizacje księgi zleceń będą powiadamiane w zdarzeniu **OnKrakenData**.

```
{
```

```
"feed": "book_snapshot",
```

```
"product_id": "PI_XBTUSD",
```

```
"timestamp": 1612269825817,
```

```
"seq": 326072249,
```

```
"tickSize": null,
```

```
"bids": [{
```

```
"price": 34892.5,
```

"qty": 6385

},

{

"price": 34892,

"qty": 10924

}

],

"asks": [{

"price": 34911.5,

"qty": 20598

},

{

"price": 34912,

"qty": 2300



```
}
```

```
]
```

```
}
```

## Ticker Lite

Kanał informacyjny ticker lite zwraca informacje o kursach notowanych produktów.

Subskrybuj strumień Spread, wywołując metodę **SubscribeTickerLite**.

```
SubscribeTickerLite(ARRAYOFCONST(("PI_XBTUSD")));
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesSubscribed**:

```
void OnKrakenFuturesSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#subscribed: " + Feed + " " + ProductId);
}
```

Anulowanie subskrypcji za pomocą metody **UnSubscribeTickerLite**:

```
UnSubscribeTickerLite(ARRAYOFCONST(("PI_XBTUSD")));
```

Jeśli anulowanie subskrypcji powiedzie się, zostanie wywołane zdarzenie **OnKrakenFuturesUnSubscribed**:

```
void OnKrakenFuturesUnSubscribed(TObject *Sender, string Feed, string ProductId)
{
    DoLog("#unsubscribed: " + Feed + " " + ProductId);
}
```

W przypadku błędu podczas próby subskrypcji lub anulowania subskrypcji zostanie wywołane zdarzenie **OnKrakenFuturesError**.

```
void OnKrakenFuturesError(TObject *Sender, string Error)
{
    DoLog("#error: " + Error);
}
```

Aktualizacje spreadu będą notyfikowane w zdarzeniu **OnKrakenData**.

```
{  "feed": "ticker_lite",
```

```
"product_id": "PI_XBTUSD",
```

"bid": 34932,

"ask": 34949.5,

"change": 3.3705205220015966,

"premium": 0.1,

"volume": 264126741,

"tag": "perpetual",

"pair": "XBT:USD",

"dtm": 0,

"maturityTime": 0

}

{

"feed": "ticker\_lite",

"product\_id": "FI\_ETHUSD\_210625",

"bid": 1753.45,

"ask": 1760.35,

"change":13.448175559936647,

"premium":9.1,

"volume":6899673.0,

"tag":"semiannual",

"pair":"ETH:USD",

"dtm":141,

"maturityTime":1624633200000

}

## HeartBeat

Kanał heartbeat publikuje wiadomości heartbeat w określonych odstępach czasu.

```
SubscribeHeartBeat();
```

```
UnSubscribeHeartBeat();
```

## Zdarzenia

**OnConnect:** wywoływane po połączeniu klienta websocket z serwerem.

**OnKrakenFuturesConnect:** wywoływane po pomyślnym nawiązaniu połączenia WebSocket i gdy serwer wysła status systemowy.

**OnKrakenFuturesSubscribed:** wywoływane po pomyślnym zasubskrybowaniu kanału.

**OnKrakenFuturesUnSubscribed:** wywoływane po pomyślnym anulowaniu subskrypcji kanału.

**OnKrakenFuturesError:** wywoływane w przypadku błędu podczas subskrypcji/anulowania subskrypcji.

**OnKrakenData:** wywoływane za każdym razem, gdy subskrypcja kanału otrzymuje aktualizację.

# API Kraken Futures | Prywatne API WebSocket

## Połączenie

URL: wss://futures.kraken.com/ws/v1

## Uwierzytelnianie

Żądania subskrypcji i anulowania subskrypcji prywatnych kanałów WebSocket wymagają podpisanej wiadomości challenge z kluczem api\_secret użytkownika.

Wyzwanie jest uzyskiwane w sposób pokazany w sekcji WebSocket API Public (przy użyciu api\_key).

Uwierzytelnione żądania muszą zawierać zarówno oryginalną wiadomość wyzwania (original\_challenge), jak i podpisaną wiadomość (signed\_challenge) w formacie JSON.

Aby uzyskać wyzwanie WebSockets, w komponencie opcji Kraken należy ustawić klucz API oraz sekret API; klucz API jest dostarczany przez Kraken na koncie użytkownika.

```
Kraken->ApiKey = "api key";  
Kraken->ApiSecret = "api secret";
```

## Metody

### Otwarte zlecenia — szczegółowy widok

Ten kanał subskrypcji publikuje informacje o otwartych zleceniach użytkownika. Kanał dodaje dodatkowe informacje o wszystkich zleceniach typu post-only, które nie mogły zostać wykonane na rynku.

```
SubscribeOpenOrdersVerbose();
```

Później można anulować subskrypcję OpenOrdersVerbose, wywołując metodę **UnSubscribeOpenOrdersVerbose**

```
UnSubscribeOpenOrdersVerbose();
```

Przykładowa odpowiedź serwera

```
{  
  'feed': 'open_orders_verbose_snapshot',  
  {  
    'instrument': 'PI_XBTUSD',  
    'time': 1567428848005,  
    'last_update': 1567428848005,  
    'account': '0f9c23b8-63e2-40e4-9592-6d5aa57c12',  
    'open_orders': [  
      {  
        'type': 'buy',  
        'size': 0.01,  
        'price': 15674.28848,  
        'status': 'open',  
        'time': 1567428848005,  
        'last_update': 1567428848005,  
        'account': '0f9c23b8-63e2-40e4-9592-6d5aa57c12',  
        'instrument': 'PI_XBTUSD',  
        'order_id': '1567428848005',  
        'order_type': 'limit',  
        'order_status': 'open',  
        'order_type_params': {  
          'limit': 15674.28848,  
          'limit_type': 'limit' }  
      }  
    ]  
  }  
}
```

### Otwarte pozycje

Ten kanał subskrypcji publikuje otwarte pozycje na koncie użytkownika.

```
SubscribeOpenPositions();
```

Później można zrezygnować z subskrypcji OpenPositions, wywołując metodę **UnSubscribeOpenPositions**

```
UnSubscribeOpenPositions();
```

Przykładowa odpowiedź serwera

```
{
  "feed": "open_positions",
  "account": "DemoUser",
  "positions": [{
    "instrument": "fi_xbtusd_180316",
    "balance": 2000.0,
    "entry_price": 11675.86541981,
    "mark_price": 11090.0,
    "index_price": 12290.5500000000001,
    "pnl": -0.00905299
  }]
}
```

## Dziennik konta

Ten kanał subskrypcji publikuje informacje o koncie.

```
SubscribeAccountLog();
```

Później można anulować subskrypcję AccountLog, wywołując metodę **UnSubscribeAccountLog**

```
UnSubscribeAccountLog();
```

Przykładowa odpowiedź serwera

```
{
  'feed': 'account_log_snapshot',
  'logs': [{
    'id': 1690,
    'date': '2019-07-11T08:00:00.000Z',
    'asset': 'bch',
    'info': 'funding
rate change ',
    'booking_uid ': '
86 fdc252 - 1 b6e - 40 ec - ac1d - c7bd46ddeebf ',
    'margin_account ': '
f - bch: usd ',
    'old_balance ': '0.01215667051,',
    'new_balance ': '0.01215736653,',
    'old_average_entry_price ': '0.0,',
    'new_average_entry_price ': '0.0,',
    'trade_price ': '0.0,',
    'mark_price ': '0.0,',
    'realized_pnl ': '0.0,',
    'fee ': '0.0,',
    'execution ': '
',
    'collateral ': '
bch ',
    'funding_rate ': '-8.7002552653e-08,',
    'realized_funding ': '6.9602e-07'}]
}
```

## Realizacje

Ten kanał subskrypcji publikuje informacje o realizacjach.

```
SubscribeFills();
```

Później można anulować subskrypcję Fills, wywołując metodę **UnSubscribeFills**

```
UnSubscribeFills();
```

Przykładowa odpowiedź serwera

```
{  
  
  "feed": "fills_snapshot",  
  
  "account": "DemoUser",  
  
  "fills": [  
  
    {  
  
      "instrument": "FI_XBTUSD_200925",  
  
      "time": 1600256910739,  
  
      "price": 10937.5,  
  
      "seq": 36,  
  
      "buy": true,  
  
      "qty": 5000.0,  
  
      "order_id": "9e30258b-5a98-4002-968a-5b0e149bcfbf",  
  
      "fill_id": "cad76f07-814e-4dc6-8478-7867407b6bff",
```

```
"fill_type":"maker",
```

```
"fee_paid":-0.00009142857,
```

```
"fee_currency":"BTC"
```

```
}]
```

```
}
```

## Otwarte zlecenia

Ten kanał subskrypcji publikuje informacje o otwartych zleceniach użytkownika.

```
SubscribeOpenOrders();
```

Później można anulować subskrypcję OpenOrders, wywołując metodę **UnSubscribeOpenOrders**

```
UnSubscribeOpenOrders();
```

Przykładowa odpowiedź serwera

```
{
```

```
"feed": "open_orders_snapshot",
```

```
"account": "e258dba9-4dd4-4da5-bfef-75beb91c098e",
```

```
"orders": [
```

```
{
```

```
"instrument": "PI_XBTUSD",
```



"time": 1612275024153,

"last\_update\_time": 1612275024153,

"qty": 1000,

"filled": 0,

"limit\_price": 34900,

"stop\_price": 13789,

"type": "stop",

"order\_id": "723ba95f-13b7-418b-8fcf-ab7ba6620555",

"direction": 1,

"reduce\_only": false,

"triggerSignal": "last"

}

]

}

## Saldo konta i depozyty zabezpieczające

Ten kanał subskrypcji zwraca informacje o saldzie i depozycie zabezpieczającym dla konta klienta.

```
SubscribeAccountBalanceAndMargins();
```

Następnie można anulować subskrypcję AccountBalance, wywołując metodę **UnSubscribeAccountBalanceAndMargins**.

```
UnSubscribeAccountBalanceAndMargins();
```

Przykładowa odpowiedź serwera

```
{

"feed": "account_balances_and_margins",

"account": "DemoUser",

"margin_accounts": [

{

"name": "xbt",

"balance": 0,

"pnl": 0,

"funding": 0,

"pv": 0,

"am": 0,
```

"im": 0,

"mm": 0

},

{

"name": "f-xbt:usd",

"balance": 9.99730211055,

"pnl": -0.00006034858674327812,

"funding": 0,

"pv": 9.997241761963258,

"am": 9.99666885201038,

"im": 0.0005729099528781564,

"mm": 0.0002864549764390782

},

],

"seq": 14

```
}
```

## Powiadomienia

Ten kanał subskrypcji publikuje powiadomienia do klienta.

```
SubscribeNotifications();
```

Później można anulować subskrypcję powiadomień, wywołując metodę **UnSubscribeNotifications**

```
UnSubscribeNotifications();
```

Przykładowa odpowiedź serwera

```
{
```

```
"feed":"notifications_auth",
```

```
"notifications":[
```

```
{
```

```
"id":5,
```

```
"type":"market",
```

```
"priority":"low",
```

```
"note":"Nota opisująca powiadomienie.",
```

```
"effective_time":1520288300000
```

```
},
```

...

]

}

# API Kraken Futures | Publiczny REST API

## Połączenie

URL: <https://futures.kraken.com/derivatives/api/v3>

Publiczne API kontraktów Kraken Futures nie wymaga żadnego uwierzytelniania.

## Konfiguracja

Jedyna konfiguracja dotyczy włączenia dziennika dla żądań HTTP REST. Należy włączyć HTTPLogOptions, jeśli zachowanie wszystkich żądań/odpowiedzi HTTP w pliku tekstowym jest pożądane.

## Zdarzenia

**OnKrakenHTTPException:** zdarzenie wywoływane w przypadku wystąpienia wyjątku podczas wykonywania żądania HTTP z REST API.

## Metody

### GetFeeSchedules

Ten punkt końcowy wyświetla listę wszystkich harmonogramów opłat. Uwierzytelnianie nie jest wymagane.

```
KrakenFutures->REST_API->GetFeeSchedules();
```

### Książka zleceń

Ten punkt końcowy zwraca pełną nieskumulowaną księgę zleceń aktualnie notowanych kontraktów Futures.

```
KrakenFutures->REST_API->GetOrderBook("PI_XBTUSD");
```

### Notowania

Ten punkt końcowy zwraca aktualne dane rynkowe dla wszystkich aktualnie notowanych kontraktów Futures i indeksów.

```
KrakenFutures->REST_API->GetTickers();
```

## Instrumenty

Ten punkt końcowy zwraca specyfikacje wszystkich aktualnie notowanych kontraktów futures i indeksów.

```
KrakenFutures->REST_API->GetInstruments();
```

## Historia

Ten punkt końcowy zwraca ostatnie 100 transakcji od podanej wartości lastTime — jeśli nie podano wartości, zwraca ostatnie 100 transakcji. Punkt końcowy zwraca historię transakcji za maksymalnie 7 dni od momentu wywołania lub od ostatniego wydania silnika transakcyjnego (w zależności od tego, które nastąpi wcześniej).

```
KrakenFutures->REST_API->GetHistory("PI_XBTUSD");
```

# API Kraken Futures | Prywatne API REST

## Połączenie

URL: <https://futures.kraken.com/derivatives/api/v3>

## Uwierzytelnianie

Prywatne REST API wymaga klucza API i sekretu API; wartości te są dostarczane przez Kraken na koncie użytkownika.

```
Kraken->ApiKey = "api key";  
Kraken->ApiSecret = "api secret";
```

## Metody

### EditOrderById

Ten punkt końcowy umożliwia edytowanie istniejącego zlecenia dla aktualnie notowanego kontraktu Futures.

**aOrderId:** identyfikator zlecenia, które ma zostać edytowane.

**aSize:** Rozmiar powiązany ze zleceniem

**aLimitPrice:** Cena limitowa powiązana ze zleceniem.

**aStopPrice:** Cena stop powiązana ze zleceniem stop. Wymagana, jeśli poprzedni typ zlecenia to Stop.

```
KrakenFutures->REST_API->EditOrderById("Order_Id", 2, 1000);
```

### EditOrderByCliOrderId

Ten punkt końcowy umożliwia edytowanie istniejącego zlecenia dla aktualnie notowanego kontraktu Futures.

**aCliOrderId:** Identyfikator zlecenia określony przez użytkownika. Musi być globalnie unikatowy.

**aSize:** Rozmiar powiązany ze zleceniem

**aLimitPrice:** Cena limitowa powiązana ze zleceniem.

**aStopPrice:** Cena stop powiązana ze zleceniem stop. Wymagane, jeśli typem zlecenia jest Stop.



```
KrakenFutures->REST_API->EditOrderByCliOrderId("Cli_Order_Id", 2, 1000);
```

## SendMarketOrder

Ten punkt końcowy umożliwia wysłanie zlecenia rynkowego.

**aSide:** Kierunek zlecenia: kupno lub sprzedaż.

**aSymbol:** Symbol kontraktów futures

**aSize:** Rozmiar powiązany ze zleceniem.

```
KrakenFutures->REST_API->SendMarketOrder(kosfBuy, "PI_XBTUSD", 1);
```

## SendLimitOrder

Ten punkt końcowy umożliwia wysłanie zlecenia Limit.

**aSide:** Kierunek zlecenia: kupno lub sprzedaż.

**aSymbol:** Symbol kontraktów futures

**aSize:** Rozmiar powiązany ze zleceniem.

**aLimitPrice:** Cena limitowa powiązana ze zleceniem.

```
KrakenFutures->REST_API->SendLimitOrder(kosfBuy, "PI_XBTUSD", 1, 1000);
```

## SendStopOrder

Ten punkt końcowy umożliwia złożenie zlecenia Stop.

**aSide:** Kierunek zlecenia: kupno lub sprzedaż.

**aSymbol:** Symbol kontraktów futures

**aSize:** Rozmiar powiązany ze zleceniem.

**aStopPrice:** Cena stop powiązana ze zleceniem stop.

**aLimitPrice:** Cena limitowa powiązana ze zleceniem.

```
KrakenFutures->REST_API->SendStopOrder(kosfBuy, "PI_XBTUSD", 1, 1000, 900);
```

## SendTakeProfitOrder

Ten punkt końcowy umożliwia wysłanie zlecenia Take Profit.

**aSide:** Kierunek zlecenia: kupno lub sprzedaż.

**aSymbol:** Symbol kontraktów futures

**aSize:** Rozmiar powiązany ze zleceniem.

**aStopPrice:** Cena stop powiązana ze zleceniem stop.

**aLimitPrice:** Cena limitowa powiązana ze zleceniem.

```
KrakenFutures->REST_API->SendTakeProfitOrder(kosfBuy, "PI_XBTUSD", 1, 1000, 900);
```

## SendOrder

Ten punkt końcowy umożliwia złożenie zlecenia limit, stop, take profit lub immediate-or-cancel dla aktualnie notowanego kontraktu Futures.

**OrderType:** wybierz jeden z następujących: kotfLMT, kotfPOST, kotfMKT, kotfSTP, kotfTAKE\_PROFIT, kotfIOC

**Symbol:** Symbol kontraktu futures

**Side:** Kierunek zlecenia (kupno lub sprzedaż).

**Size:** Rozmiar powiązany ze zleceniem.

**StopPrice:** Cena stop powiązana ze zleceniem stop.

**LimitPrice:** Cena limitu powiązana ze zleceniem.

**TriggerSignal:** W przypadku zlecenia Stop lub TakeProfit sygnał używany do wyzwolenia; należy wybrać jedną z wartości: kotsMark, kotsIndex, kotsLast

**CliOrderId:** Tożsamość zlecenia określona przez użytkownika. Musi być globalnie unikalna.

**ReduceOnly:** Należy ustawić na true, jeśli zlecenie ma jedynie zmniejszyć istniejącą pozycję. Każde zlecenie zwiększające istniejącą pozycję zostanie odrzucone. Domyślnie false.

```
TsgcHTTPKrakenFuturesOrder *oOrder = new TsgcHTTPKrakenFuturesOrder(this);
try
{
    oOrder->Side = kosfBuy;
    oOrder->Symbol = "PI_XBTUSD";
    oOrder->OrderType = kotfMKT;
    oOrder->Size = 1;
    KrakenFutures->REST_API->SendOrder(oOrder);
}
__finally
{
    oOrder->Free;
}
```

## CancelOrderByOrderId

Ten punkt końcowy umożliwia anulowanie otwartego zlecenia dla kontraktu Futures.

**aOrderId:** identyfikator zlecenia, które ma zostać edytowane.

```
KrakenFutures->REST_API->CancelOrderByOrderId("Order_Id");
```

## CancelOrderByCliOrderId

Ten punkt końcowy umożliwia anulowanie otwartego zlecenia dla kontraktu Futures.

**aCliOrderId:** Identyfikator zlecenia określony przez użytkownika. Musi być globalnie unikatowy.

```
KrakenFutures->REST_API->CancelOrderByCliOrderId("Cli_Order_Id");
```

## GetFills

Ten punkt końcowy zwraca informacje o zrealizowanych zleceniach dla wszystkich kontraktów futures.

**aLastFillDate:** Jeśli nie podano, zwracane jest ostatnie 100 realizacji w dowolnym kontrakcie futures. Jeśli podano, zwracane jest 100 wpisów przed czasem lastFillTime.

```
KrakenFutures->REST_API->GetFills("2020-07-22T13:45:00.000Z");
```

## Transfer

Ten punkt końcowy umożliwia przenoszenie środków między dwoma kontami z depozytem zabezpieczającym o tej samej walucie zabezpieczenia lub między kontem z depozytem zabezpieczającym a kontem gotówkowym.

**aFromAccount:** Nazwa konta gotówkowego lub depozytowego, z którego mają zostać przeniesione środki.

**aToAccount:** Nazwa konta gotówkowego lub marżowego, na które mają być przeniesione środki.

**aUnit:** Jednostka do przeniesienia.

**aAmount:** Kwota do transferu.

```
KrakenFutures->REST_API->Transfer("FI_XBTUSD", "cash", "xbt", 1.5);
```

## GetOpenPositions

Ten punkt końcowy zwraca rozmiar i średnią cenę wejścia wszystkich otwartych pozycji w kontraktach Futures. Obejmuje to kontrakty Futures, które wygasły, ale nie zostały jeszcze rozliczone.

```
KrakenFutures->REST_API->GetOpenPositions();
```

## GetNotifications

Ten punkt końcowy udostępnia powiadomienia platformy.

```
KrakenFutures->REST_API->GetNotifications();
```

## GetAccounts

Ten punkt końcowy zwraca kluczowe informacje dotyczące wszystkich kont Kraken Futures, które mogą być kontami gotówkowymi lub kontami depozytowymi. Obejmuje salda aktywów cyfrowych, salda instrumentów, wymagania dotyczące depozytu zabezpieczającego, szacunki wyzwalaczy depozytu oraz informacje pomocnicze takie jak dostępne środki, PnL otwartych pozycji i wartość portfela.

```
KrakenFutures->REST_API->GetAccounts();
```

## CancelAllOrders

Ten punkt końcowy umożliwia anulowanie otwartego zlecenia dla kontraktu Futures.

**Symbol:** Kontrakt futures, dla którego mają zostać anulowane wszystkie otwarte zlecenia (opcjonalne)

```
KrakenFutures->REST_API->CancelAllOrders();
```

## CancelAllOrdersAfter

Ten punkt końcowy zapewnia mechanizm Dead Man's Switch chroniący klienta przed awariami sieci. Klient może wysłać żądanie z limitem czasu w sekundach, co uruchomi odliczanie, które po upływie tego czasu anuluje wszystkie zlecenia klienta.

**aTimeout:** Limit czasu wyrażony w sekundach.

```
KrakenFutures->REST_API->CancelAllOrdersAfter(60);
```

## GetOpenOrders

Ten punkt końcowy zwraca informacje o wszystkich otwartych zleceniach dla wszystkich kontraktów Futures.

```
KrakenFutures->REST_API->OpenOrders();
```

## GetHistoricalOrders

Ten punkt końcowy zwraca historyczne zlecenia złożone na koncie.

**aSince:** Data i godzina od

**aBefore:** Data/czas przed

**aSort:** „asc” dla sortowania rosnącego, „desc” dla sortowania malejącego

**aContinuationToken:** token kontynuacji dostarczony z poprzedniej odpowiedzi, który można wykorzystać w wywołaniu w celu zwrócenia następnego zestawu dostępnych wyników

```
KrakenFutures->REST_API->GetHistoricalOrders(Now, Now - 5);
```

## GetHistoricalTriggers

Ten endpoint zwraca historyczne wyzwalacze dla konta.

**aSince:** Data i godzina od

**aBefore:** Data/czas przed

**aSort:** „asc” dla sortowania rosnącego, „desc” dla sortowania malejącego

**aContinuationToken:** token kontynuacji dostarczony z poprzedniej odpowiedzi, który można wykorzystać w wywołaniu w celu zwrócenia następnego zestawu dostępnych wyników

```
KrakenFutures->REST_API->GetHistoricalTriggers(Now, Now - 5);
```

## GetHistoricalExecutions

Ten punkt końcowy zwraca historyczne wykonania na koncie.

**aSince:** Data i godzina od

**aBefore:** Data/czas przed

**aSort:** „asc” dla sortowania rosnącego, „desc” dla sortowania malejącego

**aContinuationToken:** token kontynuacji dostarczony z poprzedniej odpowiedzi, który można wykorzystać w wywołaniu w celu zwrócenia następnego zestawu dostępnych wyników

```
KrakenFutures->REST_API->GetHistoricalExecutions(Now, Now - 5);
```

## **WithdrawalToSpotWallet**

Ten punkt końcowy umożliwia złożenie wniosku o wypłatę aktywów cyfrowych z portfela Kraken Futures do portfela Kraken Spot.

**aCurrency:** zasób cyfrowy, który ma zostać wypłacony, np. xbt lub xrp.

**aAmount:** kwota waluty, która ma zostać wypłacona.

```
KrakenFutures->REST_API->WithdrawalToSpotWallet("xbt", 1000);
```

## **GetFeeScheduleVolumes**

Ten punkt końcowy zwraca 30-dniowy wolumen w USD.

```
KrakenFutures->REST_API->GetFeeScheduleVolumes();
```

## **GetAccountLogCSV**

Ten punkt końcowy umożliwia klientom pobranie pliku CSV z dziennikami kont.

```
KrakenFutures->REST_API->GetAccountLogCSV();
```

# API Pusher

## Pusher

Pusher to łatwa i niezawodna platforma z przydatnymi funkcjami opartymi na protokole WebSocket: elastyczna obsługa wiadomości pub/sub, listy użytkowników na żywo (presence), uwierzytelnianie i wiele innych.

Pusher WebSocket API ma wersję 7.

Dane są przesyłane dwukierunkowo przez WebSocket jako dane tekstowe zawierające JSON zakodowany w UTF-8 (binarne ramki WebSocket nie są obsługiwane).

Można wywołać metodę **Ping**, aby przetestować połączenie z serwerem. Zasadniczo każda wiadomość odebrana od drugiej strony jest traktowana jako potwierdzenie aktywności połączenia. W przypadku braku wiadomości każda ze stron może sprawdzić, czy druga strona odpowiada, wysyłając wiadomość ping, na którą druga strona powinna odpowiedzieć wiadomością pong.

Przed nawiązaniem połączenia należy wypełnić następujące pola:

```
Pusher->Cluster = "eu"; // cluster where your pusher account is located
Pusher->Key = "9c3b7ef25qe97a00116c"; // your pusher api key
Pusher->Name = "js"; // optional, name of your application
Pusher->Version = "4.1"; // optional, version of your application
Pusher->TLS = true; // if encrypted, set to true
Pusher->Secret = "2dc792e1916ac49e6b3f"; // pusher secret string (needed for private and presence channels)
```

## Ważne

Pusher wymaga, aby klient WebSocket łączył się z adresem URL zbudowanym na podstawie wcześniejszych pól (klucz, klaster...). Pola te są używane do budowania adresu URL i dzieje się to w momencie przypisania klienta do komponentu Pusher. Dlatego, aby upewnić się, że adres URL jest prawidłowo zbudowany, należy ustawić klienta po wypełnieniu pól konfiguracyjnych Pusher. Poniżej znajduje się pseudokod:

```
// configure pusher fields

pusher.cluster = ...

pusher.key = ...

// ustaw klienta

pusher.client = websocket client

// start connection

websocket client.Active = true;
```

Po pomyślnym nawiązaniu połączenia wywoływane jest zdarzenie **OnPusherConnect**, które udostępnia następujące pola:

- Socket ID: Unikalny identyfikator podłączonego klienta.
- Timeout: Liczba sekund bezczynności serwera, po upływie których klient powinien zainicjować wiadomość ping (obsługiwane automatycznie przez komponent).

W przypadku błędu zostanie wywołane zdarzenie **OnPusherError** wraz z informacjami o błędzie. Błąd może zostać wysłany przez Pusher w odpowiedzi na nieprawidłowe uwierzytelnienie, nieprawidłowe polecenie itp.

#### **4000-4099**

Wskazuje błąd powodujący zamknięcie połączenia przez Pusher, przy czym próba ponownego połączenia z tymi samymi parametrami nie powiedzie się.

4000: Aplikacja akceptuje tylko połączenia SSL, należy ponownie połączyć używając wss://

4001: Aplikacja nie istnieje

4003: Aplikacja wyłączona

4004: Aplikacja przekroczyła limit połączeń

4005: Ścieżka nie znaleziona

4006: Nieprawidłowy format ciągu wersji

4007: Nieobsługiwana wersja protokołu

4008: Nie podano wersji protokołu

#### **4100-4199**

Oznacza błąd powodujący zamknięcie połączenia przez Pusher; klient może ponownie nawiązać połączenie po upływie co najmniej 1 sekundy.

4100: Przekroczono pojemność

#### **4200-4299**

Wskazuje błąd skutkujący zamknięciem połączenia przez Pusher i informuje, że klient może natychmiast ponownie nawiązać połączenie.

4200: Natychmiastowe ponowne połączenie ogólne

4201: Nie otrzymano odpowiedzi Pong: do klienta wysłano Ping, ale nie odebrano odpowiedzi — patrz wiadomości ping i pong.

4202: Zamknięto z powodu braku aktywności: Klient był nieaktywny przez długi czas (obecnie 24 godziny), a klient nie obsługuje polecenia ping. Należy zaktualizować do nowszej wersji roboczej WebSocket lub zaimplementować wersję 5 lub wyższą tego protokołu.

#### **4300-4399**

Każdy inny rodzaj błędu.

4301: Zdarzenie klienta odrzucone z powodu ograniczenia przepustowości

## **Kanały**

Kanały są fundamentalną koncepcją w Pusher. Każda aplikacja posiada wiele kanałów, a każdy klient może wybierać, które kanały subskrybuje.



Kanały zapewniają:

- Sposób filtrowania danych. Na przykład w aplikacji czatu może istnieć kanał dla osób chcących dyskutować o „psach”.
- Sposób kontrolowania dostępu do różnych strumieni informacji. Na przykład aplikacja do zarządzania projektami może chcieć autoryzować użytkowników do otrzymywania aktualizacji dotyczących 'projectX'

Zdecydowanie zaleca się filtrowanie danych przy użyciu kanałów, a nie zdarzeń. Wynika to z faktu, że wszystkie zdarzenia publikowane na kanale są wysyłane do wszystkich subskrybentów, niezależnie od ich powiązań ze zdarzeniami.

Kanały nie wymagają jawnego tworzenia i są tworzone na żądanie klienta. Oznacza to, że tworzenie kanału jest proste. Wystarczy poinformować klienta, aby subskrybował dany kanał.

Obsługiwane są następujące typy kanałów:

- **Kanały publiczne** mogą subskrybować wszyscy, którzy znają ich nazwę
- **Kanały prywatne** wprowadzają mechanizm pozwalający serwerowi kontrolować dostęp do rozgłaszanych danych
- **Kanały presence** są rozszerzeniem kanałów prywatnych. Umożliwiają rejestrowanie informacji o użytkowniku przy subskrypcji oraz informowanie innych członków kanału o tym, kto jest online.
- **Kanały pamięci podręcznej** zapamiętują ostatnie wywołane zdarzenie i wysyłają je jako pierwsze zdarzenie do nowych subskrybentów (warianty publiczne, prywatne i presence)
- **Prywatne kanały szyfrowane** zapewniają szyfrowanie end-to-end przy użyciu NaCl secretbox, gwarantując, że nawet Pusher nie może odczytać danych wiadomości.

## Kanały publiczne

Kanały publiczne powinny być używane do publicznie dostępnych danych, ponieważ nie wymagają żadnej formy autoryzacji w celu subskrypcji.

Można subskrybować i anulować subskrypcję kanałów w dowolnym momencie. Nie ma potrzeby czekania na zakończenie łączenia przez Pusher.

**Przykład:** subskrypcja kanału „my-channel”.

```
CBuilder  
APIPusher->Subscribe("my-channel");
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnPusherSubscribe**; w przypadku błędu wiadomość zostanie przekazana za pośrednictwem zdarzenia **OnPusherError**.

Wszystkie wiadomości z subskrybowanego kanału będą odbierane przez zdarzenie **OnPusherEvent**.

Gdy wywoływana jest metoda Publish i kanał jest publiczny, komponent zamiast protokołu WebSocket używa protokołu HTTP i wywołuje metodę TriggerEvent (publikowanie przez protokół WebSocket nie jest dozwolone).

## Kanały prywatne

*Wymaga Indy w wersji 10.5.7 lub nowszej*

Kanały prywatne powinny być używane, gdy dostęp do kanału musi być w jakiś sposób ograniczony. Aby użytkownik mógł subskrybować kanał prywatny, wymagana jest autoryzacja.

**Przykład:** subskrybowanie kanału „my-private-channel”.

**CBuilder**

```
APIPusher->Subscribe("my-private-channel", pscPrivateChannel);
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnPusherSubscribe**; w przypadku błędu wiadomość zostanie przekazana za pośrednictwem zdarzenia **OnPusherError**.

Wszystkie wiadomości z subskrybowanego kanału będą odbierane przez zdarzenie **OnPusherEvent**.

## Kanały obecności

*Wymaga Indy w wersji 10.5.7 lub nowszej*

Kanały Presence opierają się na bezpieczeństwie kanałów prywatnych i udostępniają dodatkową funkcję informowania o tym, kto jest subskrybentem danego kanału. Umożliwia to niezwykle łatwe tworzenie czatów i funkcji „kto jest online” w aplikacji. Doskonale sprawdza się w pokojach czatowych, przy współpracy nad dokumentem, przeglądaniu tej samej strony internetowej czy udziale w grze wieloosobowej.

Kanały obecności są subskrybowane z poziomu API klienta w taki sam sposób jak kanały prywatne, lecz nazwa kanału musi być poprzedzona prefiksem presence-. Podobnie jak w przypadku kanałów prywatnych, żądanie HTTP jest wysyłane do konfigurowalnego adresu URL uwierzytelniania w celu sprawdzenia, czy bieżący użytkownik ma uprawnienia dostępu do kanału.

Informacje o użytkownikach subskrybujących i anulujących subskrypcję kanału są dostępne poprzez powiązanie ze zdarzeniami na kanale presence, a bieżący stan użytkowników subskrybujących kanał jest dostępny za pomocą właściwości channel.members.

**Przykład:** subskrypcja kanału "my-presence-channel”.

```
APIPusher->Subscribe("my-presence-channel", pscPresenceChannel,  
    '{"user_id":"'John_Smith'", "user_info":{"name":"'John Smith'"}}');
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnPusherSubscribe**; w przypadku błędu wiadomość zostanie przekazana za pośrednictwem zdarzenia **OnPusherError**.

Wszystkie wiadomości z subskrybowanego kanału będą odbierane przez zdarzenie **OnPusherEvent**.

## Kanały pamięci podręcznej

Kanał pamięci podręcznej zapamiętuje ostatnie wywołane zdarzenie i wysyła je jako pierwsze zdarzenie do nowych subskrybentów.

Gdy zdarzenie jest wywoływane na kanale pamięci podręcznej, Pusher Channels zapisuje to zdarzenie w pamięci, a gdy klient subskrybuje kanał pamięci podręcznej, jeżeli istnieje wartość w pamięci podręcznej,

jest ona wysyłana do klienta jako pierwsze zdarzenie na tym kanale. To zachowanie pozwala programistom dostarczać stan początkowy bez dodawania dodatkowej logiki pobierania danych z innych źródeł.

Obsługiwane są następujące kanały pamięci podręcznej:

- Publiczny kanał pamięci podręcznej
- Prywatny kanał pamięci podręcznej
- Kanał pamięci podręcznej Presence

**Przykład:** subskrypcja publicznego kanału pamięci podręcznej "my-cache-channel".

```
APIPusher->Subscribe("my-cache-channel", pscCacheChannel);
```

W przypadku pomyślnej subskrypcji zostanie wywołane zdarzenie **OnPusherSubscribe**; w przypadku błędu wiadomość zostanie przekazana za pośrednictwem zdarzenia **OnPusherError**.

Wszystkie wiadomości z subskrybowanego kanału będą odbierane przez zdarzenie **OnPusherEvent**.

Jeśli podczas subskrybowania kanału cache nie istnieje żadne zbuforowane zdarzenie, zostanie wywołane zdarzenie **OnPusherCacheMiss** z podaniem nazwy kanału. Umożliwia to aplikacji obsługę sytuacji, w której żadne dane z pamięci podręcznej nie są dostępne.

## Prywatne zaszyfrowane kanały

Kanały prywatne szyfrowane zapewniają szyfrowanie end-to-end wiadomości. Podobnie jak kanały prywatne wymagają uwierzytelniania, ale dodatkowo wszystkie ładunki danych są szyfrowane przy użyciu NaCl secretbox, dzięki czemu tylko autoryzowani subskrybenci mogą odczytać treść. Nawet Pusher nie może odszyfrować wiadomości.

Aby korzystać z prywatnych zaszyfrowanych kanałów, podczas uwierzytelniania należy podać **SharedSecret**. Wspólny sekret jest używany do szyfrowania i deszyfrowania danych wiadomości.

**Przykład:** subskrybuj prywatny zaszyfrowany kanał "my-encrypted-channel".

```
APIPusher->Subscribe("my-encrypted-channel", pscPrivateEncryptedChannel);
```

Dostępny jest również wariant prywatny z szyfrowaniem i zachowaniem kanału buforującego:

```
APIPusher->Subscribe("my-encrypted-cache-channel", pscPrivateEncryptedCacheChannel);
```

Podczas używania zdarzenia **OnPusherAuthentication** z prywatnymi kanałami szyfrowanymi można ustawić właściwość **SharedSecret** w obiekcie odpowiedzi, aby podać klucz szyfrowania:

```
void OnPusherAuthenticationEvent(TObject *Sender,  
    TsgcWSPusherRequestAuthentication *AuthRequest,  
    TsgcWSPusherResponseAuthentication *AuthResponse)  
{  
    AuthResponse->SharedSecret = "your-shared-secret-key";  
}
```

## Zdarzenia Presence

Kanały obecności udostępniają dodatkowe zdarzenia powiadamiające aplikację, gdy użytkownicy dołączają do kanału lub go opuszczają, oraz umożliwiają śledzenie liczby subskrypcji.

## OnPusherMemberAdded

Wywoływane, gdy nowy członek subskrybuje kanał presence. Udostępnia nazwę kanału, identyfikator użytkownika oraz informacje o użytkowniku, który dołączył.

```
void PUSHERPusherMemberAdded(TObject *Sender,
    String Channel, String UserId, String UserInfo)
{
    Log("Member joined: " + UserId + " on " + Channel);
}
```

## OnPusherMemberRemoved

Wywoływane, gdy członek anuluje subskrypcję kanału obecności. Dostarcza nazwę kanału, identyfikator użytkownika i informacje o użytkowniku, który opuścił kanał.

```
void PUSHERPusherMemberRemoved(TObject *Sender,
    String Channel, String UserId, String UserInfo)
{
    Log("Member left: " + UserId + " on " + Channel);
}
```

## OnPusherSubscriptionCount

Wywoływane, gdy liczba subskrybentów kanału ulega zmianie. Dostarcza nazwę kanału i bieżącą liczbę subskrybentów. Zdarzenie to musi być włączone w panelu Pusher.

```
void PUSHERPusherSubscriptionCount(TObject *Sender,
    String Channel, int SubscriptionCount)
{
    Log(Channel + " has " + IntToStr(SubscriptionCount) + " subscribers");
}
```

## OnPusherCacheMiss

Wywoływane przy subskrypcji kanału pamięci podręcznej, który nie zawiera buforowanego zdarzenia. Dostarcza nazwę kanału. Umożliwia to aplikacji obsługę sytuacji, gdy nie są dostępne żadne buforowane dane — na przykład przez pobranie danych z innego źródła.

```
void PUSHERPusherCacheMiss(TObject *Sender, String Channel)
{
    Log("Cache miss on: " + Channel);
}
```

## Publikowanie wiadomości

Nie tylko można odbierać wiadomości z subskrybowanych kanałów, ale również wysyłać wiadomości do innych subskrybowanych użytkowników.

Należy wywołać metodę **Publish**, aby wysłać wiadomość do wszystkich subskrybentów kanału.

Przykład: wysyłanie zdarzenia do wszystkich subskrybentów kanału "my-channel"

```
APIPusher->Publish("my-event", "my-channel");
```

Publikuj nie więcej niż 10 wiadomości na sekundę na klienta (połączenie). Wszystkie zdarzenia wywołane powyżej tego limitu szybkości zostaną odrzucone przez Pusher API. Nie jest to problem systemowy, lecz problem po stronie klienta. 100 klientów w kanale wysyłających wiadomości z taką szybkością musiałoby każdorazowo przetwarzać 1000 wiadomości na sekundę! Choć niektóre nowoczesne przeglądarki mogłyby sobie z tym poradzić, prawdopodobnie nie jest to dobry pomysł.

## REST API

API jest hostowane pod adresem `http://api-CLUSTER.pusher.com`, gdzie CLUSTER należy zastąpić własnym klastrem aplikacji (np. eu).

Kody stanu HTTP służą do informowania o powodzeniu lub niepowodzeniu żądań. Poniżej przedstawiono typowe kody:

**200** Żądanie zostało zrealizowane pomyślnie. Treść odpowiedzi będzie zawierać skrót JSON z danymi odpowiedzi.

**400** Błąd: szczegóły w treści odpowiedzi

**401** Błąd uwierzytelniania: treść odpowiedzi będzie zawierać wyjaśnienie

**403** Forbidden: aplikacja wyłączona lub przekroczony limit wiadomości

Zaimplementowano następujące funkcje REST API.

|                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| TriggerEvent             | Wyzwała nowe zdarzenie na określonym kanale. Obsługuje opcjonalne parametry SocketId (wykluczenie klienta) i Info. |
| TriggerBatchEvents       | Wyzwała wiele zdarzeń w jednym żądaniu HTTP. Przyjmuje tablicę JSON obiektów zdarzeń.                              |
| GetChannels              | Udostępnia listę wszystkich aktywnych kanałów. Obsługuje opcjonalne parametry FilterByPrefix i Info.               |
| GetChannel               | Dostarcza informacje o określonym kanale. Obsługuje opcjonalny parametr Info.                                      |
| GetUsers                 | Dostarcza listę wszystkich użytkowników połączonych z kanałem.                                                     |
| TerminateUserConnections | Kończy wszystkie połączenia dla danego użytkownika według jego identyfikatora.                                     |

### TriggerEvent

Wyzwała zdarzenie na jednym lub kilku kanałach. Wymaga nazwy zdarzenia, nazwy kanału i ładunku danych.

|            |                                                       |
|------------|-------------------------------------------------------|
| aEventName | Nazwa zdarzenia do wywołania.                         |
| aChannel   | Nazwa kanału, na którym ma zostać wywołane zdarzenie. |
| aData      | Dane zdarzenia (ciąg JSON).                           |

|                           |                                                                                                                                                  |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| aSocketId<br>(opcjonalny) | Identyfikator gniazda, który ma zostać wykluczony z odbierania zdarzenia. Przydatne, aby zapobiec otrzymywaniu przez nadawcę własnej wiadomości. |
| aInfo (opcjonalne)        | Rozdzielana przecinkami lista atrybutów do uwzględnienia w odpowiedzi (np. "subscription_count").                                                |

```
// trigger event on a channel
APIPusher->TriggerEvent("my-event", "my-channel", "Hello World");

// trigger event excluding the sender
APIPusher->TriggerEvent("my-event", "my-channel", "Hello World", "123.456");

// trigger event requesting subscription_count in the response
APIPusher->TriggerEvent("my-event", "my-channel", "Hello World", "", "subscription_count");
```

## TriggerBatchEvents

Wyzwała wiele zdarzeń w jednym wywołaniu API, co jest wydajniejsze niż wykonywanie osobnych żądań dla każdego zdarzenia. Parametr batch musi być ciągiem JSON zawierającym tablicę obiektów zdarzeń, gdzie każdy obiekt posiada pola „channel”, „name” i „data”.

```
APIPusher->TriggerBatchEvents(
    "[{\"channel\":\"my-channel\", \"name\":\"my-event\", \"data\":\"hello\"}, {\"channel\":\"my-channel-2\", \"name\":\"my-event\", \"data\":\"world\"}]");
```

## GetChannels

Zwraca listę aktywnych kanałów. Obsługuje opcjonalne parametry umożliwiające filtrowanie wyników i żądanie dodatkowych informacji.

|                                  |                                                                                                |
|----------------------------------|------------------------------------------------------------------------------------------------|
| aFilterByPrefix<br>(opcjonalnie) | Filtruj kanały według prefiksu nazwy (np. "presence-", aby wyświetlić tylko kanały obecności). |
| aInfo (opcjonalne)               | Rozdzielona przecinkami lista atrybutów do uwzględnienia w odpowiedzi (np. "user_count").      |

```
// get all channels
APIPusher->GetChannels();

// get only presence channels with user count
APIPusher->GetChannels("presence-", "user_count");
```

## GetChannel

Zwraca informacje o określonym kanale.

|          |                                                        |
|----------|--------------------------------------------------------|
| aChannel | Nazwa kanału, o którym mają zostać pobrane informacje. |
|----------|--------------------------------------------------------|

|                    |                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------|
| aInfo (opcjonalne) | Rozdzielana przecinkami lista atrybutów do uwzględnienia (np. "user_count,subscription_count"). |
|--------------------|-------------------------------------------------------------------------------------------------|

```
// get channel info
APIPusher->GetChannel("presence-my-channel");

// get channel info with user count
APIPusher->GetChannel("presence-my-channel", "user_count,subscription_count");
```

## GetUsers

Zwraca listę użytkowników połączonych z kanałem presence. Nazwa kanału musi zawierać pełny prefiks (np. "presence-my-channel").

```
APIPusher->GetUsers("presence-my-channel");
```

## TerminateUserConnections

Kończy wszystkie połączenia ustanowione przez danego użytkownika. Może być używane do wymuszenia rozłączenia konkretnego użytkownika ze wszystkich kanałów. Identyfikator użytkownika musi odpowiadać identyfikatorowi "user\_id" użytemu podczas subskrybowania kanału obecności.

```
APIPusher->TerminateUserConnections("1234");
```

## Uwierzytelnianie niestandardowe

Pusher zezwala wyłącznie na subskrybowanie prywatnych kanałów lub kanałów obecności. Jeśli połączenie dostarcza token uwierzytelniający, umożliwia to ograniczenie dostępu.

Można zbudować własny przepływ uwierzytelniania przy użyciu zdarzenia **OnPusherAuthentication**. Zdarzenie jest wywoływane przed podpisaniem wiadomości subskrypcji tajnym kluczem dostarczonym przez Pusher. Zdarzenie ma 2 parametry: żądanie uwierzytelnienia z polami takimi jak SocketId, nazwa kanału itp., które mogą być używane przez własny serwer uwierzytelniania do autoryzacji lub odrzucenia żądania. Poniżej znajduje się zrzut ekranu przedstawiający przepływ uwierzytelniania Pusher

Gdy klient łączy się z serwerem pusher, wysyła klucz dostarczony przez pusher, a serwer zwraca identyfikator (socket\_id).

Gdy klient subskrybuje prywatny (lub presence) kanał, klient sgcWebSockets używa klucza Secret Key dostarczonego przez pusher w celu utworzenia podpisu, który jest dołączany do wiadomości subskrypcji. Korzystając ze zdarzenia OnPusherAutentication, można przechwycić pola wymagane do podpisania wiadomości, zaimplementować własne metody uwierzytelniania i, w przypadku powodzenia, zwrócić podpis, który zostanie dołączony do wiadomości subskrypcji i przesłany do serwera.

### **Przykład:**

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Pusher *oPusher = new TsgcWSAPI_Pusher(NULL);
oPusher->Client = oClient;
oPusher->Cluster = "eu";
Pusher->Name = "js";
Pusher->Version = "4.1";
```



```

Pusher->TLS = true;
Pusher->Key = "9c3b7ef25qe97a00116c";
Pusher->Secret = ""; // the secret key is not known by the client, only by the authentication module

oPusher->OnPusherAuthentication = OnPusherAuthenticationEvent;

private void OnPusherAuthenticationEvent(TObject *Sender, TsgcWSPusherRequestAuthentication
    *AuthRequest, TsgcWSPusherResponseAuthentication *AuthResponse)
{
    // if the authentication request is successful return the signature
    if (CustomAuthentication(AuthRequest->Channel, AuthRequest->SocketID))
    {
        AuthResponse->Signature = GetCustomAuthenticationSignature();
    }
}

```

Format podpisu jest następujący:

**Kanały prywatne:** key:HMAC256(SocketID, ChannelName)

**Kanały Presence:** klucz: HMAC256(SocketID, ChannelName, Data)

Obiekt **TsgcWSPusherResponseAuthentication** udostępnia następujące właściwości:

|              |                                                                                                                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sekret       | Tajny klucz Pusher używany do obliczenia sygnatury HMAC. Wstępnie wypełniany wartością Pusher.Secret, jeśli jest skonfigurowana.                                                                                      |
| Podpis       | Obliczony podpis uwierzytelniający. Jeśli pozostawiono puste, komponent obliczy go automatycznie przy użyciu klucza Secret.                                                                                           |
| SharedSecret | Wspólny klucz tajny dla prywatnych kanałów szyfrowanych. Wymagany podczas subskrybowania kanałów pscPrivateEncryptedChannel lub pscPrivateEncryptedCacheChannel. Używany do szyfrowania end-to-end danych wiadomości. |

# API Bitmex

## Bitmex

Bitmex to giełda kryptowalut i platforma handlu instrumentami pochodnymi.

Obsługiwane są następujące API:

1. **Strumienie WebSocket:** pozwalają subskrybować określone metody i otrzymywać dane w czasie rzeczywistym. Zdarzenia są wypychane do klientów przez serwer do subskrybentów. Używa protokołu WebSocket.
2. **REST API:** klienci mogą żądać danych rynkowych i konta od serwera. Wymaga klucza API i sekretu do uwierzytelniania; protokołem jest HTTPs.

## Właściwości

API Bitmex ma 2 typy metod: publiczne i prywatne. Do metod publicznych można uzyskać dostęp bez uwierzytelniania, na przykład: pobieranie cen tickerów. Niektóre są prywatne i dotyczą danych użytkownika; te metody wymagają użycia kluczy API Bitmex.

- **ApiKey:** nowy klucz API można wygenerować na koncie Bitmex; wystarczy skopiować jego wartość do tej właściwości.
- **ApiSecret:** sekret API, który należy trzymać w bezpiecznym miejscu.
- **TestNet:** jeśli włączone, połączy się z kontem demonstracyjnym Bitmex (domyślnie false).
- **HTTPLogOptions:** przechowuje dziennik żądań HTTP w pliku tekstowym
  - **Enabled:** jeśli włączone, będzie zapisywać wszystkie żądania HTTP WebSocket API.
  - **FileName:** pełna ścieżka do pliku, w którym będą przechowywane dzienniki

## Najczęstsze zastosowania

- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować kanał WebSocket](#)
- **REST API**
  - [How to Place a Bitmex Order](#)

## WebSocket API

[Subskrybuj](#) / [Anuluj subskrypcję](#)

BitMEX umożliwia subskrybowanie danych w czasie rzeczywistym. Dostęp ten nie jest ograniczony szybkością po nawiązaniu połączenia i jest najlepszym sposobem na dostarczanie aktualnych danych do programów. W niektórych tematach można przekazać symbol, aby filtrować zdarzenia według symbolu, na przykład: trades, quotes...

Następujące tematy subskrypcji są dostępne bez uwierzytelniania:

- **btmAnnouncement:** Ogłoszenia serwisu
- **btmChat:** czat Trollbox
- **btmConnected:** Statystyki połączonych użytkowników/botów
- **btmFunding:** Aktualizacje stóp finansowania swapów. Wysyłane co każdy interwał finansowania (zazwyczaj co 8 godzin)
- **btmInstrument:** Aktualizacje instrumentu, w tym obrót oraz ceny bid/ask
- **btmInsurance:** Dienne aktualizacje funduszu ubezpieczeniowego
- **btmLiquidation:** Zlecenia likwidacyjne w momencie wprowadzenia do księgi zleceń
- **btmOrderBookL2\_25:** 25 najwyższych poziomów księgi zleceń poziomu 2
- **btmOrderBookL2:** pełna księga zleceń poziomu 2
- **btmOrderBook10:** 10 najlepszych poziomów przy użyciu tradycyjnego pełnego przesyłania książki zleceń
- **btmPublicNotifications:** Powiadomienia systemowe (używane dla krótkotrwałych wiadomości)
- **btmQuote:** Najwyższy poziom arkusza zleceń
- **btmQuoteBin1m:** 1-minutowe koszyki kwotowań
- **btmQuoteBin5m:** 5-minutowe pojemniki na wyceny
- **btmQuoteBin1h:** godzinowe pojemniki kwotowania
- **btmQuoteBin1d:** Przedziały kwotowań 1-dniowych
- **btmSettlement:** Rozliczenia
- **btmTrade:** transakcje na żywo
- **btmTradeBin1m:** minutowe koszyki transakcji
- **btmTradeBin5m:** 5-minutowe koszyki transakcji
- **btmTradeBin1h:** koszyki transakcji godzinnych

- **btmTradeBin1d**: jednodniowe przedziały transakcji

Następujące tematy wymagają uwierzytelnienia:

- **btmAffiliate**: Status partnera, np. łączna liczba poleconych użytkowników i procent prowizji.
- **btmExecution**: poszczególne realizacje; może być ich wiele na jedno zlecenie
- **btmOrder**: Aktualizacje na żywo dotyczące zleceń
- **btmMargin**: Aktualizacje dotyczące bieżącego salda konta i wymagań dotyczących depozytu zabezpieczającego
- **btmPosition**: Aktualizacje dotyczące pozycji
- **btmPrivateNotifications**: Powiadomienia indywidualne — aktualnie nieużywane
- **btmTransact**: aktualizacje wpłat/wypłat
- **btmWallet**: Dane salda adresu Bitcoin, w tym łączne wpłaty i wypłaty

Przykład odebranych wiadomości:

```
{
  "table": "orderBookL2_25",
  "keys": ["symbol", "id", "side"],
  "types": {"id": "long", "price": "float", "side": "symbol", "size": "long", "symbol": "symbol"},
  "foreignKeys": {"side": "side", "symbol": "instrument"},
  "attributes": {"id": "sorted", "symbol": "grouped"},
  "action": "partial",
  "data": [
    {"symbol": "XBTUSD", "id": "17999992000", "side": "Sell", "size": "100", "price": "80"},
    {"symbol": "XBTUSD", "id": "17999993000", "side": "Sell", "size": "20", "price": "70"},
    {"symbol": "XBTUSD", "id": "17999994000", "side": "Sell", "size": "10", "price": "60"},
    {"symbol": "XBTUSD", "id": "17999995000", "side": "Buy", "size": "10", "price": "50"},
    {"symbol": "XBTUSD", "id": "17999996000", "side": "Buy", "size": "20", "price": "40"},
    {"symbol": "XBTUSD", "id": "17999997000", "side": "Buy", "size": "100", "price": "30"}
  ]
}

{
  "table": "orderBookL2_25",
  "action": "update",
  "data": [
    {"symbol": "XBTUSD", "id": "17999995000", "side": "Buy", "size": "5"}
  ]
}

{
  "table": "orderBookL2_25",
  "action": "delete",
```

```

    "data": [
      { "symbol": "XBTUSD", "id": 17999995000, "side": "Buy" }
    ]
  }
  {
    "table": "orderBookL2_25",
    "action": "insert",
    "data": [
      { "symbol": "XBTUSD", "id": 17999995500, "side": "Buy", "size": 10, "price": 45 },
    ]
  }
}

```

## Uwierzytelnianie

Aby subskrybować strumień zablokowane dla użytkownika, należy najpierw przeprowadzić uwierzytelnianie. Nieprawidłowe uwierzytelnianie spowoduje zamknięcie połączenia.

Korzystanie z BitMEX API wymaga klucza API.

Trwałe klucze API mogą być blokowane do zakresów adresów IP i odwoływane w dowolnym momencie bez narażania głównych danych uwierzytelniających. Nie wymagają też odnawiania.

Aby korzystać z uwierzytelniania kluczem API, należy wygenerować klucz API na koncie.

Przed subskrybowaniem jakiegokolwiek tematu wymagającego uwierzytelnienia należy wywołać metodę **Authenticate**.

## CancelAllAfter (Dead Man's Switch)

Metoda **CancelAllAfter** implementuje funkcję Dead Man's Switch. Po wywołaniu z wartością limitu czasu (w milisekundach) instruuje serwer, aby anulował wszystkie otwarte zlecenia, jeśli w ciągu tego czasu nie zostanie odebrane kolejne wywołanie **CancelAllAfter**. Jest to przydatne do zapewnienia anulowania zleceń w przypadku rozłączenia sieciowego.

## REST API

|                           |                                                                                                                                                                                               |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetExecutions             | Zwraca wszystkie surowe transakcje, w tym otwieranie i anulowanie zleceń oraz zmiany statusu zleceń.                                                                                          |
| GetExecutionsTradeHistory | Zwraca bardziej szczegółowe transakcje.                                                                                                                                                       |
| GetInstruments            | Zwraca wszystkie instrumenty i indeksy, w tym te, które zostały rozliczone lub są nienabyte. Należy używać tego punktu końcowego do zapytań o poszczególne instrumenty lub złożonych filtrów. |
| GetOrders                 | Aby pobierać tylko otwarte zlecenia                                                                                                                                                           |

|                              |                                                                                                     |
|------------------------------|-----------------------------------------------------------------------------------------------------|
| PlaceOrder                   | Składa zlecenie nieprzetworzone przy użyciu obiektu TsgcHTTPBitmexOrder.                            |
| PlaceMarketOrder             | Złóż nowe zlecenie MARKET.                                                                          |
| PlaceLimitOrder              | Złóż nowe zlecenie LIMIT.                                                                           |
| PlaceStopOrder               | Złóż nowe zlecenie STOP.                                                                            |
| PlaceStopLimitOrder          | Złóż nowe zlecenie STOPLIMIT.                                                                       |
| AmendOrder                   | Modyfikacja istniejącego zlecenia.                                                                  |
| CancelOrder                  | Anuluje aktywne zlecenie.                                                                           |
| CancelAllOrders              | Anuluj wszystkie aktywne zlecenia.                                                                  |
| CancelAllOrdersAfter         | Anuluj wszystkie zlecenia po pewnym czasie.                                                         |
| ClosePosition                | Zamknij otwartą pozycję.                                                                            |
| GetOrderBook                 | Pobierz bieżącą księgę zleceń w formacie pionowym                                                   |
| GetPosition                  | Pobierz swoje pozycje.                                                                              |
| SetPositionIsolate           | Włącz izolowany margin lub cross-margin dla każdej pozycji.                                         |
| SetPositionLeverage          | Wybierz dźwignię finansową per pozycja.                                                             |
| SetPositionRiskLimit         | Zaktualizuj swój limit ryzyka.                                                                      |
| SetPositionTransferMargin    | Przenieś kapitał własny do lub z pozycji.                                                           |
| GetQuotes                    | Pobieranie kursów                                                                                   |
| GetTrades                    | Pobierz transakcje                                                                                  |
| GetFunding                   | Pobierz dane dotyczące finansowania.                                                                |
| GetInsurance                 | Pobierz dane funduszu ubezpieczeniowego.                                                            |
| GetTradeBucketed             | Pobierz dane transakcji z przedziałami czasowymi (OHLCV) z konfigurowalnymi rozmiarami przedziałów. |
| GetQuoteBucketed             | Pobierz podzielone na kubelki dane notowań z konfigurowalnymi rozmiarami przedziałów.               |
| GetSettlement                | Pobieranie danych rozliczeniowych.                                                                  |
| GetLiquidation               | Pobierz zlecenia likwidacyjne.                                                                      |
| GetInstrumentIndices         | Pobierz indeksy instrumentów.                                                                       |
| GetInstrumentComposite-Index | Pobierz dane złożonego indeksu dla instrumentów.                                                    |
| GetStats                     | Pobieranie statystyk dla całej giełdy.                                                              |
| GetStatsHistory              | Pobierz historyczne statystyki giełdy.                                                              |
| GetStatsHistoryUSD           | Pobierz historyczne statystyki kursów wymiany USD.                                                  |
| GetUserMargin                | Pobierz dane dotyczące depozytu zabezpieczającego konta.                                            |
| GetUserWallet                | Pobierz informacje o swoim portfelu.                                                                |
| GetUserWalletHistory         | Pobierz historię transakcji portfela.                                                               |
| GetUserWalletSummary         | Pobierz podsumowanie swojego portfela.                                                              |

# Bitmex | Połączenie z WebSocket API

Aby połączyć się z Bitmex WebSocket API, należy utworzyć nowego klienta Bitmex API i dołączyć go do TsgcWebSocketClient.

Poniżej znajduje się przykład:

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex->Client = oClient;
oClient->Active = true;
```

# Bitmex | Subskrypcja kanału WebSocket

Bitmex oferuje wiele kanałów, za pomocą których można subskrybować aktualizacje danych rynkowych, zleceń i innych informacji w czasie rzeczywistym. Poniżej znajduje się przykład subskrypcji kanału handlowego:

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex->Client = oClient;
oBitmex->Subscribe(btmTrade, "XBTUSD");
void OnBitmexMessage(Sender: TObject; const aTopic: TwsBitmexTopics; const aMessage: string)
{
    // here you will receive the trade updates
}
```



# Bitmex | Jak składać zlecenia

Bitmex REST API oferuje publiczne i prywatne punkty końcowe. Prywatne punkty końcowe wymagają podpisywania wiadomości w celu zwiększenia bezpieczeństwa transakcji.

Najpierw należy zalogować się na konto Bitmex i utworzyć nowe API, po czym zostaną udostępnione następujące wartości:

- ApiKey
- ApiSecret

Pola te należy skonfigurować we właściwości Bitmex komponentu klienta API Bitmex. Po skonfigurowaniu można rozpocząć wysyłanie prywatnych żądań do REST API Bitmex.

## Typy zleceń

Wszystkie zlecenia wymagają symbolu. Wszystkie pozostałe pola są opcjonalne, chyba że zaznaczono inaczej.

Poniżej podano prawidłowe wartości ordType:

- **Limit:** Domyślny typ zlecenia. Należy podać orderQty i cenę.
- **Market:** Tradycyjne zlecenie rynkowe. Zlecenie rynkowe jest realizowane do momentu wypełnienia lub osiągnięcia ceny bankructwa, po czym zostaje anulowane.
- **Stop:** Zlecenie Stop Market. Należy podać wartości orderQty i stopPx. Po osiągnięciu stopPx zlecenie zostanie wpisane do arkusza.
  - W zleceniach sprzedaży zlecenie wyzwala się, gdy cena wyzwalająca jest niższa niż stopPx. W zleceniach kupna — gdy jest wyższa.
  - Uwaga: zlecenia Stop nie zajmują marży do momentu wyzwolenia. Należy upewnić się, że wymagana marża jest dostępna na koncie, aby zlecenie mogło zostać w pełni zrealizowane.
  - Zlecenia Close Stop nie wymagają podania orderQty. Więcej informacji w sekcji Instrukcje realizacji poniżej.
- **StopLimit:** podobne do Stop Market, lecz zamiast zlecenia rynkowego składa zlecenie Limit. Należy podać orderQty, stopPx i price.
- **MarketIfTouched:** Podobne do zlecenia Stop, ale wyzwalacze działają w przeciwnym kierunku. Przydatne dla zleceń Take Profit.
- **LimitIfTouched:** Jak powyżej; stosowane dla zleceń Take Profit Limit.
- **Pegged:** Zlecenia powiązane umożliwiają użytkownikom składanie zlecenia z limitem ceny względem bieżącej ceny rynkowej. Należy określić pegPriceType i pegOffsetValue.
  - Zlecenia powiązane muszą mieć execInst ustawione na Fixed. Oznacza to, że cena limitu jest ustalana w momencie przyjęcia zlecenia i nie zmienia się wraz ze zmianą ceny referencyjnej.
  - PrimaryPeg: Cena jest ustawiana względem ceny bliskiej strony rynku.
  - MarketPeg: Cena jest ustawiana względem ceny dalekiej strony rynku.
  - Wartość pegPriceType przesłana bez ordType jest traktowana jako zlecenie powiązane.

## Instrukcje wykonania

Obsługiwane są następujące instrukcje `execInst`. W przypadku podania kilku wartości należy oddzielić je przecinkiem (np. `LastPrice,Close`).

- **ParticipateDoNotInitiate:** Znane również jako zlecenie Post-Only. Jeśli zlecenie zostałoby wykonane w momencie złożenia, zostanie anulowane. Ma to na celu ochronę przed sytuacją, gdy cena rynkowa przesunie się w stronę zlecenia podczas jego transmisji. Nie jest to przeznaczone do spekulowania na oddaleniu się ceny po złożeniu zlecenia — takie zachowanie jest uznawane za nadużycie i podlega monitorowaniu.
- **MarkPrice, LastPrice, IndexPrice:** Używane przez zlecenia stop i if-touched do określenia ceny wyzwalającej. Należy użyć tylko jednej. Domyślnie używana jest `MarkPrice`. Używane również dla zleceń Pegged do zdefiniowania wartości `LastPeg`.
- **ReduceOnly:** Zlecenie `ReduceOnly` może jedynie zmniejszyć pozycję, a nie ją zwiększyć. Jeśli w księdze zleceń znajduje się oczekujące zlecenie `ReduceOnly` z limitem, a pozycja zostanie zmniejszona przez inne zlecenia, ilość tego zlecenia zostanie odpowiednio zmniejszona lub anulowana. W przypadku wielu zleceń `ReduceOnly` jako pierwsze zostanie zmienione najmniej agresywne.
- **Close:** Zamknięcie implikuje `ReduceOnly`. Zlecenie `Close` anuluje inne aktywne zlecenia z limitem ceny po tej samej stronie i instrumencie, jeśli otwarta ilość przekracza bieżącą pozycję. Jest to przydatne w przypadku zleceń stop: anulując te zlecenia, zlecenie `Close Stop` ma zapewniony wymagany depozyt zabezpieczający do realizacji i może zostać zrealizowane maksymalnie do pełnej wartości pozycji. Jeśli `orderQty` nie jest podana, zlecenie `Close` ma `orderQty` równe bieżącej wielkości pozycji.
  - Należy pamiętać, że zlecenie `Close` bez `orderQty` wymaga podania strony, aby BitMEX wiedział, czy powinno być wyzwolone powyżej, czy poniżej `stopPx`.
- **LastWithinMark:** Używany przez zlecenia stop z `LastPrice`, aby umożliwić wyzwalanie stop tylko gdy:
  - Dla Sell Stop Market / Stop Limit Order
    - $\text{Last Price} \leq \text{Stop Price}$
    - $\text{Last Price} \geq \text{Mark Price} \times (1 - 5\%)$
  - Dla Buy Stop Market / Stop Limit Order:
    - $\text{Last Price} \geq \text{Stop Price}$
    - $\text{Last Price} \leq \text{Mark Price} \times (1 + 5\%)$
- **Fixed:** Zlecenia zakotwiczone muszą mieć wartość `execInst` równą `Fixed`. Oznacza to, że cena limit jest ustawiana w momencie przyjęcia zlecenia i nie zmienia się wraz ze zmianą ceny referencyjnej.

## Zlecenia pegged

Zlecenia pegged umożliwiają użytkownikom składanie zlecenia z limitem ceny względnym wobec bieżącej ceny rynkowej. Cena graniczna jest ustalana jednorazowo w momencie składania zlecenia i nie zmienia się wraz ze zmianą ceny referencyjnej. Ten typ zlecenia nie jest przeznaczony do spekulacji na oddaleniu się ceny po złożeniu — takie zachowanie jest traktowane jako nadużycie i podlega monitorowaniu.

Zlecenia Pegged mają `ordType` równy `Pegged` i `execInst` równy `Fixed`.

Należy również podać `pegPriceType` i `pegOffsetValue`:

- `PrimaryPeg` — cena jest ustawiana względem ceny najbliższego dotyku
- `MarketPeg` — cena jest ustalana względem ceny far touch

## Zlecenia trailing stop z powiązaniem ceny

Użyj `pegPriceType` o wartości `TrailingStopPeg`, aby tworzyć zlecenia Trailing Stop.

Cena jest ustalana w momencie złożenia zlecenia i aktualizowana raz na sekundę, jeśli cena bazowa (ostatnia/mark/indeks) zmieniła się o więcej niż 0,1%. Wartość `stopPx` przesuwa się wraz z ruchem rynku od punktu `peg` i zamraża, gdy rynek zbliża się ku niemu.

Należy użyć `pegOffsetValue` do ustawienia `stopPx` zlecenia. Peg jest ustawiany na cenę wyzwalającą określoną w `execInst` (domyślnie `MarkPrice`). Należy użyć ujemnego offsetu dla zleceń stop-sell i buy-if-touched.

Wymaga parametru `ordType`: `Stop`, `StopLimit`, `MarketIfTouched`, `LimitIfTouched`.

### Trailing Stops

Można użyć `pegPriceType` o wartości „`TrailingStopPeg`”, aby tworzyć zlecenia Trailing Stop. Zakotwiczony `stopPx` przesuwa się zgodnie z ruchem rynku od punktu zakotwiczenia i zatrzymuje się, gdy rynek porusza się w jego kierunku.

Aby używać tej opcji, należy połączyć ją z wartością `pegOffsetValue` w celu ustawienia `stopPx` zlecenia. Kołek jest ustawiany na cenę wyzwalacza określoną w `execInst` (domyślnie `'MarkPrice'`). Dla zleceń stop-sell i buy-if-touched należy użyć ujemnego przesunięcia.

Wymaga `ordType`: `'Stop'`, `'StopLimit'`, `'MarketIfTouched'`, `'LimitIfTouched'`.

### Śledzenie zleceń

Jeśli chcesz samodzielnie śledzić identyfikatory zleceń, ustaw unikalny `clOrdID` dla każdego zlecenia. `clOrdID` ten będzie zwracany jako właściwość zlecenia oraz powiązanych wykonań (w tym przez `WebSocket`) i może być używany do pobrania lub anulowania zlecenia. Maksymalna długość wynosi 36 znaków.

### Przykłady:

```
// buy market order
BITMEX->REST_API->PlaceMarketOrder(bmosBuy, "XBTUSD", 100);
// sell limit order at 45000
BITMEX->REST_API->PlaceLimitOrder(bmosSell, "XBTUSD", 100, 45000.00);
// stop order at 48000
BITMEX->REST_API->PlaceStopOrder(bmosSell, "XBTUSD", 100, 48000.00);
```

# API Bitfinex

## Bitfinex

Bitfinex jest jedną z największych i najbardziej zaawansowanych platform do handlu kryptowalutami na świecie. Użytkownicy mogą wymieniać Bitcoin, Ethereum, Ripple, EOS, Bitcoin Cash, Iota, NEO, Litecoin, Ethereum Classic...

Wersja Bitfinex WebSocket API to 2.0

Każda wiadomość wysyłana i odbierana przez kanał WebSocket Bitfinex jest kodowana w formacie JSON

Symbol może być parą handlową lub walutą depozytu zabezpieczającego:

- Symbole par handlowych tworzone są przez dodanie przedrostka "t" przed parą (np. tBTCUSD, tETHUSD).
- Symbole walut na depozycie zabezpieczającym tworzone są przez dodanie przedrostka "f" przed walutą (np. fUSD, fBTC, ...)

Po pomyślnym nawiązaniu połączenia wywoływane jest zdarzenie **OnBitfinexConnect** i jako parametr przekazywany jest numer wersji Bitfinex API.

Można wywołać metodę **Ping**, aby przetestować połączenie z serwerem.

Jeśli serwer wyśle jakiegolwiek informacje, można je obsłużyć za pomocą zdarzenia **OnBitfinexInfoMessage**, w którym Code i Message są parametrami zawierającymi informacje o wiadomości wysłanej przez serwer. Przykładowe kody:

20051 : Zatrzymaj/Uruchom ponownie serwer WebSocket (proszę ponownie nawiązać połączenie)

20060: Przechodzenie w tryb konserwacji. Należy wstrzymać wszelką aktywność i wznowić ją po otrzymaniu komunikatu informacyjnego 20061 (oczekiwany czas to maksymalnie 120 sekund).

20061 : Konserwacja zakończona. Można wznowić normalną aktywność. Zaleca się ponowne subskrybowanie wszystkich kanałów.

W przypadku błędu zostanie wywołane zdarzenie **OnBitfinexError** z informacjami o błędzie. Przykładowe kody błędów:

10000 : Nieznane zdarzenie

10001 : Nieznana para

Aby zmienić konfigurację, należy wywołać metodę **Configuration** i przekazać jako parametr jedną z następujących flag:

CS\_DEC\_S = 8; // Włącz wszystkie liczby dziesiętne jako ciągi znaków.

CS\_TIME\_S = 32; // Włącz wszystkie czasy jako ciągi dat.

CS\_SEQ\_ALL = 65536; // Włącz sekwencjonowanie FUNKCJA BETA

CHECKSUM = 131072; // Włącza sumę kontrolną dla każdej iteracji księgi. Sprawdza 25 najwyższych wpisów dla każdej strony księgi. Suma kontrolna jest liczbą całkowitą ze znakiem.

## Subskrybuj kanały publiczne

Istnieją kanały publiczne, dla których uwierzytelnianie wobec serwera nie jest wymagane. Wszystkie wiadomości są przekazywane przez zdarzenie **OnBitfinexUpdate**.

## SubscribeTicker

Ticker jest ogólnym przeglądem stanu rynku. Pokazuje bieżącą najlepszą ofertę kupna i sprzedaży, a także cenę ostatniej transakcji. Zawiera również informacje, takie jak dzienny wolumen obrotu i o ile zmieniła się cena w ciągu ostatniego dnia.

```
// Trading pairs
[
  CHANNEL_ID,
  [
    BID,
    BID_SIZE,
    ASK,
    ASK_SIZE,
    DAILY_CHANGE,
    DAILY_CHANGE_PERC,
```

```

        LAST_PRICE,
        VOLUME,
        HIGH,
        LOW
    ]
]
// Funding pairs
[
    CHANNEL_ID,
    [
        FRR,
        BID,
        BID_PERIOD,
        BID_SIZE,
        ASK,
        ASK_PERIOD,
        ASK_SIZE,
        DAILY_CHANGE,
        DAILY_CHANGE_PERC,
        LAST_PRICE,
        VOLUME,
        HIGH,
        LOW
    ]
]

```

## Subscribe Trades

Ten kanał wysyła wiadomość o transakcji za każdym razem, gdy na Bitfinex dochodzi do zawarcia transakcji. Zawiera wszystkie istotne szczegóły transakcji, takie jak cena, rozmiar i czas.

```

// on trading pairs (ex. tBTCUSD)
[
    CHANNEL_ID,
    [
        [
            ID,
            MTS,
            AMOUNT,
            PRICE
        ],
        ...
    ]
]
// on funding currencies (ex. fUSD)
[
    CHANNEL_ID,
    [
        [
            ID,
            MTS,
            AMOUNT,
            RATE,
            PERIOD
        ],
        ...
    ]
]

```

```
]
]
```

## SubscribeOrderBook

Kanał Order Books umożliwia śledzenie stanu arkusza zleceń Bitfinex. Dane są udostępniane w formie zagregowanej według ceny, z konfigurowalną precyzją. Po otrzymaniu odpowiedzi przesyłany jest obraz aktualnego stanu arkusza, a następnie aktualizacje przy każdej zmianie.

```
// on trading pairs (ex. tBTCUSD)
[
  CHANNEL_ID,
  [
    [
      PRICE,
      COUNT,
      AMOUNT
    ],
    ...
  ]
]

// on funding currencies (ex. fUSD)
[
  CHANNEL_ID,
  [
    [
      RATE,
      PERIOD,
      COUNT,
      AMOUNT
    ],
    ...
  ]
]
```

## SubscribeRawOrderBook

Są to najbardziej szczegółowe arkusze zamówień.

```
// on trading pairs (ex. tBTCUSD)
[
  CHANNEL_ID,
  [
    [
      ORDER_ID,
      PRICE,
      AMOUNT
    ],
    ...
  ]
]
```

```
// on funding currencies (ex. fUSD)
[
  CHANNEL_ID,
  [
    [
      OFFER_ID,
      PERIOD,
      RATE,
      AMOUNT
    ],
    ...
  ]
]
```

## SubscribeCandles

Zapewnia dostęp do informacji o świecach wykresu. Przedziały czasowe:

1m: jedna minuta

5m : pięć minut

15m : 15 minut

30m: 30 minut

1h: jedna godzina

3h : 3 godziny

6h : 6 godzin

12h : 12 godzin

1D: jeden dzień

7D : jeden tydzień

14D : dwa tygodnie

1M : jeden miesiąc

```
[
  CHANNEL_ID,
  [
    [
      MTS,
      OPEN,
      CLOSE,
      HIGH,
      LOW,
      VOLUME
    ],
    ...
  ]
]
```



] ]

## Subskrybuj uwierzytelnione kanały

Ten kanał umożliwia śledzenie stanu konta. Można otrzymywać aktualizacje dotyczące pozycji, sald, zleceń i transakcji.

Należy użyć metody **Authenticate**, aby uwierzytelnić się względem serwera i ustawić wymagane parametry.

Po uwierzytelnieniu będą przesyłane aktualizacje dotyczące: zleceń, pozycji, transakcji, ofert finansowania, kredytów finansowania, pożyczek finansowania, portfeli, informacji o saldzie, informacji o depozycie, informacji o finansowaniu, transakcji finansowania...

Można wywołać metodę **UnAuthenticate**, aby wylogować się z serwera.

# API Kucoin

## Kucoin

Kucoin to międzynarodowa, wielojęzyczna giełda kryptowalut. Udostępnia szereg API umożliwiających dostęp do danych Kucoin. Obsługiwane są następujące API:

1. **Strumienie WebSocket:** pozwalają subskrybować określone metody i otrzymywać dane w czasie rzeczywistym. Zdarzenia są wypychane do klientów przez serwer do subskrybentów. Używa protokołu WebSocket.
2. **REST API:** klienci mogą żądać danych rynkowych i danych konta od serwera. Wymaga klucza API, sekretu i hasła do uwierzytelnienia i używa protokołu HTTPS.

## Właściwości

Kucoin API posiada 2 typy metod: publiczne i prywatne. Metody publiczne są dostępne bez uwierzytelniania, na przykład pobieranie cen ticker. Metody prywatne związane z danymi użytkownika wymagają użycia kluczy API Kucoin.

- **ApiKey:** można wygenerować nowy klucz API na koncie kucoin i skopiować jego wartość do tej właściwości.
- **ApiSecret:** ApiSecret jest wymagany tylko dla REST\_API; websocket API wymaga ApiKey jedynie dla niektórych metod.
- **Passphrase:** ciąg znaków wymagany do połączenia z serwerami Kucoin.
- **Sandbox:** jeśli włączony, połączy się z kontem demonstracyjnym Kucoin (domyślnie false).
  - **HTTPLogOptions:** przechowuje dziennik żądań HTTP w pliku tekstowym
    - **Enabled:** jeśli włączone, będzie przechowywać wszystkie żądania HTTP WebSocket API.
    - **FileName:** pełna ścieżka pliku, w którym będą przechowywane dzienniki
  - **REST:** przechowuje dziennik żądań REST API w pliku tekstowym
    - **Enabled:** jeśli włączone, będzie przechowywać wszystkie żądania HTTP REST API.
    - **FileName:** pełna ścieżka pliku, w którym będą przechowywane dzienniki.

## Najczęstsze zastosowania

- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować kanał WebSocket](#)
- **REST API**
  - [Jak pobierać dane rynkowe](#)
  - [Jak korzystać z prywatnego REST API](#)
  - [Jak handlować spot](#)
  - [Czas prywatnych żądań](#)

## WebSocket Feed

Aby subskrybować wiadomości kanału z określonego serwera, strona klienta powinna wysłać wiadomość subskrypcji do serwera.

Jeśli subskrypcja zakończy się powodzeniem, system wyśle do Ciebie wiadomości ack, gdy odpowiedź będzie ustawiona na true.

```
{
  "id": "1545910660739",
  "type": "ack"
}
```

Dopóki generowane są wiadomości dotyczące tematu, system wysyła odpowiednie wiadomości do strony klienta.

Obsługiwane są następujące metody subskrypcji i anulowania subskrypcji.

### Kanały publiczne

|                              |        |                                                                                                                                                                                                                                                                                                             |
|------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SubscribeSymbolTicker        | Symbol | Zasubskrybuj ten temat, aby otrzymywać powiadomienia push o zmianach BBO. Jeśli w ciągu jednej sekundy nie nastąpi żadna zmiana, powiadomienie nie zostanie wysłane. Dane są wysyłane co 100 ms z aktualnym BBO. Jeśli dane nie zmieniły się w stosunku do poprzednich, powiadomienie nie zostanie wysłane. |
| SubscribeAllSymbolsTicker    |        | Subskrypcja tego tematu pozwala otrzymywać powiadomienia push o zmianach BBO wszystkich symboli rynkowych.                                                                                                                                                                                                  |
| SubscribeSymbolSnapshot      | Symbol | Subskrybuj dane migawkowe dla jednego symbolu. Dane migawkowe są przesyłane w odstępach 2-sekundowych.                                                                                                                                                                                                      |
| SubscribeMarketSnapshot      | Rynek  | Subskrybuj ten temat, aby otrzymać dane migawkowe dla całego rynku. Dane migawkowe są przesyłane co 2 sekundy.                                                                                                                                                                                              |
| SubscribeLevel2MarketData    | Symbol | Subskrybuj ten temat, aby otrzymywać dane z arkusza zleceń poziomu 2. Po pomyślnym nawiązaniu subskrypcji WebSocket system będzie przysyłał do Ciebie przyrostowe dane zmian za pośrednictwem WebSocket.                                                                                                    |
| SubscribeLevel2_5BestAskBid  | Symbol | System zwróci dane 5 najlepszych zleceń ask/bid, które stanowią dane migawkowe aktualizowane co 100 milisekund (innymi słowy, dane 5 najlepszych zleceń ask/bid są zwracane co 100 milisekund w czasie rzeczywistym).                                                                                       |
| SubscribeLevel2_50BestAskBid | Symbol | System zwróci dane 50 najlepszych zleceń ask/bid, będące migawką danych co 100 milisekund (innymi słowy, dane 50 najlepszych zleceń ask/bid zwracane co 100 milisekund w czasie rzeczywistym).                                                                                                              |
| SubscribeKlines              | Symbol | Subskrybuj ten temat, aby otrzymywać dane K-Line.                                                                                                                                                                                                                                                           |
| SubscribeMatchExecutionData  | Symbol | Subskrybuj ten temat, aby otrzymywać pasujący przepływ danych zdarzeń Poziomu 3. Dla każdej zrealizowanej transakcji system wysyła wiadomości dopasowania w następującym formacie.                                                                                                                          |
| SubscribeIndexPrice          | Symbol | Subskrybuj ten temat, aby otrzymywać cenę indeksową dla handlu z depozytem zabezpieczającym.                                                                                                                                                                                                                |
| SubscribeMarkPrice           | Symbol | Subskrybuj ten temat, aby otrzymywać ceny mark price dla handlu na marży.                                                                                                                                                                                                                                   |

|                           |        |                                                                                                    |
|---------------------------|--------|----------------------------------------------------------------------------------------------------|
| SubscribeOrderBookChanged | Symbol | Subskrybuj ten temat, aby otrzymywać zmiany w księdze zleceń handlu na depozycie zabezpieczającym. |
| SubscribeLevel1           | Symbol | Subskrybowanie danych najlepszego bid/ask poziomu 1 dla danego symbolu.                            |

Jeśli parametr ACK jest ustawiony na true, po pomyślnej subskrypcji lub anulowaniu subskrypcji klient otrzymuje o tym wiadomość.

## Kanały prywatne

Wymaga ważnego ApiKey uzyskanego z konta Kucoin. ApiKey, ApiSecret i Passphrase muszą być ustawione we właściwości Kucoin komponentu klienta API.

Następujące dane są wypychane do klienta za każdym razem, gdy nastąpi zmiana. Nie jest konieczna subskrypcja żadnej metody — odbywa się ona automatycznie po ustawieniu prawidłowego ApiKey.

|                                 |                                                                                                                                                                               |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SubscribeTradeOrders            | Ten temat przekazuje wszystkie zdarzenia zmian dotyczące zamówień użytkownika.                                                                                                |
| SubscribeAccountBalance         | Ta wiadomość jest odbierana, gdy zmienia się saldo konta. Zawiera szczegóły dotyczące zmiany.                                                                                 |
| SubscribePositionStatus         | System wyśle zdarzenie zmiany, gdy zmieni się status pozycji.                                                                                                                 |
| SubscribeMarginTradeOrders      | System wyśle tę wiadomość do pożyczkodawców, gdy zlecenie zostanie dodane do arkusza zleceń.                                                                                  |
| SubscribeStopOrder              | Gdy zlecenie stop zostanie odebrane przez system, otrzymasz wiadomość z typem "open". Oznacza to, że zlecenie zostało wprowadzone do systemu i oczekuje na wyzwolenie.        |
| SubscribeTradeOrdersV2          | Subskrybuj kanał zleceń handlowych V2 w celu otrzymywania rozszerzonych powiadomień o aktualizacjach zleceń.                                                                  |
| SubscribeCrossMarginPosition    | Subskrybuje aktualizacje pozycji na marżę krzyżową. System wyśle zdarzenie zmiany, gdy zmieni się pozycja na marżę krzyżową.                                                  |
| SubscribeIsolatedMarginPosition | Subskrybuj aktualizacje izolowanej pozycji na depozycie zabezpieczającym. System wyśle zdarzenie zmiany, gdy izolowana pozycja na depozycie zabezpieczającym ulegnie zmianie. |

## REST API

Wszystkie punkty końcowe zwracają obiekt JSON lub tablicę.

## Publiczne punkty końcowe API

Te punkty końcowe są dostępne bez żadnej autoryzacji.

## Ogólne punkty końcowe

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| GetServiceStatus | Sprawdzenie łączności z Rest API i pobranie statusu usługi    |
| GetServerTime    | Sprawdź połączenie z Rest API i pobierz bieżący czas serwera. |

## Punkty końcowe danych rynkowych

|                     |        |                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetSymbolList       | Rynek  | Wyślij żądanie do tego punktu końcowego, aby uzyskać listę dostępnych par walutowych do handlu. Jeśli potrzebne są informacje rynkowe dotyczące symbolu transakcji                                                                                                                                                                                                                              |
| GetTicker           | Symbol | Wyślij żądanie do tego punktu końcowego, aby uzyskać dane rynkowe poziomu 1. Zwracana wartość obejmuje najlepszą cenę i wielkość oferty kupna, najlepszą cenę i wielkość oferty sprzedaży oraz ostatnią cenę i wielkość transakcji.                                                                                                                                                             |
| GetAllTickers       |        | Żądaj tykera rynkowego dla wszystkich par handlowych na rynku (w tym wolumen 24h).                                                                                                                                                                                                                                                                                                              |
| Get24hrStats        | Symbol | Żądanie wysłane do tego punktu końcowego zwraca statystyki określonego tickera z ostatnich 24 godzin.                                                                                                                                                                                                                                                                                           |
| GetMarketList       |        | Użyj tego punktu końcowego, aby uzyskać walutę transakcji dla całego rynku handlowego.                                                                                                                                                                                                                                                                                                          |
| GetPartOrderBook20  | Symbol | Wysyłając żądanie do tego punktu końcowego, można uzyskać listę otwartych zleceń dla symbolu. Arkusz zleceń poziomu 2 zawiera wszystkie oferty bid i ask (zagregowane według ceny); ten poziom zwraca tylko jeden rozmiar dla każdej aktywnej ceny (tak, jakby istniało tylko jedno zlecenie dla danej ceny). System zwróci 20 pozycji danych (dane bid i ask) z arkusza zleceń.                |
| GetPartOrderBook100 | Symbol | Wyślij żądanie do tego punktu końcowego, aby uzyskać listę otwartych zleceń dla symbolu. Arkusz zleceń poziomu 2 zawiera wszystkie oferty kupna i sprzedaży (zagregowane według ceny); ten poziom zwraca tylko jeden rozmiar dla każdej aktywnej ceny (tak jakby istniało tylko jedno zlecenie dla tej ceny). System zwróci 100 pozycji danych (dane ofert kupna i sprzedaży) z arkusza zleceń. |
| GetFullOrderBook    | Symbol | Żądanie przez ten punkt końcowy zwraca arkusz zleceń dla podanego symbolu. Arkusz zleceń poziomu 2 zawiera wszystkie oferty kupna i sprzedaży (zagregowane według ceny). Ten poziom zwraca tylko jeden zagregowany rozmiar dla każdej ceny (jakby istniało tylko jedno zlecenie dla tej ceny). To API zwraca dane z pełną głębokością.                                                          |
| GetKLines           | Symbol | Żądaj przez ten punkt końcowy, aby uzyskać kline dla określonego symbolu. Dane są zwracane w zgrupowanych segmentach na podstawie żądanego typu.                                                                                                                                                                                                                                                |
| GetCurrencies       |        | Wyślij żądanie do tego punktu końcowego, aby uzyskać listę walut.                                                                                                                                                                                                                                                                                                                               |

|                   |        |                                                                                                                     |
|-------------------|--------|---------------------------------------------------------------------------------------------------------------------|
| GetCurrencyDetail | Waluta | Wyślij żądanie do tego punktu końcowego, aby uzyskać szczegóły waluty dla określonej waluty                         |
| GetFiatPrice      |        | Wyślij żądanie do tego punktu końcowego, aby uzyskać szczegóły waluty dla określonej waluty                         |
| GetPartOrderBook1 | Symbol | Wyślij żądanie do tego punktu końcowego, aby uzyskać najlepszą ofertę kupna/sprzedaży poziomu 1 dla danego symbolu. |

## Prywatne punkty końcowe API

Wymaga podania APIKey i APISecret w celu autoryzacji przez serwer.

### Punkty końcowe użytkownika

|                              |           |                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetAllSubAccounts            |           | Informacje o użytkowniku wszystkich sub-użytkowników można uzyskać przez ten interfejs.                                                                                                                                                                                                                                                                                                                       |
| GetListAccounts              |           | Pobierz listę kont.                                                                                                                                                                                                                                                                                                                                                                                           |
| GetAccount                   | AccountId | Informacje dotyczące pojedynczego konta. Tego punktu końcowego należy użyć, gdy znany jest identyfikator accountId.                                                                                                                                                                                                                                                                                           |
| GetAccountBalance-SubAccount | SubUserId | Ten punkt końcowy zwraca informacje o koncie użytkownika podrzędne-<br>go określonego przez subUserId.                                                                                                                                                                                                                                                                                                        |
| InnerTransfer                |           | Ten punkt końcowy API umożliwia wewnętrzne przelewy środków między kontami. Użytkownicy mogą bezpłatnie przenosić środki między kontem głównym, kontem handlowym, kontem marży krzyżowej i kontem marży izolowanej. Obsługiwany jest transfer środków z konta głównego, konta marży krzyżowej i konta handlowego na konto futures, jednak transfer środków z kont futures na inne konta nie jest obsługiwany. |
| GetDepositAddresses          | Waluta    | Pobierz adresy depozytów dla waluty.                                                                                                                                                                                                                                                                                                                                                                          |
| CreateDepositAddress         | Waluta    | Utwórz nowy adres depozytu dla waluty.                                                                                                                                                                                                                                                                                                                                                                        |
| GetDepositList               |           | Pobierz historię depozytów.                                                                                                                                                                                                                                                                                                                                                                                   |
| GetAccountLedgers            |           | Pobierz wpisy księgi rachunkowej konta.                                                                                                                                                                                                                                                                                                                                                                       |
| GetTradeFees                 | Symbole   | Pobiera opłaty transakcyjne dla wskazanych instrumentów.                                                                                                                                                                                                                                                                                                                                                      |

### Punkty końcowe wypłat

|                              |                      |                                                            |
|------------------------------|----------------------|------------------------------------------------------------|
| GetWithdrawalsList           |                      | Pobierz listę wypłat.                                      |
| GetHistoricalWithdrawalsList |                      | Lista historycznych wypłat KuCoin V1.                      |
| GetWithdrawalsQuotes         | Waluta               | Pobierz limity wypłat                                      |
| ApplyWithdraw                | Waluta, adres, kwota | Utwórz wypłatę                                             |
| CancelWithdraw               | WithdrawalId         | Anulować można tylko żądania wypłat o statusie PROCESSING. |

## Punkty końcowe handlu

|                            |                                                                                                                                                                                                                                                                                                                       |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PlaceOrder                 | Można składać dwa rodzaje zleceń: limitowane i rynkowe. Zlecenia mogą być składane wyłącznie wtedy, gdy konto posiada wystarczające środki. Po złożeniu zlecenia środki na koncie zostają zablokowane na czas jego realizacji. Wysokość i rodzaj zablokowanych środków zależy od typu zlecenia i podanych parametrów. |
| PlaceMarketOrder           | Składa zlecenie rynkowe.                                                                                                                                                                                                                                                                                              |
| PlaceLimitOrder            | Składa zlecenie Limit.                                                                                                                                                                                                                                                                                                |
| PlaceMarginOrder           | Składa zlecenie depozytowe z depozytem zabezpieczającym.                                                                                                                                                                                                                                                              |
| CancelOrder                | Anuluje zlecenie według identyfikatora zlecenia.                                                                                                                                                                                                                                                                      |
| CancelOrderByClientOid     | Anuluje zlecenie według identyfikatora zlecenia klienta.                                                                                                                                                                                                                                                              |
| CancelAllOrders            | Anuluj wszystkie otwarte zlecenia.                                                                                                                                                                                                                                                                                    |
| ListOrders                 | Wyślij żądanie do tego punktu końcowego, aby pobrać bieżącą listę zleceń. Elementy są stronicowane i posortowane w kolejności od najnowszych                                                                                                                                                                          |
| GetRecentOrders            | Żądanie przez ten punkt końcowy pozwala pobrać 1000 zleceń z ostatnich 24 godzin.                                                                                                                                                                                                                                     |
| GetOrder                   | Wysyłając żądanie do tego punktu końcowego, można uzyskać informacje o pojedynczym zleceniu według identyfikatora zlecenia.                                                                                                                                                                                           |
| GetOrderByClientOid        | Żądanie do tego endpointu w celu uzyskania informacji o pojedynczym zleceniu na podstawie identyfikatora zlecenia klienta.                                                                                                                                                                                            |
| ListFills                  | Użyj tego punktu końcowego, aby uzyskać ostatnie realizacje.                                                                                                                                                                                                                                                          |
| GetRecentFills             | Wysyłając żądanie do tego punktu końcowego, można uzyskać listę 1000 transakcji z ostatnich 24 godzin.                                                                                                                                                                                                                |
| PlaceStopOrder             | Składa zlecenie Stop.                                                                                                                                                                                                                                                                                                 |
| PlaceStopMarketOrder       | Składa zlecenie Stop Market.                                                                                                                                                                                                                                                                                          |
| PlaceStopLimitOrder        | Składa zlecenie Stop Limit.                                                                                                                                                                                                                                                                                           |
| CancelStopOrder            | Anuluje otwarte zlecenie Stop według identyfikatora zlecenia                                                                                                                                                                                                                                                          |
| CancelStopOrderByClientOid | Anuluje otwarte zlecenie stop na podstawie identyfikatora zlecenia klienta                                                                                                                                                                                                                                            |
| CancelAllStopOrders        | Anuluj wszystkie zlecenia stop                                                                                                                                                                                                                                                                                        |
| GetStopOrder               | Żądanie przez ten interfejs w celu uzyskania informacji o zleceniu stop na podstawie identyfikatora zlecenia.                                                                                                                                                                                                         |

|                           |                                                                                                                                                         |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetStopOrderByClientOid   | Wyślij żądanie przez ten interfejs, aby uzyskać informacje o zleceniu stop za pomocą identyfikatora zlecenia klienta.                                   |
| ListStopOrders            | Żądanie do tego punktu końcowego pozwala pobrać bieżącą listę nieaktywowanych zleceń stop. Elementy są stronicowane i posortowane malejąco według daty. |
| PlaceHFOOrder             | Złóż zlecenie o wysokiej częstotliwości.                                                                                                                |
| CancelHFOOrder            | Anuluje zlecenie wysokiej częstotliwości na podstawie identyfikatora zlecenia.                                                                          |
| CancelHFOOrderByClientOid | Anulowanie zlecenia wysokiej częstotliwości według identyfikatora zlecenia klienta.                                                                     |
| CancelAllHFOOrders        | Anuluj wszystkie zlecenia o wysokiej częstotliwości.                                                                                                    |
| GetHFActiveOrders         | Pobierz aktywne zlecenia wysokiej częstotliwości.                                                                                                       |
| GetHFDoneOrders           | Pobierz zakończone zlecenia o wysokiej częstotliwości.                                                                                                  |
| GetHFOOrder               | Pobierz określone zlecenie o wysokiej częstotliwości według identyfikatora zlecenia.                                                                    |

## Zdarzenia

Wiadomości Kucoin są odbierane w komponencie `TsgcWebSocketClient`; można użyć następujących zdarzeń:

### OnConnect

Po pomyślnym nawiązaniu połączenia z serwerem Kucoin.

### OnDisconnect

Po rozłączeniu z serwerem Kucoin

### OnMessage

Wiadomości wysyłane przez serwer do klienta są obsługiwane w tym zdarzeniu.

### OnError

W przypadku wystąpienia błędu protokołu wywoływane jest to zdarzenie.

### OnException

Jeżeli wystąpi nieobsługiwany wyjątek, zostanie wywołane to zdarzenie.

Dodatkowo w komponencie Kucoin API istnieje specyficzne zdarzenie o nazwie **OnKucoinHTTPException**, wywoływane za każdym razem, gdy wystąpi błąd podczas wywołania żądania HTTP (REST API lub WebSocket Feeds).



# Kucoin | Połącz WebSocket API

Aby połączyć się z Kucoin WebSocket API, wystarczy utworzyć nowego klienta API Kucoin i podłączyć go do TsgcWebSocketClient.

Poniżej znajduje się przykład:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin();
oKucoin->Client = oClient;
oClient->Active = true;
```

# Kucoin | Subskrypcja kanału WebSocket

Kucoin oferuje wiele kanałów, do których można subskrybować aktualizacje danych rynkowych, zleceń itd. w czasie rzeczywistym. Poniżej znajduje się przykład subskrypcji kanału Ticker:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin();
oKucoin->Client = oClient;
oKucoin->SubscribeSymbolTicker("BTC-USDT");
void OnMessage(TsgcWSConnection *Connection, const string aText)
{
    // here you will receive the ticker updates
}
```

## Kucoin | Pobieranie danych rynkowych

Kucoin udostępnia publiczne dane rynkowe przez punkty końcowe REST. Po wywołaniu jednego z tych punktów końcowych otrzymuje się migawkę żądanych danych rynkowych.

Punkty końcowe danych rynkowych nie wymagają uwierzytelniania i są ogólnodostępne dla wszystkich użytkowników.

**Przykład:** aby uzyskać migawkę tickera BTC-USDT, należy wykonać następujące wywołanie

```
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin(this);  
ShowMessage(oKucoin->REST_API->GetTicker("BTC-USDT"));
```

# Kucoin | Prywatne REST API

REST API Kucoin oferuje publiczne i prywatne punkty końcowe. Prywatne punkty końcowe wymagają podpisywania wiadomości w celu zwiększenia bezpieczeństwa transakcji.

Najpierw należy zalogować się na konto Kucoin i utworzyć nowe API. Zostaną udostępnione następujące wartości:

- ApiKey
- ApiSecret
- Hasło dostępu

Poniższe pola muszą być skonfigurowane we właściwości Kucoin komponentu klienta API Kucoin.

Po skonfigurowaniu można rozpocząć wysyłanie prywatnych żądań do Kucoin Pro REST API.

\*Prywatne żądania wymagają zsynchronizowania czasu lokalnego maszyny. W przeciwnym razie żądania zostaną odrzucone przez serwer Kucoin. Zapoznaj się z następującym artykułem na ten temat: [Kucoin Private Requests Time](#).

```
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin(this);  
oKucoin->Kucoin->ApiKey = "<your api key>";  
oKucoin->Kucoin->ApiSecret = "<your api secret>";  
oKucoin->Kucoin->Passphrase = "<your passphrase>";  
ShowMessage(oKucoin->REST_API->GetListAccounts());
```

# Kucoin | Handel Spot

Kucoin umożliwia handel spot za pomocą REST API.

## Konfiguracja

Najpierw należy utworzyć **klucz API** na koncie Kucoin i przyznać uprawnienia do handlu z Spot.

Po wykonaniu tych czynności można rozpocząć handel spot.

Najpierw **ustaw ApiKey, ApiSecret i Passphrase** w komponencie klienta Kucoin; wartości te będą używane do podpisywania żądań wysyłanych do serwera Kucoin.

## Złóż zlecenie

Aby złożyć nowe zlecenie, wystarczy wywołać metodę **REST\_API.PlaceOrder** komponentu klienta Kucoin.

W zależności od typu zlecenia (rynkowe, limit...) API wymaga mniejszej lub większej liczby pól.

## Parametry

| Param      | type   | Opis                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| clientOid  | String | Unikalny identyfikator zlecenia utworzony przez użytkowników w celu identyfikacji ich zleceń, np. UUID.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| strona     | String | <b>kupno</b> lub <b>sprzedaż</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| symbol     | String | prawidłowy kod symbolu handlowego, np. ETH-BTC                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| type       | String | [Opcjonalnie] <b>limit</b> lub <b>market</b> (domyślnie <b>limit</b> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| uwaga      | String | [Opcjonalnie] uwaga do zlecenia; długość nie może przekraczać 100 znaków utf8                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| stp        | String | [Opcjonalnie] zapobieganie handlowi z samym sobą, <b>CN</b> , <b>CO</b> , <b>CB</b> lub <b>DC</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| trade-Type | String | [Opcjonalnie] Typ handlu: <b>TRADE</b> (handel spot) , <b>MARGIN_TRADE</b> (handel z depozytem). Wartość domyślna: <b>TRADE</b> . Uwaga: W celu poprawy wydajności systemu i przyspieszenia składania oraz przetwarzania zleceń KuCoin dodał nowy interfejs do składania zleceń z depozytem. Handlowcy nadal korzystający z obecnego interfejsu powinni jak najszybciej przejść na nowy. Obecny interfejs przestanie przyjmować zlecenia z depozytem od 1 maja 2021 r. (UTC). KuCoin poinformuje użytkowników o tym zdarzeniu poprzez ogłoszenie. |

## PARAMETRY ZLECENIA Z LIMITEM CENY

| Param       | type   | Opis                                                                                                                             |
|-------------|--------|----------------------------------------------------------------------------------------------------------------------------------|
| cena        | String | cena za walutę bazową                                                                                                            |
| size        | String | ilość waluty bazowej do kupna lub sprzedaży                                                                                      |
| timeInForce | String | [Opcjonalnie] <b>GTC</b> , <b>GTT</b> , <b>IOC</b> lub <b>FOK</b> (domyślnie <b>GTC</b> ); patrz <a href="#">Czas ważności</a> . |
| cancelAfter | long   | [Opcjonalnie] anuluj po <b>n</b> sekundach, wymaga ustawienia <b>timeInForce</b> na <b>GTT</b>                                   |

| Param       | type                  | Opis                                                                                    |
|-------------|-----------------------|-----------------------------------------------------------------------------------------|
| postOnly    | boolean [Opcjonalne]  | Flaga tylko post, nieprawidłowa gdy <b>timeInForce</b> wynosi <b>IOC</b> lub <b>FOK</b> |
| ukryty      | boolean [Opcjonalnie] | Zlecenie nie będzie wyświetlane w księdze zleceń                                        |
| iceberg     | boolean [Opcjonalnie] | W arkuszu zleceń wyświetlana jest tylko część zlecenia                                  |
| visibleSize | String [Opcjonalnie]  | Maksymalny widoczny rozmiar zlecenia górę lodową                                        |

## PARAMETRY ZLECENIA RYNKOWEGO

| Param  | type                 | Opis                                     |
|--------|----------------------|------------------------------------------|
| size   | String [Opcjonalnie] | Żądana kwota w walucie bazowej           |
| środki | String [Opcjonalnie] | Żądana kwota waluty kwotowanej do użycia |

•

Podczas składania zlecenia możliwe są 2 scenariusze:

1. **Sukces:** funkcja PlaceOrder zwraca wiadomość wysłaną przez serwer Kucoin.
2. **Error:** wyjątek jest zwracany w zdarzeniu OnKucointHTTPException.

### Złóż zlecenie rynkowe 1 BTC-USDT

```
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin(this);
oKucoin->Kucoin->ApiKey = "<api key>";
oKucoin->Kucoin->ApiSecret = "<api secret>";
oKucoin->Kucoin->Passphrase = "<passphrase>";
ShowMessage(oKucoin->REST_API->PlaceMarketOrder(kosBuy, "BTC-USDT", 1));
```

-

### Złóż zlecenie Limit 1 BTC-USDT po 40000

```
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin(this);
oKucoin->Kucoin->ApiKey = "<api key>";
oKucoin->Kucoin->ApiSecret = "<api secret>";
oKucoin->Kucoin->Passphrase = "<passphrase>";
ShowMessage(oKucoin->REST_API->PlaceLimitOrder(kosBuy, "BTC-USDT", 1, 40000));
```

## Kucoin | Czas prywatnych żądań

Podczas wykonywania prywatnego żądania do Kucoin wiadomość jest podpisywana w celu zwiększenia bezpieczeństwa żądań. Wiadomość zawiera lokalny czas i jest wysyłana w podpisanej wiadomości; jeśli różnica między lokalnym czasem a serwerami Kucoin jest większa niż 5 sekund, żądanie zostanie odrzucone. Ważne jest zatem sprawdzenie, czy lokalny czas jest zsynchronizowany, co można wykonać przy użyciu metody synchronizacji czasu dostępnej w systemie operacyjnym.

Można sprawdzić czas serwera Kucoin, wywołując metodę **GetServerTime**, która zwróci czas serwera Kucoin.

# API Kucoin Futures

## Kucoin Futures

Kucoin to międzynarodowa, wielojęzyczna giełda kryptowalut. Udostępnia szereg API umożliwiających dostęp do danych Kucoin. Obsługiwane są następujące API:

1. **Strumienie WebSocket:** pozwalają subskrybować określone metody i otrzymywać dane w czasie rzeczywistym. Zdarzenia są wypychane do klientów przez serwer do subskrybentów. Używa protokołu WebSocket.
2. **REST API:** klienci mogą żądać danych rynkowych i danych konta od serwera. Wymaga klucza API, sekretu i hasła do uwierzytelnienia i używa protokołu HTTPS.

## Właściwości

Kucoin API posiada 2 typy metod: publiczne i prywatne. Metody publiczne są dostępne bez uwierzytelniania, na przykład: pobieranie cen tickers. Metody prywatne związane z danymi użytkownika wymagają użycia kluczy Kucoin API.

- **ApiKey:** można wygenerować nowy klucz API na koncie kucoin i skopiować jego wartość do tej właściwości.
- **ApiSecret:** ApiSecret jest wymagany tylko dla REST\_API; websocket API wymaga ApiKey jedynie dla niektórych metod.
- **Passphrase:** ciąg znaków wymagany do połączenia z serwerami Kucoin.
- **Sandbox:** jeśli włączony, połączy się z kontem demonstracyjnym Kucoin (domyślnie false).
  - **HTTPLogOptions:** przechowuje dziennik żądań HTTP w pliku tekstowym
    - **Enabled:** jeśli włączone, będzie przechowywać wszystkie żądania HTTP WebSocket API.
    - **FileName:** pełna ścieżka pliku, w którym będą przechowywane dzienniki
  - **REST:** przechowuje dziennik żądań REST API w pliku tekstowym
    - **Enabled:** jeśli włączone, będzie przechowywać wszystkie żądania HTTP REST API.
    - **FileName:** pełna ścieżka pliku, w którym będą przechowywane dzienniki.

## Najczęstsze zastosowania

- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować kanał WebSocket](#)
- **REST API**
  - [Jak pobrać dane rynkowe](#)
  - [Jak korzystać z prywatnego REST API](#)
  - [Jak handlować kontraktami futures](#)
  - [Czas prywatnych żądań](#)



## WebSocket Feed

Aby subskrybować wiadomości kanału z określonego serwera, strona klienta powinna wysłać wiadomość subskrypcji do serwera.

Jeśli subskrypcja zakończy się powodzeniem, system wyśle do Ciebie wiadomości ack, gdy odpowiedź będzie ustawiona na true.

```
{  
  "id": "1545910660739",  
  "type": "ack"  
}
```

Dopóki generowane są wiadomości dotyczące tematu, system wysyła odpowiednie wiadomości do strony klienta.

Obsługiwane są następujące metody subskrypcji i anulowania subskrypcji.

### Kanały publiczne

|                                |        |                                                                                                                                                                                                                                                                                              |
|--------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SubscribeSymbolTickerView2     | Symbol | Subskrypcja tego tematu umożliwia otrzymywanie aktualizacji BBO w czasie rzeczywistym. Po subskrypcji, przy każdej zmianie w księdze zleceń, system będzie wysyłać aktualne informacje o symbolu ticker. Zalecane jest korzystanie z nowego tematu w celu uzyskiwania aktualnych informacji. |
| SubscribeSymbolTicker          | Symbol | Subskrybuj ten temat, aby otrzymywać aktualizacje zmian BBO w czasie rzeczywistym. Kanał ticker dostarcza aktualizacje cen w czasie rzeczywistym przy każdym dopasowaniu. Jeśli wiele zleceń zostanie dopasowanych jednocześnie, przesłane zostanie tylko ostatnie zdarzenie dopasowania.    |
| SubscribeLevel2MarketData      | Symbol | Zasubskrybuj ten temat, aby otrzymywać dane arkusza zleceń poziomu 2.                                                                                                                                                                                                                        |
| SubscribeExecutionData         | Symbol | Dla każdego wykonanego zlecenia system wysyła wiadomości o dopasowaniu w następującym formacie.                                                                                                                                                                                              |
| SubscribeLevel2_5BestAskBid    | Symbol | Zwracane co najwyżej co 100 milisekund.                                                                                                                                                                                                                                                      |
| SubscribeLevel2_50BestAskBid   | Symbol | Zwracane co najwyżej co 100 milisekund.                                                                                                                                                                                                                                                      |
| SubscribeContractMarketData    | Symbol | Zasubskrybuj ten temat, aby otrzymywać dane rynkowe kontraktu.                                                                                                                                                                                                                               |
| SubscribeSystemAnnouncements   | Symbol | Subskrybuj ten temat, aby otrzymywać komunikaty systemowe.                                                                                                                                                                                                                                   |
| SubscribeTransactionStatistics | Symbol | Statystyki transakcji będą przesyłane do użytkowników co 5 sekund.                                                                                                                                                                                                                           |
| SubscribeKlines                | Symbol | Subskrybuj dane klines (świece japońskie) dla kontraktu.                                                                                                                                                                                                                                     |
| SubscribeFundingFeeSettlement  | Symbol | Subskrybuj powiadomienia o rozliczeniach opłat finansowania.                                                                                                                                                                                                                                 |

Jeśli parametr ACK jest ustawiony na true, po pomyślnej subskrypcji lub anulowaniu subskrypcji klient otrzymuje o tym wiadomość.

## Kanały prywatne

Wymaga ważnego ApiKey uzyskanego z konta Kucoin. ApiKey, ApiSecret i Passphrase muszą być ustawione we właściwości Kucoin komponentu klienta API.

Następujące dane są wypychane do klienta za każdym razem, gdy nastąpi zmiana. Nie jest konieczna subskrypcja żadnej metody — odbywa się ona automatycznie po ustawieniu prawidłowego ApiKey.

|                              |                                                                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SubscribeTradeOrders         | Ten temat przekazuje wszystkie zdarzenia zmian dotyczące zamówień użytkownika.                                                                                         |
| SubscribeAccountBalance      | Ta wiadomość jest odbierana, gdy zmienia się saldo konta. Zawiera szczegóły dotyczące zmiany.                                                                          |
| SubscribePositionChange      | System wyśle zdarzenie zmiany, gdy zmieni się status pozycji.                                                                                                          |
| SubscribeStopOrder           | Gdy zlecenie stop zostanie odebrane przez system, otrzymasz wiadomość z typem "open". Oznacza to, że zlecenie zostało wprowadzone do systemu i oczekuje na wyzwolenie. |
| SubscribeMarginMode          | Subskrybuj zmiany trybu depozytów zabezpieczających. System wyśle zdarzenie zmiany po zaktualizowaniu trybu depozytów.                                                 |
| SubscribeCrossMarginLeverage | Subskrybuj zmiany dźwigni cross margin. System wypychać będzie zdarzenie zmiany po zaktualizowaniu dźwigni cross margin.                                               |

## REST API

Wszystkie punkty końcowe zwracają obiekt JSON lub tablicę.

### Publiczne punkty końcowe API

Te punkty końcowe są dostępne bez żadnej autoryzacji.

#### Ogólne punkty końcowe

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| GetServiceStatus | Sprawdzenie łączności z Rest API i pobranie statusu usługi    |
| GetServerTime    | Sprawdź połączenie z Rest API i pobierz bieżący czas serwera. |

## Punkty końcowe danych rynkowych

|                          |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetOpenContractList      |        | Wyślij żądanie pobrania informacji o wszystkich otwartych kontraktach.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| GetOrderInfoContract     |        | Wyślij żądanie, aby uzyskać informacje o podanym kontrakcie.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| GetTicker                | Symbol | Ticker w czasie rzeczywistym zawiera ostatnią cenę transakcji, ostatnią wielkość transakcji, identyfikator transakcji, stronę dostawcy płynności, najlepszą cenę i wielkość oferty kupna oraz oferty sprzedaży, a także czas transakcji zleceń. Wiadomości te można uzyskać również przez WebSocket. Numer sekwencji służy do oceny ciągłości wiadomości przesyłanych przez WebSocket.                                                                                                                                                                                                                                                                        |
| GetPartOrderBook20       | Symbol | Pobiera migawkę zagregowanych otwartych zleceń dla danego symbolu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| GetPartOrderBook100      | Symbol | Pobiera migawkę zagregowanych otwartych zleceń dla danego symbolu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| GetFullOrderBook         | Symbol | Pobiera migawkę zagregowanych otwartych zleceń dla danego symbolu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| GetLevel2PullingMessages | Symbol | Jeśli wiadomości przesyłane przez WebSocket nie są ciągłe, można wysłać poniższe żądanie i ponownie pobrać dane, aby upewnić się, że sekwencja jest kompletna. W żądaniu parametr start oznacza numer sekwencji ostatnio odebranej wiadomości plus 1, a parametr end oznacza numer sekwencji aktualnie odebranej wiadomości minus 1. Po ponownym pobraniu wiadomości i zastosowaniu ich do lokalnej książki zleceń można kontynuować jej aktualizację za pomocą strumieniowego kanału WebSocket. Jeśli różnica między parametrami end i start przekracza 500, należy zaprzestać korzystania z tego żądania i zaleca się przebudowanie książki zleceń Level 2. |
| GetTradeHistory          | Symbol | Lista ostatnich 100 transakcji dla symbolu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| GetInterestRateList      | Symbol | Sprawdzenie listy stóp procentowych.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| GetIndexList             | Symbol | Sprawdź listę indeksów                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| GetCurrentMarkPrice      | Symbol | Sprawdź bieżącą cenę mark.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| GetPremiumIndex          | Symbol | Wysyła żądanie pobrania indeksu premii.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| GetCurrentFundingRate    | Symbol | Wysyłanie żądania sprawdzenia bieżącej ceny mark.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| GetKLine                 | Symbol | Pobierz dane K Line kontraktu                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Prywatne punkty końcowe API

Wymaga podania APIKey i APISecret w celu autoryzacji przez serwer.

## Punkty końcowe użytkownika

|                       |                                                                                                                                                                                                                                                                                                |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetAccountOverview    | Pobierz przegląd konta                                                                                                                                                                                                                                                                         |
| GetTransactionHistory | Jeśli istnieją otwarte pozycje, status pierwszej zwróconej strony będzie Pending, co wskazuje na zrealizowany zysk i stratę w bieżącym 8-godzin-<br>nym okresie rozliczeniowym. Należy podać minimalny numer przesunię-<br>cia bieżącej strony w polu offset, aby przejść do następnej strony. |

## Punkty końcowe handlu

|                             |                                                                                                                                                                                                                                                                                                    |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PlaceOrder                  | Można składać dwa rodzaje zleceń: limit i market. Zlecenia mogą być składane wyłącznie wtedy, gdy konto dysponuje wystarczającymi środkami. Po złożeniu zlecenia środki zostają zablokowane na czas jego trwania. Wysokość zablokowanych środków zależy od rodzaju zlecenia i podanych parametrów. |
| PlaceMarketOrder            | Składa zlecenie rynkowe.                                                                                                                                                                                                                                                                           |
| PlaceLimitOrder             | Składa zlecenie Limit.                                                                                                                                                                                                                                                                             |
| CancelOrder                 | Anuluje zlecenie według identyfikatora zlecenia.                                                                                                                                                                                                                                                   |
| LimitOrderMassCancellation  | Anuluje wszystkie otwarte zlecenia (z wyjątkiem zleceń stop). Odpowiedź zawiera listę identyfikatorów anulowanych zleceń.                                                                                                                                                                          |
| StopOrderMassCancellation   | Anuluj wszystkie niewyzwolone zlecenia stop. Odpowiedź zawiera listę identyfikatorów anulowanych zleceń stop. Aby anulować wyzwolone zlecenia stop, należy użyć funkcji „Limit Order Mass Cancellation”.                                                                                           |
| GetOrderList                | Wyświetl bieżące zlecenia.                                                                                                                                                                                                                                                                         |
| GetUntriggeredStopOrderList | Pobierz listę niewyzwolonych zleceń stop.                                                                                                                                                                                                                                                          |
| GetListOrdersCompleted24hr  | Pobierz listę ostatnich 1000 zleceń z ostatnich 24 godzin. Jeśli wymagane jest pobieranie historii zrealizowanych zleceń z małym opóźnieniem, można skorzystać z tego punktu końcowego.                                                                                                            |
| GetOrder                    | Pobierz pojedyncze zlecenie według identyfikatora (w tym zlecenie stop).                                                                                                                                                                                                                           |
| GetOrderByClientOid         | Pobierz pojedyncze zlecenie według identyfikatora zlecenia klienta (łącznie ze zleceniem stop).                                                                                                                                                                                                    |
| GetFills                    | Pobierz listę ostatnich transakcji.                                                                                                                                                                                                                                                                |
| GetRecentFills              | Pobierz listę ostatnich 1000 transakcji z ostatnich 24 godzin. Jeśli wymagane jest uzyskanie historii ostatnio zrealizowanych zleceń z niskim opóźnieniem, można skorzystać z tego punktu końcowego.                                                                                               |
| ActiveOrderValueCalculation | Można zapytać ten punkt końcowy, aby uzyskać łączną liczbę i wartość wszystkich aktywnych zleceń.                                                                                                                                                                                                  |
| GetPositionDetails          | Pobierz szczegóły określonej pozycji.                                                                                                                                                                                                                                                              |
| GetPositionList             | Pobierz szczegóły określonej pozycji.                                                                                                                                                                                                                                                              |
| AutoDepositMargin           | Włącz/Wyłącz automatyczne uzupełnianie depozytu zabezpieczającego                                                                                                                                                                                                                                  |
| AddMarginManually           | Dodaj depozyt zabezpieczający ręcznie                                                                                                                                                                                                                                                              |

|                             |                                                                                                                                                                                                      |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ObtainFuturesRiskLimitLevel | Interfejs ten może być używany do uzyskiwania informacji o poziomie limitu ryzyka dla określonego kontraktu                                                                                          |
| AdjustRiskLimitLevel        | Ten interfejs służy do dostosowania poziomu limitu ryzyka. Zmiana poziomu spowoduje anulowanie otwartego zlecenia; odpowiedź może jedynie wskazywać, czy żądanie zmiany zostało pomyślnie przesłane. |
| GetFundingHistory           | Prześlij żądanie, aby uzyskać historię finansowania.                                                                                                                                                 |
| GetMaxOpenSize              | Pobierz maksymalny rozmiar otwartej pozycji dla kontraktu.                                                                                                                                           |
| SwitchMarginMode            | Przełączanie między trybem cross margin a isolated margin.                                                                                                                                           |
| GetMarginMode               | Pobierz bieżący tryb marży dla kontraktu.                                                                                                                                                            |

## Zdarzenia

Wiadomości Kucoin są odbierane w komponencie TsgcWebSocketClient; można użyć następujących zdarzeń:

### OnConnect

Po pomyślnym nawiązaniu połączenia z serwerem Kucoin.

### OnDisconnect

Po rozłączeniu z serwerem Kucoin

### OnMessage

Wiadomości wysyłane przez serwer do klienta są obsługiwane w tym zdarzeniu.

### OnError

W przypadku wystąpienia błędu protokołu wywoływane jest to zdarzenie.

### OnException

Jeżeli wystąpi nieobsługiwany wyjątek, zostanie wywołane to zdarzenie.

Dodatkowo w komponencie Kucoin API istnieje specyficzne zdarzenie o nazwie **OnKucoinHTTPException**, wywoływane za każdym razem, gdy wystąpi błąd podczas wywołania żądania HTTP (REST API lub WebSocket Feeds).

# Kucoin | Podłącz WebSocket API dla kontraktów Futures

Aby połączyć się z Kucoin WebSocket API, wystarczy utworzyć nowego klienta API Kucoin i podłączyć go do TsgcWebSocketClient.

Poniżej znajduje się przykład:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures();
oKucoin->Client = oClient;
oClient->Active = true;
```

# Kucoin | Subskrypcja kanału WebSocket dla kontraktów futures

Kucoin oferuje wiele kanałów, do których można subskrybować aktualizacje danych rynkowych, zleceń itd. w czasie rzeczywistym. Poniżej znajduje się przykład subskrypcji kanału Ticker:

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures();
oKucoin->Client = oClient;
oKucoin->SubscribeSymbolTickerV2("XBTUSD");
void OnMessage(TsgcWSConnection *Connection, const string aText)
{
    // here you will receive the ticker updates
}
```

## Kucoin | Futures Get Market Data

Kucoin udostępnia publiczne dane rynkowe przez punkty końcowe REST. Po wywołaniu jednego z tych punktów końcowych otrzymuje się migawkę żądanych danych rynkowych.

Punkty końcowe danych rynkowych nie wymagają uwierzytelniania i są ogólnodostępne dla wszystkich użytkowników.

**Przykład:** aby uzyskać migawkę tickera BTC-USDT, należy wykonać następujące wywołanie

```
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures(this);  
ShowMessage(oKucoin->REST_API->GetTicker("XBTUSD"));
```



# Kucoin | Prywatne REST API kontraktów futures

REST API Kucoin oferuje publiczne i prywatne punkty końcowe. Prywatne punkty końcowe wymagają podpisywania wiadomości w celu zwiększenia bezpieczeństwa transakcji.

Najpierw należy zalogować się na konto Kucoin i utworzyć nowe API. Zostaną udostępnione następujące wartości:

- ApiKey
- ApiSecret
- Hasło dostępu

Poniższe pola muszą być skonfigurowane we właściwości Kucoin komponentu klienta API Kucoin.

Po skonfigurowaniu można rozpocząć wysyłanie prywatnych żądań do Kucoin Pro REST API.

\*Żądania prywatne wymagają synchronizacji czasu lokalnego z czasem serwera Kucoin; w przeciwnym razie żądania zostaną odrzucone. Informacje na ten temat znajdują się w artykule [Czas żądań prywatnych Kucoin](#).

```
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures(this);
oKucoin->Kucoin->ApiKey = "<your api key>";
oKucoin->Kucoin->ApiSecret = "<your api secret>";
oKucoin->Kucoin->Passphrase = "<your passphrase>";
ShowMessage(oKucoin->REST_API->GetAccountOverview());
```

# Kucoin | Handel futures

Kucoin umożliwia handel kontraktami futures za pośrednictwem REST API.

## Konfiguracja

Najpierw należy utworzyć **klucz API** na koncie Kucoin i dodać uprawnienia do handlu kontraktami Futures.

Po wykonaniu tych czynności można rozpocząć handel kontraktami terminowymi.

Najpierw **ustaw ApiKey, ApiSecret i Passphrase** w komponencie klienta Kucoin; wartości te będą używane do podpisywania żądań wysyłanych do serwera Kucoin.

## Złóż zlecenie

Aby złożyć nowe zlecenie, wystarczy wywołać metodę **REST\_API.PlaceOrder** komponentu klienta Kucoin.

W zależności od typu zlecenia (rynkowe, limit...) API wymaga mniejszej lub większej liczby pól.

## Parametry

| Parametr      | type    | Opis                                                                                                                                                                                |
|---------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| clientOrderId | String  | Unikalny identyfikator zamówienia tworzony przez użytkowników w celu identyfikacji ich zleceń, np. UUID. Dozwolone są wyłącznie cyfry, litery, podkreślenie (_) oraz separator (-). |
| strona        | String  | <b>kupno</b> lub <b>sprzedaż</b>                                                                                                                                                    |
| symbol        | String  | prawidłowy kod kontraktu, np. XBTUSDM                                                                                                                                               |
| type          | String  | [opcjonalnie] Albo <b>limit</b> , albo <b>market</b>                                                                                                                                |
| dźwignia      | String  | Dźwignia zlecenia                                                                                                                                                                   |
| uwaga         | String  | [opcjonalnie] uwaga do zlecenia; długość nie może przekraczać 100 znaków utf8                                                                                                       |
| stop          | String  | [opcjonalnie] Wartość <b>down</b> lub <b>up</b> . Wymaga zdefiniowania <b>stopPrice</b> i <b>stopPriceType</b>                                                                      |
| stopPriceType | String  | [opcjonalnie] Wartość <b>TP</b> , <b>IP</b> lub <b>MP</b> ; musi być zdefiniowana, jeśli podano parametr <b>stop</b> .                                                              |
| stopPrice     | String  | [opcjonalnie] Musi być zdefiniowane, jeśli podano parametr <b>stop</b> .                                                                                                            |
| reduceOnly    | boolean | [opcjonalnie] Znacznik ograniczający wyłącznie do zmniejszania pozycji. Domyślnie ustawiony na false. Gdy reduceOnly ma wartość true, wymagane jest podanie rozmiaru pozycji.       |
| closeOrder    | boolean | [opcjonalnie] Znacznik zamknięcia pozycji. Domyślnie ustawiony na false. Zamyka wszystkie pozycje, gdy closeOrder ma wartość true.                                                  |

| Parametr   | type    | Opis                                                                                                                                                                                                                                                                     |
|------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| force-Hold | boolean | [opcjonalnie] Znacznik wymuszający blokadę środków dla zlecenia, nawet jeśli jest to zlecenie zmniejszające rozmiar pozycji. Pomaga to utrzymać zlecenie w arkuszu zleceń i zapobiega jego anulowaniu w przypadku zmiany rozmiaru pozycji. Domyślnie ustawione na false. |

## PARAMETRY ZLECENIA Z LIMITEM CENY

| Param        | type    | Opis                                                                                                                                                                 |
|--------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cena         | String  | Cena limitowana                                                                                                                                                      |
| rozmiar      | Integer | Rozmiar zlecenia. Musi być liczbą dodatnią                                                                                                                           |
| timeIn-Force | String  | [opcjonalnie] <b>GTC</b> , <b>IOC</b> (domyślnie <b>GTC</b> ), zob. <a href="#">Time In Force</a>                                                                    |
| postOnly     | boolean | [opcjonalnie] Flaga tylko dla zleceń post, nieprawidłowa gdy <b>timeInForce</b> wynosi <b>IOC</b> . Przy wyborze postOnly nie można wybrać opcji hidden ani iceberg. |
| ukryty       | boolean | [opcjonalnie] Zlecenia niewidoczne w arkuszu zleceń. Po wybraniu opcji ukrycia, wybór postOnly jest niedozwolony.                                                    |
| iceberg      | boolean | [opcjonalnie] W księdze zleceń wyświetlana jest tylko widoczna część zlecenia. Po wybraniu opcji iceberg nie można wybrać postOnly.                                  |
| visibleSize  | Integer | [opcjonalnie] Maksymalny widoczny rozmiar zlecenia iceberg                                                                                                           |

## PARAMETRY ZLECENIA RYNKOWEGO

| Param | type    | Opis                                                       |
|-------|---------|------------------------------------------------------------|
| size  | Integer | [opcjonalnie] liczba kontraktów do kupienia lub sprzedania |

## Podczas składania zlecenia możliwe są 2 scenariusze:

1. **Sukces:** funkcja PlaceOrder zwraca wiadomość wysłaną przez serwer Kucoin.
2. **Error:** wyjątek jest zwracany w zdarzeniu OnKucoinHTTPException.

## Złóż zlecenie rynkowe na 1 XBTUSDM

```
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures(this);
oKucoin->Kucoin->ApiKey = "<api key>";
oKucoin->Kucoin->ApiSecret = "<api secret>";
```

```
oKucoin->Kucoin->Passphrase = "<passphrase>";  
ShowMessage(oKucoin->REST_API->PlaceMarketOrder(kosBuy, "XBTUSD", 1));
```

-

### **Złóż zlecenie Limit 1 XBTUSD po cenie 40000**

```
TsgcWSAPI_Kucoin_Futures *oKucoin = new TsgcWSAPI_Kucoin_Futures(this);  
oKucoin->Kucoin->ApiKey = "<api key>";  
oKucoin->Kucoin->ApiSecret = "<api secret>";  
oKucoin->Kucoin->Passphrase = "<passphrase>";  
ShowMessage(oKucoin->REST_API->PlaceLimitOrder(kosBuy, "XBTUSD", 1, 40000));
```

## Kucoin | Czas prywatnych żądań Futures

Podczas wykonywania prywatnego żądania do Kucoin wiadomość jest podpisywana w celu zwiększenia bezpieczeństwa żądań. Wiadomość zawiera lokalny czas i jest wysyłana w podpisanej wiadomości; jeśli różnica między lokalnym czasem a serwerami Kucoin jest większa niż 5 sekund, żądanie zostanie odrzucone. Ważne jest zatem sprawdzenie, czy lokalny czas jest zsynchronizowany, co można wykonać przy użyciu metody synchronizacji czasu dostępnej w systemie operacyjnym.

Można sprawdzić czas serwera Kucoin, wywołując metodę **GetServerTime**, która zwróci czas serwera Kucoin.

# API 3Commas

## 3Commas

### Obsługiwane API

- **WebSockets API:** połączenie z publicznym serwerem WebSocket i udostępnianie aktualizacji danych rynkowych w czasie rzeczywistym.
- **REST API:** REST API udostępnia punkty końcowe do zarządzania kontem i zleceniami oraz do pobierania publicznych danych rynkowych.

### WebSockets API

Kanał WebSocket dostarcza aktualizacje danych rynkowych w czasie rzeczywistym dla transakcji i operacji

Możliwa jest subskrypcja następujących **kanałów publicznych**:

**Subscribe-SmartTrades**  
**Subscribe-Deals**

Te kanały wymagają **uwierzytelnienia** wobec serwerów 3Commas. Należy najpierw uzyskać klucze API na koncie 3Commas, a następnie ustawić ich wartości we właściwości ThreeComas komponentu:

- ApiKey
- ApiSecret

Jeśli subskrypcja się powiedzie, zostanie wywołane zdarzenie **OnThreeCommasConfirmSubscription**. W przeciwnym razie zostanie wywołane zdarzenie **OnThreeCommasRejectSubscription**; powód odrzucenia można uzyskać za pomocą parametru **aRawMessage**.

## REST API

### Test łączności

#### GetPing

#### GetServer-Time

Zwraca czas serwera

### Konto

#### GetAccounts

Lista połączonych giełd użytkownika

#### GetMarketList

Lista obsługiwanych rynków

#### GetMarketPairs

**aMarketCode:** kod rynku

Wszystkie pary rynkowe

#### GetCurrencyRatesWithLeverageData

**aMarketCode:** kod rynku  
**aPair:** nazwa pary

Kursy walut i limity z danymi o dźwigni

#### GetCurrencyRates

**aMarketCode:** kod rynku  
**aPair:** nazwa pary

Kursy walut i limity

#### GetBalances

**aAccountId:** identyfikator konta

Równoważ obciążenie dla określonej giełdy

#### GetAccountTableData

**aAccountId:** identyfikator konta

Informacje o wszystkich saldach użytkownika na wskazanej giełdzie

#### GetAccountLeverage

**aAccountId:** identyfikator konta  
**aPair:** nazwa pary

Informacje o dźwigni finansowej konta

#### GetAccountInfo

**aAccountId:** identyfikator konta

Informacje o pojedynczym koncie

### Inteligentne transakcje

#### GetSmartTradeHistory

Pobierz historię transakcji

#### PlaceMarketOrder

**aAccountId:** identyfikator konta  
**aOrderSide:** kupno lub sprzedaż  
**aPair:** na-

Składa zlecenie rynkowe

|                                |                                                                                                                                                              |                                                            |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
|                                | zwa pary <b>aQuantity:</b> ilość                                                                                                                             |                                                            |
| <b>PlaceLimitOrder</b>         | <b>aAccountId:</b> identyfikator konta <b>aOrderSide:</b> kupno lub sprzedaż <b>aPair:</b> nazwa pary <b>aQuantity:</b> ilość <b>aPrice:</b> cena limitowana | Składa zlecenie z limitem ceny                             |
| <b>GetSmartTrade</b>           | <b>aId:</b> identyfikator transakcji                                                                                                                         | Pobierz Smart Trade na podstawie identyfikatora transakcji |
| <b>CancelSmartTrade</b>        | <b>aId:</b> identyfikator transakcji                                                                                                                         | Anuluje Smart Trade na podstawie identyfikatora transakcji |
| <b>CloseByMarketSmartTrade</b> | <b>aId:</b> identyfikator transakcji                                                                                                                         |                                                            |
| <b>EditSmartTrade</b>          | <b>aId:</b> identyfikator transakcji                                                                                                                         | Edytuj istniejący Smart Trade                              |
| <b>ForceStartSmartTrade</b>    | <b>aId:</b> identyfikator transakcji                                                                                                                         | Wymuś uruchomienie Smart Trade                             |
| <b>AddFundsSmartTrade</b>      | <b>aId:</b> identyfikator transakcji                                                                                                                         | Dodaj środki do Smart Trade                                |
| <b>GetSmartTradeTrades</b>     | <b>aId:</b> identyfikator transakcji                                                                                                                         | Pobierz transakcje Smart Trade                             |

## DCA Bot

|                                  |                                |                                           |
|----------------------------------|--------------------------------|-------------------------------------------|
| <b>CreateDCA-Bot</b>             |                                | Utwórz nowego bota DCA                    |
| <b>GetDCABot</b>                 | <b>aId:</b> identyfikator bota | Pobieranie bota DCA według identyfikatora |
| <b>GetDCABots</b>                |                                | Pobierz wszystkie boty DCA                |
| <b>EnableDCA-Bot</b>             | <b>aId:</b> identyfikator bota | Włącz bota DCA                            |
| <b>DisableDCA-Bot</b>            | <b>aId:</b> identyfikator bota | Wyłącz bota DCA                           |
| <b>DeleteDCA-Bot</b>             | <b>aId:</b> identyfikator bota | Usuń bota DCA                             |
| <b>CancelDCA-Bot</b>             | <b>aId:</b> identyfikator bota | Anulowanie bota DCA                       |
| <b>GetDCABot-Stats</b>           |                                | Pobierz statystyki bota DCA               |
| <b>GetAvailableStrategy-List</b> |                                | Pobierz dostępną listę strategii          |
| <b>GetBlacklist-Pairs</b>        |                                | Pobierz pary z czarnej listy              |



## Transakcje

|                          |                                      |                                       |
|--------------------------|--------------------------------------|---------------------------------------|
| <b>GetDeals</b>          |                                      | Pobierz wszystkie transakcje          |
| <b>GetDeal</b>           | <b>aId:</b> identyfikator transakcji | Pobierz transakcję po identyfikatorze |
| <b>UpdateDeal</b>        | <b>aId:</b> identyfikator transakcji | Zaktualizuj umowę                     |
| <b>CancelDeal</b>        | <b>aId:</b> identyfikator transakcji | Anulowanie transakcji                 |
| <b>CloseAtMarketDeal</b> | <b>aId:</b> identyfikator transakcji | Zamknij transakcję po cenie rynkowej  |

## Bot siatkowy

|                       |                                |                                           |
|-----------------------|--------------------------------|-------------------------------------------|
| <b>CreateGridBot</b>  |                                | Utwórz nowego bota siatki                 |
| <b>GetGridBot</b>     | <b>aId:</b> identyfikator bota | Pobierz bota siatki według identyfikatora |
| <b>GetGridBots</b>    |                                | Pobierz wszystkie boty siatki             |
| <b>EnableGridBot</b>  | <b>aId:</b> identyfikator bota | Włącz bota siatki                         |
| <b>DisableGridBot</b> | <b>aId:</b> identyfikator bota | Wyłącz bota siatki                        |
| <b>DeleteGridBot</b>  | <b>aId:</b> identyfikator bota | Usuń bota siatki                          |

## Zdarzenia

### OnConnect

Gdy nowe połączenie WebSocket jest otwarte

### OnDisconnect

Po zamknięciu połączenia WebSocket

## **OnThreeCommasConnect**

Gdy klient odbierze wiadomość Welcome z serwera 3Commas, oznacza to, że połączenie jest gotowe.

## **OnThreeCommasConfirmSubscription**

Potwierdza poprzednią subskrypcję wysłaną przez klienta.

## **OnThreeCommasRejectSubscription**

Wystąpił błąd podczas próby subskrypcji kanału 3Commas.

## **OnThreeCommasMessage**

Tutaj klient odbiera dane wysyłane przez serwer związane z subskrybowanymi kanałami

## **OnThreeCommasPing**

Ping wysłany przez serwer do klienta.

## **OnThreeCommasHTTPException**

Jeśli podczas wywoływania metod HTTP REST wystąpi błąd, zostanie wywołane to zdarzenie.

# API OKX

## OKX

### Obsługiwane API

- **WebSockets API:** łączy się z serwerem WebSocket i dostarcza aktualizacje danych rynkowych w czasie rzeczywistym, zmiany na koncie oraz umożliwia składanie zleceń handlowych.

### Właściwości

Kanały WebSocket są podzielone na dwie kategorie: publiczne i prywatne.

- **Kanały publiczne:** obejmują kanał tickerów, kanał K-Line, kanał ceny limitowej, kanał księgi zleceń oraz kanał ceny mark itp. — nie wymagają logowania.
- **Kanały prywatne:** w tym kanał konta, kanał zleceń i kanał pozycji itd. — wymagają zalogowania.

W właściwości OKX można skonfigurować następujące właściwości.

- **ApiKey:** nowy klucz API można wygenerować na koncie OKX; należy skopiować jego wartość do tej właściwości.
- **ApiSecret:** to tajny klucz API.
- **Passphrase:** niestandardowy ciąg znaków zdefiniowany podczas tworzenia nowego klucza API.
- **IsDemo:** po włączeniu następuje połączenie z kontem demonstracyjnym OKX (domyślnie wyłączone).
- **IsPrivate:** jeśli jest włączone, możliwe jest połączenie z kanałami prywatnymi (domyślnie wyłączone).
- **IsBusiness:** jeśli włączone, komponent połączy się z biznesowym endpointem WebSocket (wss://ws.okx.com:8443/ws/v5/business). Należy używać dla kanałów świecowych, kanałów zleceń algorytmicznych i innych zaawansowanych kanałów (domyślnie wyłączone).

### Połączenie

Gdy klient pomyślnie połączy się z serwerami OKX, wywoływane jest zdarzenie **OnOKXConnect**. Jeśli wystąpi błąd podczas próby połączenia, zdarzenie **OnOKXError** zostanie wywołane ze szczegółami błędu.

Po wywołaniu zdarzenia **OnOKXConnect** można rozpocząć **wysyłanie i odbieranie wiadomości** z serwerów OKX.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_OKX *oOKX = new TsgcWSAPI_OKX(NULL);
oOKX->Client = oClient;
oOKX->OKX->ApiKey = "alsdjk23kandfnasbdfdkjhdsf";
oOKX->OKX->ApiSecret = "aldskjfk3jkadknfajndsffj23j";
oOKX->OKX->Passphrase = "secret_passphrase";
oClient->Active = true;
void OnOKXConnect(TObject *Sender, string aMessage, string aCode, string aRawMessage)
{
    DoLog("#OKX Connected");
}
void OnOKXError(TObject *Sender, string aCode, string aMessage, string aRawMessage)
{
    DoLog("#error: " + aMessage);
}
```

## Kanały publiczne

Kanał WebSocket dostarcza aktualizacje rynkowe w czasie rzeczywistym dla zleceń i transakcji. Kanał WebSocket posiada publiczne kanały takie jak ticker, trades...

Możliwa jest subskrypcja następujących **kanałów publicznych**:

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SubscribeInstruments</b>           | Pełna lista instrumentów zostanie wypchnięta po raz pierwszy po subskrypcji. Następnie instrumenty będą wypychane, jeśli nastąpi jakakolwiek zmiana stanu instrumentu (np. dostawa FUTURES, wykonanie OPTION, notowanie nowych kontraktów / par handlowych, zawieszenie handlu itp.).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>SubscribeTicker</b>                | Pobierz ostatnią cenę transakcji, cenę kupna, cenę sprzedaży oraz 24-godzinny wolumen obrotu dla instrumentów. Dane są wypychane co 100 ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>SubscribeOpenInterest</b>          | Pobierz otwarte zainteresowanie. Dane będą wypychane co 3 sekundy.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>SubscribeCandlestick</b>           | Pobiera dane świec dla instrumentu. Dane są przesyłane z maksymalną częstotliwością co 500 ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>SubscribeTrades</b>                | Pobiera dane o ostatnich transakcjach. Dane są przesyłane za każdym razem, gdy dojdzie do transakcji.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>SubscribeEstimatedPrices</b>       | <p>Pobierz szacowaną cenę dostawy/wykonania kontraktów FUTURES oraz OPTION.</p> <p>Przybliżona cena dostawy/wykonania będzie przesyłana wyłącznie na godzinę przed dostawą/wykonaniem i tylko w przypadku zmiany ceny.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>SubscribeMarkPrice</b>             | Pobiera cenę mark. Dane są przesyłane co 200 ms przy zmianie ceny mark oraz co 10 sekund, gdy cena mark nie zmienia się.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SubscribeMarkPrice-Candlestick</b> | Pobiera dane wykresów świecowych dla ceny mark. Dane są wypychane co 500 ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>SubscribePriceLimit</b>            | Pobiera maksymalną cenę kupna i minimalną cenę sprzedaży instrumentu. Dane są przekazywane co 5 sekund w przypadku zmian limitów; gdy nie ma zmian, dane nie są wysyłane.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>SubscribeOrderBook</b>             | <p>Pobiera dane z arkusza zleceń.</p> <p>Należy używać books dla 400 poziomów głębokości, book5 dla 5 poziomów głębokości, bbo-tbt dla tick-by-tick z 1 poziomem głębokości, books50-12-tbt dla tick-by-tick z 50 poziomami głębokości oraz books-12-tbt dla tick-by-tick z 400 poziomami głębokości.</p> <ul style="list-style-type: none"><li>• <b>books:</b> Pełna migawka z 400 poziomami głębokości zostanie wypchnięta na początku. Dane przyrostowe będą wypychane co 100 ms przy każdej zmianie arkusza zleceń.</li><li>• <b>books5:</b> 5 poziomów głębokości będzie wypychanych za każdym razem. Dane będą wypychane co 100 ms przy każdej zmianie arkusza zleceń.</li><li>• <b>bbo-tbt:</b> 1 poziom głębokości będzie wypychany za każdym razem. Dane będą wypychane co 10 ms przy każdej zmianie arkusza zleceń.</li><li>• <b>books-12-tbt:</b> Pełna migawka z 400 poziomami głębokości zostanie wypchnięta na początku. Dane przyrostowe będą wypychane co 10 ms przy każdej zmianie arkusza zleceń.</li><li>• <b>books50-12-tbt:</b> Pełna migawka z 50 poziomami głębokości zostanie wypchnięta na początku. Dane przyrostowe będą wypychane co 10 ms przy każdej zmianie arkusza zleceń. Jeśli asks lub bids jest pustą tablicą,</li></ul> |

oznacza to zmiany na głębokości 400, a nie 50. W przypadku lokalnego przechowywania danych arkusza zleceń należy ignorować puste asks i bids.

|                                            |                                                                                                                                                                                                                          |
|--------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SubscribeOptionSummary</b>              | Pobiera szczegółowe informacje o cenach wszystkich kontraktów OPTION. Dane zostaną przesłane jednorazowo.                                                                                                                |
| <b>SubscribeFundingRate</b>                | Pobierz stopę finansowania. Dane będą przesyłane co 30–90 sekund.                                                                                                                                                        |
| <b>SubscribeIndexCandlestick</b>           | Pobierz dane świecowe indeksu. Dane będą wypychane co 500 ms.                                                                                                                                                            |
| <b>SubscribeIndexTicker</b>                | Pobiera dane tickerów indeksowych                                                                                                                                                                                        |
| <b>SubscribeStatus</b>                     | Pobierz status konserwacji systemu i wypychaj dane po zmianie statusu konserwacji. Pierwsze subskrybowanie: „Wypychaj najnowsze dane o zmianach”; przy każdej zmianie stanu wypychana jest zmieniona zawartość           |
| <b>SubscribePublicStructureBlockTrades</b> | Dane są przesyłane za każdym razem, gdy dochodzi do transakcji blokowej.                                                                                                                                                 |
| <b>SubscribeBlockTickers</b>               | Pobiera najnowszy wolumen transakcji blokowych z ostatnich 24 godzin. Dane są przesyłane po wyzwoleniu przez zdarzenie wykonania transakcji. Ponadto dane są przesyłane co 5 minut zgodnie z granularnością subskrypcji. |
| <b>SubscribeAllTrades</b>                  | Pobiera wszystkie dane transakcji. Dane są przesyłane w momencie każdej transakcji.                                                                                                                                      |
| <b>SubscribeLiquidationOrders</b>          | Pobierz zlecenia likwidacji. Dane będą przesyłane po pojawieniu się zlecenia likwidacji.                                                                                                                                 |
| <b>SubscribeADLWarning</b>                 | Pobierz dane ostrzegawcze ADL. Dane są przesyłane, gdy ryzyko automatycznego delewarowania wzrasta.                                                                                                                      |
| <b>SubscribeEconomicCalendar</b>           | Pobiera zdarzenia kalendarza ekonomicznego. Dane są przesyłane w momencie aktualizacji zdarzeń ekonomicznych.                                                                                                            |
| <b>SubscribePublicBlockTrades</b>          | Pobierz publiczne transakcje blokowe. Dane będą przesyłane po każdej transakcji blokowej.                                                                                                                                |
| <b>SubscribeOptionTrades</b>               | Pobierz dane transakcji opcji. Dane są przesyłane za każdym razem, gdy nastąpi transakcja opcji.                                                                                                                         |
| <b>SubscribeCallAuctionDetails</b>         | Pobierz szczegóły aukcji call. Dane będą przesyłane po aktualizacji informacji o aukcji call.                                                                                                                            |

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_OKX *oOKX = new TsgcWSAPI_OKX(NULL);
oOKX->Client = oClient;
oOKX->OKX->ApiKey = "alsdjk23kandfnasbdfdkjhdsdf";
oOKX->OKX->ApiSecret = "aldskjfk3jkadknfajndsjfj23j";
oOKX->OKX->Passphrase = "secret_passphrase";
oClient->Active = true;
void OnOKXConnect(TObject *Sender, string aMessage, string aCode, string aRawMessage)
{
    oOKX->SubscribeInstruments(okxitFutures);
}
```

## Kanały prywatne

Kanały wymagające logowania: kanał konta, kanał zleceń, kanał pozycji i inne.

Można subskrybować następujące **kanały prywatne**:

|                         |                                                                                                                                                                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SubscribeAccount</b> | Pobieranie informacji o koncie. Dane będą przesyłane po wyzwoleniu przez zdarzenia takie jak złożenie zlecenia, anulowanie zlecenia, wykonanie transakcji itp. Będą również przesyłane w regularnych odstępach czasu zgodnie z granularnością subskrypcji. |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                                             |                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SubscribePositions</b>                   | Pobierz informacje o pozycji. Początkowa migawka zostanie przesłana zgodnie z granularnością subskrypcji. Dane będą przesyłane po wyzwoleniu zdarzeń takich jak składanie/anulowanie zlecenia, a także regularnie zgodnie z granularnością subskrypcji.                                                         |
| <b>SubscribeBalanceAnd-Position</b>         | Pobierz informacje o saldzie i pozycjach konta. Dane będą przesyłane po wyzwoleniu zdarzeń, takich jak zrealizowane zlecenie lub transfer środków.                                                                                                                                                              |
| <b>SubscribeOrders</b>                      | Pobieranie informacji o zleceniu. Dane nie są przesyłane przy pierwszej subskrypcji. Dane są przesyłane tylko po wyzwoleniu przez zdarzenia takie jak złożenie lub anulowanie zlecenia.                                                                                                                         |
| <b>SubscribeOrdersAlgo</b>                  | Pobiera zlecenia algo (obejmuje zlecenia trigger, oco i warunkowe). Dane nie są przesyłane po pierwszej subskrypcji. Dane są przesyłane wyłącznie po wyzwoleniu przez zdarzenia, takie jak złożenie/anulowanie zlecenia.                                                                                        |
| <b>SubscribeAdvanceAlgo</b>                 | Pobierz zaawansowane zlecenia algorytmiczne (w tym zlecenia Iceberg, TWAP, Trailing). Dane są przesyłane po pierwszej subskrypcji oraz po wystąpieniu zdarzeń takich jak złożenie lub anulowanie zlecenia.                                                                                                      |
| <b>SubscribePositionRisk</b>                | Ten kanał push jest używany wyłącznie jako ostrzeżenie o ryzyku i nie jest zalecany jako podstawa oceny ryzyka w handlu strategicznym.<br>W przypadku gdy rynek nie porusza się gwałtownie, istnieje możliwość, że pozycja została zlikwidowana w tym samym momencie, w którym ta wiadomość została wypchnięta. |
| <b>SubscribeAccountGre-eks</b>              | Pobieranie informacji o greckich parametrach konta. Dane są przesyłane po wyzwoleniu przez zdarzenia, takie jak zwiększenie lub zmniejszenie pozycji albo salda gotówkowego na koncie, a także są przesyłane w regularnych odstępach czasu zgodnie z granularnością subskrypcji.                                |
| <b>SubscribeRfqqs</b>                       | Pobierz zapytania RFQ.                                                                                                                                                                                                                                                                                          |
| <b>SubscribeQuotes</b>                      | Pobierz notowania.                                                                                                                                                                                                                                                                                              |
| <b>SubscribePrivateStructureBlockTrades</b> | Pobierz transakcje blokowe struktury.                                                                                                                                                                                                                                                                           |
| <b>SubscribeSpotGridAlgo-Orders</b>         | Pobierz zlecenia algorytmiczne siatki spot. Dane są wypychane przy pierwszej subskrypcji. Dane są wypychane po wyzwoleniu przez zdarzenia, takie jak złożenie/anulowanie zlecenia.                                                                                                                              |
| <b>SubscribeContactGridAlgoOrders</b>       | Pobierz zlecenia algorytmiczne siatki kontraktów. Dane są przesyłane przy pierwszej subskrypcji oraz w reakcji na zdarzenia, takie jak składanie lub anulowanie zleceń.                                                                                                                                         |
| <b>SubscribeGridPositions</b>               | Pobiera pozycje siatki. Dane są przesyłane po pierwszej subskrypcji. Dane są przesyłane po wyzwoleniu przez zdarzenia, takie jak złożenie/anulowanie zlecenia.                                                                                                                                                  |
| <b>SubscribeGridSubOrders</b>               | Pobiera podzlecenia siatki. Dane są przesyłane po pierwszej subskrypcji. Dane są przesyłane po wyzwoleniu przez zdarzenia, takie jak złożenie zlecenia.                                                                                                                                                         |
| <b>SubscribeFills</b>                       | Pobiera dane zrealizowanych zleceń. Dane są wypychane po realizacji zlecenia.                                                                                                                                                                                                                                   |
| <b>SubscribeDepositInfo</b>                 | Pobranie informacji o wpłacie. Dane zostaną przesłane po aktualizacji statusu wpłaty.                                                                                                                                                                                                                           |
| <b>SubscribeWithdrawalInfo</b>              | Pobiera informacje o wypłacie. Dane są przekazywane w przypadku aktualizacji statusu wypłaty.                                                                                                                                                                                                                   |

```

TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_OKX *oOKX = new TsgcWSAPI_OKX(NULL);
oOKX->Client = oClient;
oOKX->OKX->ApiKey = "alsdjk23kandfnasbdfdkjhdsf";
oOKX->OKX->ApiSecret = "aldskjfk3jkadknfajnds fj23j";
oOKX->OKX->Passphrase = "secret_passphrase";
oClient->Active = true;
void OnOKXConnect(TObject *Sender, string aMessage, string aCode, string aRawMessage)
{
    oOKX->SubscribeOrders(okxitFutures, "BTC-USD", "BTC-USD-200329");
}

```

## Handel

WebSocket Trade wymaga **uwierzytelniania**.

Zlecenie można złożyć tylko w przypadku posiadania wystarczających środków. Poniżej znajduje się tabela z parametrami żądania:

| Parametr     | Typ     | Wymagane   | Opis                                                                                                                                                                                                                                                                  |
|--------------|---------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id           | String  | Tak        | Unikalny identyfikator wiadomości<br>Dostarczany przez klienta. Zostanie zwrócony w wiadomości odpowiedzi do identyfikacji odpowiedniego żądania.<br>Kombinacja alfanumeryczna z uwzględnieniem wielkości liter, same cyfry lub same litery, o długości do 32 znaków. |
| > instId     | String  | Tak        | Identyfikator instrumentu, np. BTC-USD-190927-5000-C                                                                                                                                                                                                                  |
| > tdMode     | String  | Tak        | Tryb handlu<br>Tryb marży isolated cross<br>Tryb bez marży cash                                                                                                                                                                                                       |
| > ccy        | String  | Nie        | Waluta depozytu zabezpieczającego<br>Dotyczy wyłącznie cross MARGIN zleceń w Single-currency margin.                                                                                                                                                                  |
| > clOrdId    | String  | Nie        | Identyfikator zlecenia dostarczony przez klienta<br>Kombinacja znaków alfanumerycznych z rozróżnieniem wielkości liter, składająca się wyłącznie z cyfr lub wyłącznie z liter, o długości do 32 znaków.                                                               |
| > tag        | String  | Nie        | Etykieta zlecenia<br>Kombinacja rozróżniających wielkość liter znaków alfanumerycznych, samych cyfr lub samych liter, o długości do 16 znaków.                                                                                                                        |
| > side       | String  | Tak        | Strona zlecenia, buy sell                                                                                                                                                                                                                                             |
| > posSide    | String  | Opcjonalne | Strona pozycji<br>Domyślnie jest to net w trybie net<br>Jest wymagana w trybie long/short i może wynosić wyłącznie long lub short.<br>Dotyczy wyłącznie FUTURES/SWAP.                                                                                                 |
| > ordType    | String  | Tak        | Typ zlecenia<br>market: zlecenie rynkowe<br>limit: zlecenie z limitem ceny<br>post_only: zlecenie post-only<br>fok: zlecenie fill-or-kill<br>ioc: zlecenie immediate-or-cancel<br>optimal_limit_ioc : zlecenie rynkowe immediate-or-cancel                            |
| > sz         | String  | Tak        | Ilość do kupienia lub sprzedaży.                                                                                                                                                                                                                                      |
| > px         | String  | Opcjonalne | Cena<br>Dotyczy wyłącznie zlecenia typu limit,post_only,fok,ioc .                                                                                                                                                                                                     |
| > reduceOnly | Boolean | Nie        | Czy zmniejszyć tylko pozycję, true false, wartość domyślna to false.<br>Dotyczy wyłącznie zleceń MARGIN oraz zleceń FUTURES/SWAP w trybie net<br>Dotyczy wyłącznie Single-currency margin oraz Multi-currency margin                                                  |
| > tgtCcy     | String  | Nie        | Typ ilości<br>base_ccy: Waluta bazowa ,quote_ccy: Waluta kwotowana<br>Dotyczy wyłącznie SPOT handlowanego zleceniem rynkowym<br>Domyślnie quote_ccy dla kupna, base_ccy dla sprzedaży                                                                                 |

| Parametr   | Typ     | Wymagane | Opis                                                                                                                                                                                                                 |
|------------|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| > banAmend | Boolean | Nie      | Czy zakazać modyfikowania zlecenia spot, true lub false, domyślnie false. Złożenie zlecenia nie powiedzie się, jeśli saldo jest niewystarczające, gdy banAmend wynosi true. Dotyczy wyłącznie zleceń rynkowych SPOT. |

## Przykład złożenia zlecenia

Zlecenie można złożyć tylko wtedy, gdy dostępne są wystarczające środki.

```
// Place Market Order
TsgcWSAPI_OKX1->PlaceMarketOrder(okxosBuy, "ETH-BTC", 1);
// Place Limit Order
TsgcWSAPI_OKX1->PlaceLimitOrder(okxosBuy, "ETH-BTC", 1, 0.25);
```

## Przykład anulowania zlecenia

Anuluj niekompletne zlecenie

```
TsgcWSAPI_OKX1->CancelOrder("ETH-BTC", "457589362405027840");
```

## Zmiana zlecenia

Zmień niekompletne zlecenie.

```
TsgcWSAPI_OKX1->AmendOrder("ETH-BTC", "457589362405027840", "", 2);
```

## Wsadowe operacje handlowe

WebSocket Trade API obsługuje również operacje wsadowe umożliwiające jednoczesne składanie, anulowanie i modyfikowanie wielu zleceń. Operacje te wymagają **uwierzytelnienia**.

|                          |                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>BatchPlaceOrders</b>  | Składa wiele zleceń w jednym żądaniu. Jednorazowo można złożyć maksymalnie 20 zleceń.                            |
| <b>BatchCancelOrders</b> | Anuluje wiele zleceń w jednym żądaniu. Maksymalnie 20 zleceń może zostać anulowanych jednocześnie.               |
| <b>BatchAmendOrders</b>  | Modyfikuj wiele niezrealizowanych zleceń w jednym żądaniu. Jednorazowo można zmodyfikować maksymalnie 20 zleceń. |
| <b>MassCancelOrders</b>  | Masowe anulowanie wszystkich oczekujących zleceń dla określonego typu instrumentu.                               |



# API XTB

## XTB

### Obsługiwane API

- [WebSockets API](#): łączy się z serwerem WebSocket i dostarcza aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta oraz możliwość składania zleceń handlowych.

### Właściwości

Protokół WebSocket umożliwia 2 typy żądań: **polecenia strumieniowe** (odbieranie aktualizacji na żywo) oraz **pobieranie danych transakcyjnych** (wysyłanie żądania do serwera w celu uzyskania określonych informacji).

W właściwości XTB można skonfigurować następujące parametry.

- **User**: nazwa użytkownika identyfikująca połączenie.
- **Password**: to tajne hasło użytkownika.
- **Demo**: jeśli włączone, nastąpi połączenie z kontem demonstracyjnym XTB (domyślnie wyłączone).

### Połączenie

Gdy klient pomyślnie połączy się z serwerami XTB, wywoływane jest zdarzenie **OnXTBConnect**. Jeśli podczas próby połączenia wystąpi błąd, zdarzenie **OnXTBError** zostanie wywołane ze szczegółami błędu.

Po wywołaniu zdarzenia **OnXTBConnect** można rozpocząć **wysyłanie i odbieranie wiadomości** z serwerów XTB.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_XTB *oXTB = new TsgcWSAPI_XTB(NULL);
oXTB->Client = oClient;
oXTB->XTB->User = "user_0001";
oXTB->XTB->Password = "secret_0001";
oClient->Active = true;
void OnXTBConnect(TObject *Sender, string aStreamSessionId)
{
    DoLog("#XTB Connected");
}
void OnXTBError(TObject *Sender, string aCode, string aDescription, string aRawMessage)
{
    DoLog("#error: " + aDescription);
}
```

## Polecenia połączenia

### Login

Aby wykonać jakąkolwiek operację, aplikacja kliencka musi przeprowadzić proces logowania. Żadna funkcjonalność nie jest dostępna przed prawidłowym zalogowaniem. Metoda logowania jest wywoływana automatycznie po połączeniu klienta z serwerem WebSocket, jeśli wartości User/Password są ustawione.

### Wylogowanie

## Polecenia strumieniowania

Można subskrybować następujące kanały:

### SubscribeBalance

Umożliwia pobieranie aktualnych wartości wskaźników konta w czasie rzeczywistym, gdy tylko staną się dostępne w systemie.

### SubscribeCandles

Subskrybuje i anuluje subskrypcję świec wykresu API. Interwał każdej świecy wynosi 1 minutę. Nowa świeca pojawia się co minutę.

### SubscribeKeepAlive

Subskrybuje i anuluje subskrypcję wiadomości „keep alive”. Nowa wiadomość „keep alive” jest wysyłana przez API co 3 sekundy.

### SubscribeNews

Subskrybuje i anuluje subskrypcję wiadomości.

### SubscribeProfits

Subskrybuje zyski i anuluje subskrypcję zysków.

### SubscribeTickPrices

Ustanawia subskrypcję kwotowań i umożliwia uzyskiwanie odpowiednich informacji w czasie rzeczywistym, natychmiast po ich pojawieniu się w systemie. Polecenie `getTickPrices` można wywoływać wielokrotnie dla tego samego symbolu, jednak dla danego symbolu zostanie utworzona tylko jedna subskrypcja. Należy mieć na uwadze, że gdy dostępnych jest wiele rekordów, kolejność ich otrzymywania nie jest gwarantowana.

### SubscribeTrades

Ustanawia subskrypcję danych statusu transakcji użytkownika i umożliwia uzyskanie odpowiednich informacji w czasie rzeczywistym, gdy tylko staną się dostępne w systemie. Należy pamiętać, że gdy dostępnych jest wiele rekordów, kolejność ich odbierania nie jest gwarantowana.

### SubscribeTradeStatus

Umożliwia pobieranie statusu wysłanych żądań handlowych w czasie rzeczywistym, natychmiast gdy jest dostępny w systemie. Należy pamiętać, że gdy dostępnych jest wiele rekordów, kolejność ich odbierania nie jest gwarantowana.

### SubscribePing

Regularne wywoływanie tej funkcji wystarczy do odświeżenia wewnętrznego stanu wszystkich komponentów w systemie. Połączenie strumieniowe, gdy klient nie wysyła żadnego polecenia w trakcie sesji, generuje ruch sieciowy tylko w jednym kierunku. Zaleca się, aby każda aplikacja, która nie wykonuje innych poleceń, wywoływała to polecenie co najmniej raz na 10 minut.

## Pobieranie danych handlowych

Można wysyłać następujące zapytania:

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetAllSymbols</b>        | Zwraca tablicę wszystkich symboli dostępnych dla użytkownika.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>GetCalendar</b>          | Zwraca kalendarz z wydarzeniami rynkowymi.<br>Należy pamiętać, że ta funkcja może być zazwyczaj zastąpiona jej odpowiednikiem strumieniowym <code>getCandles</code> , który jest zalecanym sposobem pobierania bieżących danych świecowych. Zwraca informacje o wykresie, od daty początkowej do bieżącego czasu. Jeśli wybrany okres <code>CHART_LAST_INFO_RECORD</code> jest większy niż 1 minuta, ostatnia świeca zwrócona przez API może zmieniać się do końca okresu (świeca jest automatycznie aktualizowana co minutę). |
| <b>GetChartLastRequest</b>  | Należy pamiętać, że tę funkcję można zazwyczaj zastąpić jej odpowiednikiem strumieniowym <code>getCandles</code> , który jest preferowanym sposobem pobierania bieżących danych świecowych. Zwraca informacje o wykresie z danymi między podanymi datami początkowymi i końcowymi.                                                                                                                                                                                                                                             |
| <b>GetChartRangeRequest</b> | Zwraca obliczenie prowizji i kursu wymiany. Wartość jest obliczana jako wartość oczekiwana i dlatego może nie być w pełni dokładna.                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>GetCommissionDef</b>     | Zwraca informacje o walucie konta oraz dźwigni finansowej konta.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>GetCurrentUserData</b>   | Zwraca dane IB z podanego zakresu czasowego.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>GetIbsHistory</b>        | Należy pamiętać, że funkcja ta może być zwykle zastąpiona przez odpowiednik strumieniowy <code>getBalance</code> , który jest preferowanym sposobem pobierania wskaźników konta. Zwraca różne wskaźniki konta.                                                                                                                                                                                                                                                                                                                 |
| <b>GetMarginLevel</b>       | Zwraca oczekiwany depozyt zabezpieczający dla danego instrumentu i wolumenu. Wartość jest obliczana jako oczekiwana wartość depozytu i dlatego może nie być w pełni dokładna.                                                                                                                                                                                                                                                                                                                                                  |
| <b>GetMarginTrade</b>       | Należy pamiętać, że tę funkcję można zazwyczaj zastąpić jej odpowiednikiem strumieniowym <code>getNews</code> , który jest preferowanym sposobem pobierania danych o wiadomościach. Zwraca wiadomości z serwera handlowego, które zostały wysłane w określonym przedziale czasowym.                                                                                                                                                                                                                                            |
| <b>GetNews</b>              | Oblicza szacowany zysk dla podanych danych transakcji. Należy stosować wyłącznie w aplikacjach typu kalkulator. Zysk dla otwartych transakcji powinien być pobierany z serwera ze względu na większą precyzję obliczeń serwerowych.                                                                                                                                                                                                                                                                                            |
| <b>GetProfitCalculation</b> | Zwraca bieżący czas na serwerze handlowym                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>GetServerTime</b>        | Zwraca listę reguł krokowych dla DMA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>GetStepRules</b>         | Zwraca informacje o symbolu dostępnym dla użytkownika                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>GetSymbol</b>            | Należy pamiętać, że tę funkcję można zazwyczaj zastąpić jej odpowiednikiem strumieniowym <code>getTickPrices</code> , który jest preferowanym sposobem pobierania danych o transakcjach. Zwraca tablicę bieżących kwotowań dla podanych symboli — zwracane są wyłącznie kwotowania, które zmieniły się od podanego znacznika czasu. Nowy znacznik czasu uzyskany z odpowiedzi będzie użyty jako argument kolejnego wywołania tego polecenia.                                                                                   |
| <b>GetTickPrices</b>        | Zwraca tablicę transakcji wymienionych w argumencie <code>orders</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>GetTradeRecords</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetTrades</b>              | Należy pamiętać, że ta funkcja może być zazwyczaj zastąpiona przez jej odpowiednik strumieniowy <code>getTrades</code> , który jest preferowanym sposobem pobierania danych o transakcjach. Zwraca tablicę transakcji użytkownika.                                                                                                                                                                                                                  |
| <b>GetTradesHistory</b>       | Należy pamiętać, że tę funkcję można zwykle zastąpić jej odpowiednikiem strumieniowym <code>getTrades</code> , który jest preferowanym sposobem pobierania danych o transakcjach. Zwraca tablicę transakcji użytkownika zamkniętych w określonym przedziale czasu.                                                                                                                                                                                  |
| <b>GetTradingHours</b>        | Zwraca notowania i godziny handlu.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>GetVersion</b>             | Zwraca bieżącą wersję API.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Ping</b>                   | Regularne wywoływanie tej funkcji jest wystarczające do odświeżenia wewnętrznego stanu wszystkich komponentów w systemie. Zaleca się, aby każda aplikacja, która nie wykonuje innych poleceń, wywoływała to polecenie przynajmniej raz na 10 minut. Należy pamiętać, że strumieniowy odpowiednik tej funkcji to kombinacja poleceń <code>ping</code> i <code>getKeepAlive</code> .                                                                  |
| <b>TradeTransaction</b>       | Rozpoczyna transakcję handlową. <code>tradeTransaction</code> wysyła główne informacje o transakcji do serwera.                                                                                                                                                                                                                                                                                                                                     |
| <b>TradeTransactionStatus</b> | Należy pamiętać, że tę funkcję można zwykle zastąpić jej odpowiednikiem strumieniowym <code>getTradeStatus</code> , który jest preferowanym sposobem pobierania danych o statusie transakcji. Zwraca bieżący status transakcji. W dowolnym momencie przetwarzania transakcji klient może sprawdzić status transakcji po stronie serwera. W tym celu klient musi podać unikatowy numer zlecenia uzyskany z wywołania <code>tradeTransaction</code> . |

# API Bybit

Bybit

## Obsługiwane API

- **WebSocket API:** łączy się z serwerem websocket i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta i inne informacje.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

Aktualnie obsługiwana wersja API to V5. API V5 wprowadza jednolitość i wydajność dla linii produktów Bybit, łącząc Spot, Derivatives i Options w jeden zestaw specyfikacji.

| Wersja<br>OpenAPI | Typ konta                      | USDT Per-<br>petual | Liniowy             |                   | Odwrotny          |              | Spot | Opcje |
|-------------------|--------------------------------|---------------------|---------------------|-------------------|-------------------|--------------|------|-------|
|                   |                                |                     | USDC Per-<br>petual | USDC Fu-<br>tures | Beztermi-<br>nowy | Futu-<br>res |      |       |
| V5                | Ujednolicone<br>konto handlowe | ✓                   | ✓                   | ✓                 |                   | patrz uwaga  | ✓    | ✓     |
|                   | Konto klasyczne                | ✓                   |                     |                   | ✓                 | ✓            | ✓    |       |
| V3                | Ujednolicone<br>konto handlowe | ✓                   | ✓                   |                   |                   |              |      | ✓     |
|                   | Konto klasyczne                | ✓                   |                     |                   | ✓                 | ✓            | ✓    |       |

\*Uwaga: konto Unified obsługuje handel odwrócony. Jednak używany depozyt zabezpieczający pochodzi z portfela pochodnych odwróconych, a nie z portfela unified.

## Właściwości

Następujące właściwości można skonfigurować we właściwości Bybit.

- **ApiKey:** klucz API można uzyskać na koncie Bybit. Należy skopiować jego wartość do tej właściwości. Jeśli APIKey jest ustawiony, klient połączy się z prywatnym serwerem WebSocket. Jeśli pole jest puste, klient połączy się z publicznym serwerem WebSocket.
- **ApiSecret:** to tajny klucz API.
- **SignatureExpires:** liczba sekund, po których podpis wygasa (domyślnie 10 sekund).
- **TestNet:** jeśli włączone, połączy się z kontem demonstracyjnym Bybit TestNet (domyślnie wyłączone).

## Połączenie

Gdy klient pomyślnie połączy się z serwerami Bybit, wywoływane jest zdarzenie **OnConnect**. Po jego wywołaniu można rozpocząć **wysyłanie i odbieranie wiadomości** do/z serwerów Bybit. W przypadku łączenia się z prywatnym kanałem WebSocket należy poczekać na wywołanie zdarzenia **OnBybitAuthentication** i sprawdzić, czy parametr success ma wartość true, przed subskrypcją jakiegokolwiek kanału.

Klient obsługuje kilka API; właściwość BybitClient służy do określenia, które API ma być używane:

- **bybSpot**
- **bybInverse**
- **bybLinear**
- **bybPerpetual**

Poniżej znajduje się przykład połączenia z prywatnym WebSocket Spot API.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bybit *oBybit = new TsgcWSAPI_Bybit(NULL);
oBybit->Client = oClient;
oBybit->Bybit->ApiKey = "alsdjk23kandfnasbdfdkjhdsf";
oBybit->Bybit->ApiSecret = "alidskjfk3jkdaknfajndsifj23j";
oBybit->BybitClient = bybSpot;
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#Bybit Connected");
}
```

Po pomyślnym połączeniu ze Spot WebSocket Server można rozpocząć subskrybowanie kanałów WebSocket. Wystarczy uzyskać dostęp do właściwości **SPOT**, a następnie wywołać dowolną z dostępnych metod subskrypcji/anulowania subskrypcji.

## Zdarzenia

Klient bybit implementuje następujące zdarzenia do kontrolowania przepływu połączenia i odbierania danych wysyłanych przez serwer WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnBybitAuthentication:** wywoływane, gdy klient uwierzytelnia się na prywatnym serwerze WebSocket Bybit.
- **OnBybitSubscribe:** wywoływane, gdy klient subskrybuje kanał WebSocket.
- **OnBybitUnSubscribe:** wywoływane, gdy klient anuluje subskrypcję kanału websocket.
- **OnBybitData:** wywoływane, gdy klient otrzymuje dane z serwera.
- **OnBybitError:** gdy wystąpi błąd podczas połączenia websocket bybit.
- **OnBybitHTTPException:** wywoływane w przypadku wystąpienia błędu podczas żądania REST.

## WebSocket API

Kanał WebSocket dostarcza aktualizacje rynkowe w czasie rzeczywistym dla zleceń i transakcji. Kanał WebSocket posiada publiczne kanały takie jak ticker, trades...

Można subskrybować następujące **kanały**:

|                                 |           |                                                                  |
|---------------------------------|-----------|------------------------------------------------------------------|
| <b>SubscribeOrderBook</b>       | Publiczny | Subskrybuje strumień arkusza zleceń. Obsługuje różne głębokości. |
| <b>SubscribeTrade</b>           | Publiczny | Subskrybuj strumień ostatnich transakcji.                        |
| <b>SubscribeTicker</b>          | Publiczny | Subskrybuj strumień kursów.                                      |
| <b>SubscribeKLine</b>           | Publiczny | Subskrybuj strumień klines.                                      |
| <b>SubscribeLiquidation</b>     | Publiczny | Subskrybuj strumień likwidacji                                   |
| <b>SubscribeLT_KLine</b>        | Publiczny | Subskrybuj strumień kline tokenów z dźwignią.                    |
| <b>SubscribeLT_Ticker</b>       | Publiczny | Subskrybuj strumień tickera tokenu lewarowanego.                 |
| <b>SubscribeLT_Nav</b>          | Publiczny | Subskrybuj strumień tickera tokenu lewarowanego.                 |
| <b>SubscribePosition</b>        | Prywatny  | Subskrybuj strumień NAV tokenów z dźwignią.                      |
| <b>SubscribeExecution</b>       | Prywatny  | Subskrybuj                                                       |
| <b>SubscribeOrder</b>           | Prywatny  | Subskrybuj                                                       |
| <b>SubscribeWallet</b>          | Prywatny  | Subskrybuj                                                       |
| <b>SubscribeGreek</b>           | Prywatny  | Subskrybuj                                                       |
| <b>SubscribeDcp</b>             | Prywatny  | Subskrybuj                                                       |
| <b>SubscribeInsurance</b>       | Publiczny | Subskrybuj strumień funduszu ubezpieczeniowego.                  |
| <b>SubscribeOrderPriceLimit</b> | Publiczny | Subskrybuj strumień limitu ceny zlecenia.                        |
| <b>SubscribeADLAlert</b>        | Publiczny | Subskrybowanie strumienia alertów automatycznego delewarowania.  |
| <b>SubscribeFastExecution</b>   | Prywatny  | Subskrybuj szybki strumień zleceń.                               |

Poniżej znajduje się przykład subskrybowania prywatnych kanałów WebSocket po pomyślnym uwierzytelnieniu.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bybit *oBybit = new TsgcWSAPI_Bybit(NULL);
oBybit->Client = oClient;
oBybit->Bybit->ApiKey = "alsdjk23kandfnasbdfdkjhdsf";
oBybit->Bybit->ApiSecret = "aljdkfk3jkdaknfajndsifj23j";
oBybit->BybitClient = bybSpot;
oClient->Active = true;
void OnBybitAuthentication(TObject *Sender, bool aSuccess, const string aError, const string aRawMessage)
{
    if (aSuccess == true)
    {
        oClient->SubscribeOrderBook("BTCUSD");
        oClient->SubscribeTrade("BTCUSD");
    }
}
```

```
}  
}
```

## REST API

REST API zawiera listę publicznych i prywatnych metod do pobierania danych dotyczących rynków, prywatnego konta i portfela. Poniżej znajduje się lista dostępnych metod.

|                                      |           |
|--------------------------------------|-----------|
| <b>GetServerTime</b>                 | Publiczny |
| <b>GetKLine</b>                      | Publiczny |
| <b>GetMarkPriceKLine</b>             | Publiczny |
| <b>GetIndexPriceKLine</b>            | Publiczny |
| <b>GetPremiumIndexPriceKLine</b>     | Publiczny |
| <b>GetInstrumentsInfo</b>            | Publiczny |
| <b>GetOrderBook</b>                  | Publiczny |
| <b>GetTickers</b>                    | Publiczny |
| <b>GetFundingRateHistory</b>         | Publiczny |
| <b>GetPublicRecentTradingHistory</b> | Publiczny |
| <b>GetOpenInterest</b>               | Publiczny |
| <b>GetHistoricalVolatility</b>       | Publiczny |
| <b>GetInsurance</b>                  | Publiczny |
| <b>GetRiskLimit</b>                  | Publiczny |
| <b>GetDeliveryPrice</b>              | Publiczny |
| <b>GetLongShortRatio</b>             | Publiczny |
| <b>PlaceOrder</b>                    | Prywatny  |
| <b>PlaceMarketOrder</b>              | Prywatny  |
| <b>PlaceLimitOrder</b>               | Prywatny  |
| <b>AmendOrder</b>                    | Prywatny  |
| <b>CancelOrder</b>                   | Prywatny  |
| <b>GetOpenOrders</b>                 | Prywatny  |
| <b>CancelAllOrders</b>               | Prywatny  |
| <b>GetOrderHistory</b>               | Prywatny  |
| <b>GetPositionInfo</b>               | Prywatny  |
| <b>SetLeverage</b>                   | Prywatny  |
| <b>SwitchCrossIsolatedMargin</b>     | Prywatny  |
| <b>SetTPSLMode</b>                   | Prywatny  |
| <b>SwitchPositionMode</b>            | Prywatny  |
| <b>SetRiskLimit</b>                  | Prywatny  |
| <b>SetTradingStop</b>                | Prywatny  |
| <b>SetAutoAddMargin</b>              | Prywatny  |
| <b>AddOrReduceMargin</b>             | Prywatny  |
| <b>GetExecution</b>                  | Prywatny  |
| <b>GetClosedPNL</b>                  | Prywatny  |
| <b>ConfirmNewRiskLimit</b>           | Prywatny  |
| <b>GetWalletBalance</b>              | Prywatny  |
| <b>GetAccountInfo</b>                | Prywatny  |
| <b>GetTransactionLog</b>             | Prywatny  |



|                                |          |
|--------------------------------|----------|
| <b>BatchPlaceOrder</b>         | Prywatny |
| <b>BatchAmendOrder</b>         | Prywatny |
| <b>BatchCancelOrder</b>        | Prywatny |
| <b>SetDCP</b>                  | Prywatny |
| <b>GetFeeRate</b>              | Prywatny |
| <b>GetCollateralInfo</b>       | Prywatny |
| <b>SetMarginMode</b>           | Prywatny |
| <b>GetBorrowHistory</b>        | Prywatny |
| <b>GetCoinGreeks</b>           | Prywatny |
| <b>GetCoinInfo</b>             | Prywatny |
| <b>GetAllCoinsBalance</b>      | Prywatny |
| <b>CreateInternalTransfer</b>  | Prywatny |
| <b>GetInternalTransferList</b> | Prywatny |
| <b>GetDepositRecords</b>       | Prywatny |
| <b>GetDepositAddress</b>       | Prywatny |
| <b>CreateWithdrawal</b>        | Prywatny |
| <b>CancelWithdrawal</b>        | Prywatny |
| <b>GetWithdrawalRecords</b>    | Prywatny |

Poniżej znajduje się przykład pobierania otwartych zleceń.

```

TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bybit *oBybit = new TsgcWSAPI_Bybit(NULL);
oBybit->Client = oClient;
oBybit->Bybit->ApiKey = "alsdjk23kandfnasbdfdkjhdsf";
oBybit->Bybit->ApiSecret = "aldskjfk3jdkadknfajndsjsfj23j";
oBybit->BybitClient = bybSpot;
oBybit->REST_API->GetAccountInfo();

```

# API Forex

[Forex.com](#) (StoneX / CityIndex)

Forex.com to detaliczna platforma handlowa obsługiwana przez StoneX. Komponent opakowuje Trading REST API oraz usługę strumieniowania opartą na Lightstreamer, używaną przez Forex.com, StoneX i CityIndex (wszystkie trzy korzystają z tej samej platformy). Obsługiwane są następujące interfejsy API:

1. **Streaming API:** Lightstreamer TLCP 2.5 przez WebSocket. Ticki cenowe, aktualizacje zleceń, aktualizacje pozycji, marża konta i kwotowania są przesyłane w czasie rzeczywistym po pomyślnym zalogowaniu przez REST.
2. **REST API:** logowanie do sesji (UserName + Password + AppKey), zarządzanie zleceniami handlowymi i stop-limit, zapytania o konto i dane rynkowe. Używa HTTPS.

## Komponenty

- **TsgcWSAPI\_Forex:** główny klient. Przechowuje **Credentials**, udostępnia **REST\_API**, wewnętrzny klient strumieniowy **Lightstreamer** oraz zdarzenia **OnForex\***.
- **TsgcHTTP\_API\_Forex:** komponent REST dostępny za pośrednictwem właściwości **REST\_API** głównego klienta.
- **TsgcWSPClient\_Lightstreamer:** wewnętrzny klient protokołu TLCP 2.5 używany do strumieniowych subskrypcji (PRICES / ORDERS / QUOTES / CLIENTACCOUNTMARGIN / TRADEMARGIN).

## Opublikowane właściwości

- **Credentials:** grupa pól wymaganych do zalogowania się do REST API.
  - **UserName:** nazwa użytkownika konta Forex.com / StoneX / CityIndex.
  - **Password:** hasło do konta.
  - **AppKey:** klucz aplikacji wydany przez dostawcę.
  - **AppVersion:** opcjonalny ciąg wersji klienta.
  - **AppComments:** dowolny tekst rejestrowany podczas logowania.
  - **StreamingHost:** host Lightstreamer (domyślnie publiczny host strumieniowania).
- **AutoReconnect:** po włączeniu przekazuje ustawienie do wewnętrznego WatchDog klienta WebSocket, dzięki czemu sesja strumieniowania automatycznie nawiązuje połączenie ponownie po zerwaniu sieci (domyślnie true).
- **REST\_API:** odwołanie do komponentu **TsgcHTTP\_API\_Forex**.
- **Lightstreamer:** odwołanie do wewnętrznego klienta protokołu **TsgcWSPClient\_Lightstreamer** (AdapterManager jest ustawiony na STREAMINGALL).
- **SessionToken:** tylko do odczytu. Token sesji zwrócony przez wywołanie REST LogOn; używany jako *LS\_password* dla Lightstreamer.
- **ClientAccountId / TradingAccountId:** tylko do odczytu. Wypełniane po pomyślnym zalogowaniu (LogOn).

## Opublikowane zdarzenia

### OnForexConnect

Wywoływane po utworzeniu sesji strumieniowania i możliwości rozpoczęcia subskrypcji.

### OnForexDisconnect

Wywoływane po zakończeniu sesji strumieniowej (zamknięcie przez serwer, utrata sieci lub jawne rozłączenie).

### OnForexException

Wywoływane przy błędach REST (w tym błędzie LogOn) oraz przy awariach sesji strumieniowania.

### OnForexPriceTick

Wywoływane przy każdej aktualizacji PRICES: Bid, Offer, Price, Change, Direction, AuditId, TickDate.

### OnForexOrderUpdate

Wywoływane dla każdej aktualizacji ORDERS (nowe, zrealizowane, anulowane, odrzucone).

### OnForexPositionUpdate

Wywoływane dla każdej aktualizacji otwartej pozycji na aktywnym koncie handlowym (TradingAccount).

### OnForexAccountMargin

Wywoływane dla każdej aktualizacji CLIENTACCOUNTMARGIN / TRADEMARGIN (kapitał własny, depozyt zabezpieczający, wolny depozyt, waluta).

### OnForexQuote

Wywoływane przy każdej aktualizacji QUOTES (przepływ pracy request-for-quote).

### OnForexSubscribe

Wywoływane za każdym razem, gdy serwer potwierdzi subskrypcję Lightstreamer (SUB-OK / SUBCMD). Parametry: *aAdapter* (PRICES, ORDERS, QUOTES, CLIENTACCOUNTMARGIN lub TRADEMARGIN), *aGroup* (grupa elementów, np. *ID.<ClientAccountId>*), *aItems*, *aFields*.

### OnForexUnsubscribe

Uruchamiane po zakończeniu subskrypcji — za pośrednictwem **UnwatchMarket**, przy rozłączeniu strumienia lub gdy serwer wyśle ramkę UNSUB. Parametry: *aAdapter*, *aGroup*.

## Najczęstsze zastosowania

- **REST API**
  - [Jak połączyć się z REST API](#)
  - [Jak składać zlecenia handlowe i stop-limit](#)
  - [Jak odpytywać konta, pozycje i historię zleceń](#)
  - [Jak pobierać dane rynkowe](#)
- **WebSockets API**
  - [Jak połączyć się z WebSocket API](#)
  - [Jak subskrybować adaptery strumieniowe](#)

## Przykład

Utwórz klienta, ustaw poświadczenia, połącz się, subskrybuj strumień cen rynkowych i rozłącz się.

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->OnForexPriceTick = ForexPriceTick;
oForex->Connect();
oForex->WatchMarket(401484830);
oForex->Disconnect();
```

# Forex | Connect REST API

Każde wywołanie do Forex.com / StoneX / CityIndex Trading API wymaga aktywnej sesji. Stan sesji jest uzyskiwany za pomocą **LogOn**, kończony przez **LogOff** i utrzymywany za pomocą **Ping**.

## Poświadczenia

Ustaw dane uwierzytelniające w głównym komponencie klienta. Są one kopiowane do sesji REST przy pierwszym wywołaniu.

- **UserName:** nazwa użytkownika konta Forex.com / StoneX / CityIndex.
- **Hasło:** hasło konta.
- **AppKey:** klucz aplikacji wystawiony przez giełdę.
- **AppVersion / AppComments:** opcjonalne ciągi rejestrowane podczas logowania.

## Metody sesji

|                  |                            |                                                                                                                                                                                                                                      |
|------------------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LogOn            | UserName, Password, AppKey | Otwiera sesję REST. Zwraca pełną odpowiedź JSON i przechowuje SessionToken, ClientAccountId oraz TradingAccountId w głównym kliencie. Wszystkie kolejne wywołania REST i strumieniowania są uwierzytelniane przy użyciu tego tokenu. |
| LogOff           |                            | Unieważnia sesję na serwerze i czyści token.                                                                                                                                                                                         |
| Ping             |                            | Utrzymuje sesję aktywną. Sesja serwera wygasa po okresie nieaktywności; należy okresowo wysyłać Ping, gdy nie jest wykonywane inne wywołanie REST.                                                                                   |
| GetServiceStatus |                            | Raportuje bieżący status usługi (bez wymaganego uwierzytelniania).                                                                                                                                                                   |

## Przepływ tokenu

Token zwrócony przez **LogOn** jest dołączany do każdego kolejnego żądania REST jako nagłówek HTTP *UserName*. Ten sam token jest używany ponownie przez klienta strumieniowego jako pole *LS\_password* podczas tworzenia sesji Lightstreamer. Nie ma potrzeby dwukrotnego wywoływania LogOn: wywołanie **Connect** na głównym kliencie najpierw wykonuje REST LogOn, a dopiero potem otwiera gniazdo strumieniowe.

## Przykład

### Otwórz sesję REST i wywołaj Ping

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
ShowMessage(oForex->REST_API->LogOn(oForex->Credentials->UserName,
                                     oForex->Credentials->Password,
                                     oForex->Credentials->AppKey));

ShowMessage(oForex->REST_API->Ping());
oForex->REST_API->LogOff();
```

# Forex | Zlecenia handlowe i Stop-Limit

REST API udostępnia pięć punktów końcowych do zarządzania zleceniami. Każdy punkt końcowy jest dostępny w trzech wariantach przeciążonych, dzięki czemu wywołujący może wybrać wymagany poziom kontroli.

## Punkty końcowe

|                      |                                                                                       |
|----------------------|---------------------------------------------------------------------------------------|
| NewTradeOrder        | Składa zlecenie rynkowe / po najlepszej cenie na rynku CFD.                           |
| UpdateTradeOrder     | Dołącza lub zmienia dołączoną ochronę stop/limit dla istniejącej otwartej transakcji. |
| NewStopLimitOrder    | Składa oczekujące zlecenie stop lub limit (jeszcze nieobecne na rynku).               |
| UpdateStopLimitOrder | Modyfikuje istniejące oczekujące zlecenie stop lub limit.                             |
| CancelOrder          | Anuluje oczekujące zlecenie stop lub limit.                                           |

## Poziomy przeciążenia

Metody zleceń są dostępne w dwóch przeciążeniach:

1. **Przeciążenie ze skalarowymi parametrami** (tylko NewTradeOrder i CancelOrder) — przyjmuje najczęściej używane pola (MarketId, TradingAccountId, Direction, Quantity, BidPrice / OfferPrice, AuditId dla NewTradeOrder; OrderId, TradingAccountId, MarketId dla CancelOrder). Metoda pomocnicza buduje żądanie wewnętrznie. Przydatna dla prostych zleceń bez ochrony.
2. **Typowane przeciążenie obiektu zlecenia** — przyjmuje **TsgcHTTPForexTradeOrder**, **TsgcHTTPForexStopLimitOrder** lub **TsgcHTTPForexCancelOrder**. Obiekty te udostępniają wszystkie udokumentowane pola, w tym **IfDone** (dołączone nawiasy stop/limit), tablice **Close** (semantyka zamknięcia według identyfikatora zlecenia), **PositionMethodId**, **Reference** oraz flagi częściowego wypełnienia. Jest to zalecana ścieżka dla wszystkiego poza prostym zleceniem rynkowym; konwencja budowania obiektów jest wspólna z innymi giełdami, takimi jak Kucoin.

## Pola TsgcHTTPForexTradeOrder

- **MarketId**: numeryczny identyfikator rynku zwrócony przez ListCfdMarkets lub FullSearchWithTags.
- **TradingAccountId**: aktywne konto handlowe. Należy je odczytać z głównego klienta po wywołaniu LogOn.
- **Direction**: *buy* lub *sell*.
- **Quantity**: rozmiar transakcji w jednostce natywnej rynku.
- **BidPrice / OfferPrice**: ceny zaobserwowane przez klienta w momencie składania zlecenia (Forex.com używa ich do wykrywania nieaktualnych kwotowań).
- **AuditId**: identyfikator audytu ceny z ostatniego impulsu cenowego.
- **PositionMethodId**: tryb nettingu/hedgingu (*1 = Netting*, *2 = Hedging*).
- **IfDone**: lista dołączonych zleceń (stop, limit, gwarantowany stop, trailing stop), które aktywują się po realizacji zlecenia nadrzędnego.
- **Close**: tablica identyfikatorów zleceń do zamknięcia po realizacji tego zlecenia (częściowe zamknięcie).
- **Reference**: referencja klienta podana przez wywołującego, powtarzana w aktualizacjach zleceń.

## Przykład

### Kup 1000 EUR/USD po cenie rynkowej z dołączonym zleceniem take-profit i stop-loss

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->Connect();

TsgcHTTPForexTradeOrder *oOrder = new TsgcHTTPForexTradeOrder();
oOrder->MarketId = 401484830;
```

```
oOrder->TradingAccountId = oForex->TradingAccountId;
oOrder->Direction          = fdBuy;
oOrder->Quantity            = 1000;
oOrder->BidPrice             = 1.09120;
oOrder->OfferPrice           = 1.09130;
ShowMessage(oForex->NewTradeOrder(oOrder));
delete oOrder;
```

# Forex | Zapytania o konto

Te punkty końcowe odczytują stan konta i historię zleceń. Wymagają aktywnej sesji (patrz [Connect REST API](#)).

## Punkty końcowe

|                            |                                 |                                                                                                                                                                                                    |
|----------------------------|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetClientAndTradingAccount |                                 | Zwraca konto klienta oraz listę kont handlowych powiązanych z uwierzytelnionym użytkownikiem.                                                                                                      |
| ListOpenPositions          | TradingAccountId                | Zwraca wszystkie aktualnie otwarte pozycje na rachunku handlowym.                                                                                                                                  |
| ListActiveStopLimitOrders  | TradingAccountId                | Zwraca wszystkie oczekujące zlecenia stop i limit.                                                                                                                                                 |
| GetOrder                   | OrderId                         | Zwraca pojedyncze zlecenie według identyfikatora, wraz z historią realizacji i dołączonymi zleceniami IfDone.                                                                                      |
| ListTradeHistory           | TradingAccountId, MaxResults    | Zwraca historię zleceń handlowych dla konta, posortowaną od najnowszych.                                                                                                                           |
| ListStopLimitOrderHistory  | TradingAccountId, MaxResults    | Zwraca historię zleceń stop-limit dla konta, zaczynając od najnowszych.                                                                                                                            |
| SimulateTrade              | JSON / skalar / obiekt zlecenia | Uruchamia zlecenie handlowe przez walidator serwera bez jego składania. Przydatne do wstępnego sprawdzenia marginu i tolerancji ceny. Przyjmuje te same trzy warianty przeciążeń co NewTradeOrder. |

## Wycofane punkty końcowe

Oficjalna dokumentacja Forex.com / StoneX / CityIndex nadal opisuje kilka punktów końcowych, których aktywny serwer już nie obsługuje. Celowo nie są one udostępniane przez komponent:

- **GetSystemLookup** — dokumentacja opisuje ogólny punkt końcowy wyszukiwania, jednak serwer odpowiada kodem HTTP 404 dla każdego ze znanych kluczy wyszukiwania. Zamiast tego należy używać **GetClientAndTradingAccount** i **ListCfdMarkets**.
- **/news/headlines/\*** — rodzina kanałów informacyjnych jest nadal wymieniona w dokumentacji, jednak na serwerze produkcyjnym zwraca HTTP 500. Brak dostępnego zamiennika w Trading API; wiadomości są dostępne wyłącznie przez interfejs webowy.

## Przykład

### Lista otwartych pozycji oraz ostatnich 50 transakcji

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->Connect();
ShowMessage(oForex->REST_API->ListOpenPositions(oForex->TradingAccountId));
ShowMessage(oForex->REST_API->ListTradeHistory(oForex->TradingAccountId, 50));
```

# Forex | Dane rynkowe

Statyczne metadane rynku i historyczne ceny. Te punkty końcowe służą do odnajdywania wartości **MarketId**, które następnie są przekazywane do punktów końcowych transakcji i adapterów strumieniowania.

## Punkty końcowe

|                      |                                                 |                                                                                                                        |
|----------------------|-------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| GetMarketInformation | MarketId                                        | Zwraca pełny deskryptor rynku (symbol, miejsca dziesiętne, rozmiar pipsa, współczynnik marży, godziny handlu).         |
| ListCfdMarkets       | SearchByMarketName, ClientAccountId, MaxResults | Wyświetla rynki CFD pasujące do podanego prefiksu nazwy.                                                               |
| FullSearchWithTags   | Query, Tags, MaxResults                         | Swobodne wyszukiwanie tekstowe na wszystkich rynkach z opcjonalnymi filtrami tagów (FX, Indices, Shares, Commodities). |
| GetPriceBars         | MarketId, Interval, Span, PriceBars             | Historyczne słupki OHLC. Interval to jeden z MINUTE, HOUR, DAY, WEEK; Span to liczba interwałów na słupek.             |
| GetPriceTicks        | MarketId, PriceTicks                            | Najnowsza migawka ticków dla rynku (bid / ask / mid).                                                                  |

## Przykład

### Wyszukaj parę EUR/USD i odczytaj ostatnie 100 słupków jednominutowych

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->Connect();
ShowMessage(oForex->REST_API->ListCfdMarkets("EUR/USD", 0, 10));
ShowMessage(oForex->REST_API->GetPriceBars(401484830, "MINUTE", 1, 100));
```



# Forex | Połącz WebSocket API

Forex.com / StoneX / CityIndex przesyła strumieniowo dane rynkowe i aktualizacje kont przez Lightstreamer. Komponent obsługuje natywny protokół TLCP 2.5 Lightstreamer przez WebSocket; żadna natywna biblioteka Lightstreamer nie jest wymagana.

## Transport

- **Subprotokół WebSocket:** *TLCP-2.5.0.lightstreamer.com*. Anonsowany w nagłówku żądania *Sec-WebSocket-Protocol*.
- **Zestaw adapterów:** *STREAMINGALL*. Skonfigurowany na wewnętrznym kliencie subprotokołu **Lightstreamer**; wszystkie adaptory danych Forex.com (PRICES, ORDERS, QUOTES, CLIENTACCOUNTMARGIN, TRADEMARGIN) działają w ramach tego zestawu adapterów.
- **Authentication:** token sesji uzyskany z wywołania REST **LogOn** jest wysyłany jako *LS\_password* podczas tworzenia sesji Lightstreamer, wraz z *LS\_user* ustawionym na nazwę użytkownika konta.

## Ponowne połączenie

Gdy włączony jest **AutoReconnect** (domyślnie), wbudowany WatchDog klienta WebSocket ponownie otwiera gniazdo po błędach sieciowych. Sam protokół Lightstreamer sygnalizuje ramki *LOOP*, gdy serwer chce, aby klient przeszedł do innego adresu URL lub ponownie powiązał sesję; komponent obsługuje je transparentnie i wywołuje **OnForexDisconnect** / **OnForexConnect** tylko przy pełnych przerwach sesji.

Gdy sesja REST wygasa (brak sygnału Ping przez długi czas), serwer strumieniowy zamyka sesję. Komponent wywołuje ponownie **LogOn** przy użyciu przechowywanych poświadczeń, odświeża **SessionToken** i restartuje sesję Lightstreamer z nowym tokenem.

## Przykład

### Otwórz sesję przesyłania strumieniowego

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->OnForexConnect = ForexConnected;
oForex->OnForexDisconnect = ForexDisconnected;
oForex->OnForexException = ForexException;
oForex->AutoReconnect = true;
oForex->Connect();
oForex->Disconnect();
```

# Forex | Subskrypcja adapterów strumieniowych

Dane strumieniowe są dostarczane przez pięć adapterów danych Lightstreamer w zestawie adapterów *STREAMINGALL*. Każdy adapter ma własną konwencję nazewnictwa elementów i tryb subskrypcji — **MERGE** dla stanu w postaci migawki (bieżąca cena, bieżący depozyt zabezpieczający konta) i **RAW** dla kanałów strumieniowania zdarzeń (aktualizacje zleceń, aktualizacje pozycji, zdarzenia kwotowania).

## Adaptory

|                      |                               |       |                       |                                                                                                                                                                       |
|----------------------|-------------------------------|-------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CENY                 | ID.{MarketId}                 | MERGE | OnForexPriceTick      | MarketId, TickDate, Bid, Offer, Price, High, Low, Change, Direction, AuctionId, StatusSummary                                                                         |
| ORDERS               | ORDERS.{ClientAccountId}      | RAW   | OnForexOrderUpdate    | OrderId, MarketId, TradingAccountId, Direction, Quantity, Price, Status, OrderType, Currency, LastChangedDateTimeUTC                                                  |
| QUOTES               | QUOTES.{ClientAccountId}      | RAW   | OnForexQuote          | QuoteId, OrderId, RequestDateTimeUTC, ApprovalDateTimeUTC, BidPrice, OfferPrice, Status                                                                               |
| CLIENTACCOUNT-MARGIN | ID.{ClientAccountId}          | MERGE | OnForexAccountMargin  | ClientAccountId, Cash, Margin, MarginIndicator, NetEquity, OpenTradeEquity, TotalMarginRequirement, TradeableFunds, CurrencyId, CurrencyISOCode                       |
| TRADEMARGIN          | TRADEMARGIN.{ClientAccountId} | RAW   | OnForexPositionUpdate | OrderId, ClientAccountId, MarketId, MarketTypeId, DirectionId, Direction, Quantity, OpenPrice, CurrentPrice, Multiplier, MarginRequirement, Currency, OpenTradeEquity |

### Uwagi:

- PRICES i CLIENTACCOUNTMARGIN używają ujednoliconego formatu elementu *ID.{numericId}* i przesyłają strumieniowo pojedynczy zmienny rekord na element (MERGE).
- ORDERS, QUOTES i TRADEMARGIN używają starszego prefiksu per-adapter (*ORDERS.{id}*, *QUOTES.{id}*, *TRADEMARGIN.{id}*) i strumieniają wszystkie zdarzenia (RAW). Jeden element zawiera wszystkie zlecenia / kwotowania / transakcje dla zalogowanego **ClientAccountId**; rekordy identyfikują się przez **OrderId** / **QuoteId**.
- Rekord TRADEMARGIN odpowiada oficjalnemu *TradeMarginDTO* z dziedzicznego SDK CityIndex. Pole **DirectionId** przyjmuje wartość numeryczną (1 = kupno, 2 = sprzedaż). Biblioteka udostępnia również pole **Direction** jako ciąg znaków "buy" / "sell". Pole **OpenPrice** to cena wejścia w transakcję (pole serwera *PriceTakenAt*), **CurrentPrice** to ostatnia cena wyceny (*PriceCalculatedAt*), a **OpenTradeEquity** to zmienny zysk/strata w walucie rachunku (*OTECoinverted*).
- Pole **Direction** w ORDERS zawiera kod numeryczny publikowany przez serwer (1 / 2); **Status** jest ciągiem znaków (np. "Accepted", "Open", "Cancelled"); **OrderType** jest ciągiem znaków ("Market" / "Stop" / "Limit").
- Kanał TRADEMARGIN nie publikuje per-transakcję pól **TradingAccountId**, **MarketName** ani znacznika czasu per-transakcję; jeden element zawiera wszystkie pozycje dla zalogowanego **ClientAccountId**. Należy wywołać **REST\_API.ListOpenPositions**, aby uzyskać te pola. Kanał ORDERS nie publikuje **TriggerPrice** ani **Applicability**; gdy są wymagane, należy wywołać **REST\_API.GetOrder**.

## Metody pomocnicze

Główny klient obudowuje mechanizm SUB/UNSUB Lightstreamer. Najczęściej używane to:

- **WatchMarket(aMarketId):** subskrybuje cenniki (PRICES) dla jednego rynku. Aktualizacje są dostarczane przez zdarzenie **OnForexPriceTick**.
- **UnwatchMarket(aMarketId):** usuwa subskrypcję.
- **WatchAccount:** subskrybuje ORDERS, CLIENTACCOUNTMARGIN i TRADEMARGIN dla zalogowanych kont. Aktualizacje przychodzą w odpowiadających zdarzeniach.
- **WatchQuotes:** subskrybuje adapter QUOTES dla przepływów pracy żądania wyceny.

## Zdarzenia cyklu życia subskrypcji

Główny klient wywołuje dwa zdarzenia cyklu życia odpowiadające ramkom TLCP SUBOK / UNSUB:

- **OnForexSubscribe(Sender, aAdapter, aGroup, aItems, aFields):** wywoływane, gdy serwer potwierdza dowolną subskrypcję Lightstreamer. *aAdapter* to nazwa adaptera danych (PRICES, ORDERS, QUOTES, CLIENTACCOUNTMARGIN, TRADEMARGIN); *aGroup* to grupa elementów (np. *ID.<ClientAccountId>* lub *ORDERS.<ClientAccountId>*); *aItems* i *aFields* to liczby zgłoszone przez serwer w ramce SUBOK / SUBCMD.
- **OnForexUnsubscribe(Sender, aAdapter, aGroup):** wywoływane po zakończeniu subskrypcji za pomocą **UnwatchMarket**, po rozłączeniu strumieniowania lub gdy serwer wyśle ramkę UNSUB.

## Przykład

### Subskrybuj tiki cenowe EUR/USD

```
TsgcWSAPI_Forex *oForex = new TsgcWSAPI_Forex(NULL);
oForex->Credentials->UserName = "<username>";
oForex->Credentials->Password = "<password>";
oForex->Credentials->AppKey = "<appkey>";
oForex->OnForexPriceTick = ForexPriceTick;
oForex->Connect();
oForex->WatchMarket(401484830);
```

# API Blockchain

## Blockchain

Blockchain WebSocket API umożliwia deweloperom otrzymywanie powiadomień w czasie rzeczywistym o nowych transakcjach i blokach.

Po otwarciu WebSocket można zasubskrybować kanał:

- **SubscribeTransactions:** Subskrybuj powiadomienia o wszystkich nowych transakcjach bitcoin.
- **UnsubscribeTransactions:** Anulowanie subskrypcji powiadomień o wszystkich nowych transakcjach bitcoin.
- **SubscribeAddress:** Odbieranie nowych transakcji dla określonego adresu bitcoin.
- **UnSubscribeAddress:** Zatrzymuje odbieranie nowych transakcji dla określonego adresu bitcoin.

Transakcje są odbierane przez zdarzenie **OnNewTransaction**:

```
{
  "op": "utx",
  "x": {
    "lock_time": 0,
    "ver": 1,
    "size": 192,
    "inputs": [
      {
        "sequence": 4294967295,
        "prev_out": {
          "spent": true,
          "tx_index": 99005468,
          "type": 0,
          "addr": "1BwGf3z7n2fHk6NoVJNkv32qwyAYsMhkWf",
          "value": 65574000,
          "n": 0,
          "script": "76a91477f4c9ee75e449a74c21a4decfb50519cbc245b388ac"
        },
        "script": "483045022100e4ff962c292705f051c2c2fc519fa775a4d8955bce1a3e29884b2785277999ed02200b537ebd22a9f25f"
      }
    ],
    "time": 1440086763,
    "tx_index": 99006637,
    "vin_sz": 1,
    "hash": "0857b9de1884eec314ecf67c040a2657b8e083e1f95e31d0b5ba3d328841fc7f",
    "vout_sz": 1,
    "relayed_by": "127.0.0.1",
  }
}
```

```

    "out": [
      {
        "spent": false,
        "tx_index": 99006637,
        "type": 0,
        "addr": "1A828tTnkVFJfSvLCqF42ohZ51ksS3jJgX",
        "value": 65564000,
        "n": 0,
        "script": "76a914640cfd7b79d94d1c980133e3587bd6053f091f388ac"
      }
    ]
  }
}

```

- **SubscribeBlocks:** Otrzymuj powiadomienia po znalezieniu nowego bloku. Uwaga: jeśli łańcuch się rozgałęzia, dla danej wysokości bloku zostanie odebrane więcej niż jedno powiadomienie.
- **UnSubscribeBlocks:** Zatrzymaj otrzymywanie powiadomień po znalezieniu nowego bloku. Uwaga: jeśli łańcuch się rozwidli, otrzymasz więcej niż jedno powiadomienie dla określonej wysokości bloku.

Bloki są odbierane przez zdarzenie **OnNewBlock**:

```

{
  "op": "block",
  "x": {
    "txIndexes": [
      3187871,
      3187868
    ],
    "nTx": 0,
    "totalBTCSent": 0,
    "estimatedBTCSent": 0,
    "reward": 0,
    "size": 0,
    "blockIndex": 190460,
    "prevBlockIndex": 190457,
    "height": 170359,
    "hash": "00000000000006436073c07dfa188a8fa54fefadf571fd774863cda1b884b90f",
    "mrklRoot": "94e51495e0e8a0c3b78dac1220b2f35ceda8799b0a20cfa68601ed28126cfcc2",
    "version": 1,
    "time": 1331301261,
    "bits": 436942092,
    "nonce": 758889471
  }
}

```

# API Cex

## Cex

WebSocket API umożliwia otrzymywanie powiadomień w czasie rzeczywistym bez konieczności wysyłania dodatkowych żądań, co sprawia, że jest to szybszy sposób na uzyskiwanie danych z giełdy

Komponent Cex posiada właściwość `Cex`, w której można podać klucze API dostarczone przez Cex w celu uzyskania dostępu do danych konta.

## Kodowanie wiadomości

- Wszystkie wiadomości są zakodowane w formacie JSON.
- Ceny są prezentowane jako ciągi znaków, aby uniknąć błędów zaokrąglania podczas analizowania JSON po stronie klienta.
- Kompresja ramek WebSocket nie jest obsługiwana przez serwer.
- Czas jest przedstawiony jako całkowity znacznik czasu UNIX w sekundach.

## Uwierzytelnianie

Aby uzyskać dostęp do danych WebSocket CEX.IO, wymagana jest autoryzacja.

- Zaloguj się na konto CEX.IO.
- Przejdź na stronę <https://cex.io/trade/profile#/api>.
- Wybierz typ wymaganych uprawnień.
- Należy kliknąć przycisk „Generuj klucz” i zapisać klucz tajny, ponieważ będzie on niedostępny po aktywacji.
- Aktywuj swój klucz.

## Łączność

- Jeśli podłączone gniazdo jest nieaktywne przez 15 sekund, serwer CEX.IO wyśle wiadomość PING.
- Tylko serwer może być inicjatorem żądania PING.
- Serwer wysyła ping tylko do uwierzytelnionego użytkownika.
- Użytkownik musi odpowiedzieć wiadomością PONG. W przeciwnym razie połączenie WebSocket zostanie ROZŁĄCZONE. Jest to obsługiwane automatycznie przez bibliotekę.
- W przypadku uwierzytelnionego użytkownika, jeśli przez 15 sekund nie nadejdzie żadne powiadomienie ani sygnał ping z serwera, bezpieczniej jest wysłać żądanie w rodzaju „ticker” lub „get-balance” i odebrać odpowiedź, aby potwierdzić łączność i uwierzytelnienie.

## Kanały publiczne

Te kanały nie wymagają uwierzytelniania. Odpowiedzi z serwera są odbierane za pośrednictwem zdarzenia OnCexMessage.

- **SubscribeTickers:** Strumień tickerów zawierający tylko ceny transakcji dla wszystkich par (przestarzałe)

```
{
  "e": "tick",
  "data": {
    "symbol1": "BTC",
    "symbol2": "USD",
    "price": "428.0123"
  }
}
```

- **SubscribeChart:** Kanały danych wykresu OHLCV z wartościami Open, High, Low, Close, Volume (przestarzałe)

```
{
  'e': 'ohlcv24',
  'pair': 'BTC:USD',
  'data': [
    '418.2936',
    '420.277',
    '412.09',
    '416.9778',
    '201451078368'
  ]
}
```

- **Subskrybuj Parę:** Kanał głębokości rynku (przestarzały)

```
{
  'e': 'md_grouped',
  'data': {
    'pair': 'BTC:USD',
    'id': 11296131,
    'sell': {
      '427.5000': 1000000,
      '480.0000': 263544334,
      ...
    },
    'buy': {
      '385.0000': 3630000,
      '390.0000': 1452458642,
      ... 400+ pairs together with 'sell' pairs
    }
  }
}
```

```
}  
}
```

- **Subskrybuj parę:** Kanał książki zleceń (przestarzały)

```
{  
  'e': 'md',  
  'data': {  
    'pair': 'BTC:USD',  
    'buy_total': 63221099,  
    'sell_total': 112430315118,  
    'id': 11296131,  
    'sell': [  
      [426.45, 10000000],  
      [426.5, 66088429300],  
      [427, 10000000],  
      ... 50 pairs overaall  
    ],  
    'buy': [  
      [423.3, 4130702],  
      [423.2701, 10641168],  
      [423.2671, 10000000],  
      ... 50 pairs overaall  
    ]  
  }  
}
```

## Kanały prywatne

Aby uzyskać dostęp do tych kanałów, należy najpierw wywołać metodę `Authenticate`. Odpowiedzi z serwera są odbierane przez zdarzenie `OnCexMessage`.

## GetTicker

```
{  
  "e": "ticker",  
  "data": {  
    "timestamp": "1471427037",  
    "low": "290",  
    "high": "290",  
    "last": "290",  
    "volume": "0.02062068",  
    "volume30d": "14.38062068",  
    "bid": 240,  
    "ask": 290,  
    "pair": [  
      "BTC",  
      "USD"  
    ]  
  },  
  "oid": "1471427036908_1_ticker",  
}
```



```
    "ok": "ok"
  }
```

## GetBalance

```
{
  "e": "get-balance",
  "data": {
    "balance": {
      'LTC': '10.00000000',
      'USD': '1024.00',
      'RUB': '35087.98',
      'EUR': '217.53',
      'GHS': '10.00000000',
      'BTC': '9.00000000'
    },
    "obalance": {
      'BTC': '0.12000000',
      'USD': "512.00",
    },
  },
  "time": 1435927928597
  "oid": "1435927928274_2_get-balance",
  "ok": "ok"
}
```

## SubscribeOrderBook

```
{
  "e": "order-book-subscribe",
  "data": {
    "timestamp": 1435927929,
    "bids": [
      [
        241.947,
        155.91626
      ],
      [
        241,
        981.1255
      ],
    ],
    "asks": [
      [
        241.95,
        15.4613
      ],
      [
        241.99,
        17.3303
      ],
    ],
  },
  "pair": "BTC:USD",
  "id": 67809
},
```

```
"oid": "1435927928274_5_order-book-subscribe",
"ok": "ok"
}
```

## UnSubscribeOrderBook

```
{
  "e": "order-book-unsubscribe",
  "data": {
    "pair": "BTC:USD"
  },
  "oid": "1435927928274_4_order-book-unsubscribe",
  "ok": "ok"
}
```

## GetOpenOrders

```
{
  "e": "open-orders",
  "data": [
    {
      "id": "2477098",
      "time": "1435927928618",
      "type": "buy",
      "price": "241.9477",
      "amount": "0.02000000",
      "pending": "0.02000000"
    },
    {
      "id": "2477101",
      "time": "1435927928634",
      "type": "sell",
      "price": "241.9493",
      "amount": "0.02000000",
      "pending": "0.02000000"
    }
  ],
  "oid": "1435927928274_9_open-orders",
  "ok": "ok"
}
```

## PlaceOrder

```
{
  "e": "place-order",
  "data": {
    "complete": false,
    "id": "2477098",
    "time": 1435927928618,
    "pending": "0.02000000",
    "amount": "0.02000000",
    "type": "buy",
    "price": "241.9477"
  },
  "oid": "1435927928274_7_place-order",
  "ok": "ok"
}
```

## CancelReplaceOrder

```
{
  "e": "cancel-replace-order",
  "data": {
    "complete": false,
    "id": "2689009",
    "time": 1443464955904,
    "pending": "0.04000000",
    "amount": "0.04000000",
    "type": "buy",
    "price": "243.25"
  },
  "oid": "1443464955209_16_cancel-replace-order",
  "ok": "ok"
}
```

## GetOrderRequest

W systemie CEX.IO zlecenia mogą znajdować się w silniku handlowym lub w archiwalnej bazie danych. Mogą występować okresy (~2 sekundy lub dłużej), gdy zlecenie jest zrealizowane/anulowane, ale jeszcze nie przeniesione do archiwalnej bazy danych. Oznacza to, że nie można go zobaczyć przy użyciu wywołań: `archived-orders/open-orders`. To wywołanie umożliwia uzyskanie informacji o zleceniu w każdym przypadku. Odpowiedzi mogą mieć różny format w zależności od lokalizacji zleceń.

```
{
  "e": "get-order",
  "data": {
    "user": "xxx",
    "type": "buy",
    "symbol1": "BTC",
    "symbol2": "USD",
    "amount": "0.02000000",
    "remains": "0.02000000",
  }
}
```

```

    "price": "50.75",
    "time": 1450214742160,
    "tradingFeeStrategy": "fixedFee",
    "tradingFeeBuy": "5",
    "tradingFeeSell": "5",
    "tradingFeeUserVolumeAmount": "nil",
    "a:USD:c": "1.08",
    "a:USD:s": "1.08",
    "a:USD:d": "0.00",
    "status": "a",
    "orderId": "5582060"
  },
  "oid": "1450214742135_10_get-order",
  "ok": "ok"
}

```

## CancelOrderRequest

```

{
  "e": "cancel-order",
  "data": {
    "order_id": "2477098"
    "time": 1443468122895
  },
  "oid": "1435927928274_12_cancel-order",
  "ok": "ok"
}

```

## GetArchivedOrders

```

{
  "e": "archived-orders",
  "data": [
    {
      "type": "buy",
      "symbol1": "BTC",
      "symbol2": "USD",
      "amount": 0,
      "amount2": 5000,
      "remains": 0,
      "time": "2015-04-17T10:46:27.971Z",
      "tradingFeeBuy": "2",
      "tradingFeeSell": "2",
      "ta:USD": "49.00",
      "fa:USD": "0.98",
      "orderId": "2340298",
      "status": "d",
      "a:BTC:cds": "0.18151851",
      "a:USD:cds": "50.00",
      "f:USD:cds": "0.98"
    },
    {
      "type": "buy",
      "symbol1": "BTC",
      "symbol2": "USD",

```

```

        "amount": 0,
        "amount2": 10000,
        "remains": 0,
        "time": "2015-04-08T15:46:04.651Z",
        "tradingFeeBuy": "2.99",
        "tradingFeeSell": "2.99",
        "ta:USD": "97.08",
        "fa:USD": "2.91",
        "orderId": "2265315",
        "status": "d",
        "a:BTC:cds": "0.39869578",
        "a:USD:cds": "100.00",
        "f:USD:cds": "2.91"
    }
],
"oid": "1435927928274    15_archived-orders",
"ok": "ok"
}

```

## OpenPosition

```

{
    "e": "open-position",
    "oid": "1435927928274_7_open-position",
    "data": {
        'amount': '1',
        'symbol': 'BTC',
        "pair": [
            "BTC",
            "USD"
        ],
        'leverage': '2',
        'ptype': 'long',
        'anySlippage': 'true',
        'eoprice': '650.3232',
        'stopLossPrice': '600.3232'
    }
}

```

## GetPosition

```

{
    "e": "get_position",
    "ok": "ok",
    "data": {
        "user": "ud100036721",
        "pair": "BTC:USD",
        "amount": "1.00000000",
        "symbol": "BTC",
        "msymbol": "USD",
        "omamount": "1528.77",
        "lsymbol": "USD",
        "lamount": "3057.53",
        "slamount": "3380.11",
        "leverage": "3",

```

```

    "stopLossPrice": "3380.1031",
    "dfl": "3380.10310000",
    "flPrice": "3057.53333333",
    "otime": 1513002370342,
    "psymbol": "BTC",
    "ptype": "long",
    "ofee": "10",
    "pfee": "10",
    "cfee": "10",
    "tfeeAmount": "152.88",
    "rinterval": "14400000",
    "okind": "Manual",
    "a:BTC:c": "1.00000000",
    "a:BTC:s": "1.00000000",
    "oorder": "89101551",
    "pamount": "1.00000000",
    "lremains": "3057.53",
    "slremains": "3380.11",
    "oprice": "4586.3000",
    "status": "a",
    "id": "125531",
    "a:USD:cds": "4739.18"
  }
}

```

## GetOpenPositions

```

{
  'e': 'open_positions',
  "oid": "1435927928256_7_open-positions",
  'ok': 'ok',
  'data': [
    {
      'user': 'ud100036721',
      'id': '104102',
      'otime': 1475602208467,
      'symbol': 'BTC',
      'amount': '1.00000000',
      'leverage': '2',
      'ptype': 'long',
      'psymbol': 'BTC',
      'msymbol': 'USD',
      'lsymbol': 'USD',
      'pair': 'BTC:USD',
      'oprice': '607.5000',
      'stopLossPrice': '520.3232',
      'ofee': '1',
      'pfee': '3',
      'cfee': '4',
      'tfeeAmount': '3.04',
      'pamount': '1.00000000',
      'omamount': '303.75',
      'lamount': '303.75',
      'oorder': '34106774',
      'rinterval': '14400000',
      'dfl': '520.32320000',
      'slamount': '520.33',
      'slremains': '520.33',
      'lremains': '303.75',
      'flPrice': '303.75000000',
      'a:BTC:c': '1.00000000',

```

```
        'a:BTC:s': '1.00000000',
        'a:USD:cds': '610.54',
    },
    ...
]
}
```

## ClosePosition

```
{
  'e': 'close_position',
  'oid': "1435927928364_7_close-position",
  'ok': 'ok',
  'data': {
    'id': 104034,
    'ctime': 1475484981063,
    'ptype': 'long',
    'msymbol': 'USD'
    'pair': {
      'symbol1': 'BTC',
      'symbol2': 'USD'
    }
    'price': '607.1700',
    'profit': '-12.48',
  }
}
```

# API Cex Plus

Cex Plus

## Obsługiwane API

- [WebSockets API](#): połączenie z publicznym serwerem WebSocket zapewniające aktualizacje danych rynkowych w czasie rzeczywistym.

## WebSockets API

WebSocket to protokół komunikacyjny full-duplex oparty na TCP. Full-duplex oznacza, że obie strony mogą asynchronicznie wysyłać do siebie wiadomości za pośrednictwem tego samego kanału komunikacyjnego. Niniejsza sekcja opisuje, jakie wiadomości powinny wymieniać Exchange Plus i klient. Wszystkie wiadomości powinny być prawidłowymi obiektami JSON.

WebSocket API jest używany głównie do uzyskiwania informacji lub wykonywania działań, które nie są dostępne lub trudne do wykonania przy użyciu REST API. Jednak niektóre żądania lub działania można realizować zarówno przez REST API, jak i WebSocket API. Exchange Plus wysyła wiadomości do klienta jako odpowiedź na żądanie wysłane wcześniej przez klienta lub jako powiadomienie o zdarzeniu (bez poprzedniego żądania klienta).

## Publiczne wywołania API

Publiczny limit liczby żądań API jest stosowany w celu ochrony systemu przed atakami DDoS i zapewnienia wszystkim klientom równego, stabilnego dostępu do punktów końcowych Exchange Plus API. Publiczne żądania są ograniczane na podstawie adresu IP, z którego są kierowane. Limity żądań są wyznaczane na podstawie kosztu skojarzonego z każdym publicznym wywołaniem API. Domyślnie każde publiczne żądanie ma koszt 1 punktu, jednak dla niektórych żądań koszt ten może być wyższy. Aktualne informacje o kosztach limitów żądań można znaleźć w specyfikacji każdej metody.

Exchange Plus ogranicza wywołania publicznego API do maksymalnie 100 punktów na minutę, biorąc pod uwagę, że każde wywołanie publicznego API ma swój koszt (patrz poniżej). Po osiągnięciu limitu szybkości żądań Exchange Plus odpowiada błędem, wysyła zdarzenie rozłączenia do klienta i następnie zamyka połączenie WS. Exchange Plus wznowi obsługę klienta od początku kolejnej minuty kalendarzowej. W poniższym przykładzie licznik żądań zostanie zresetowany o 11:02:00.000.



|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetTicker</b>             | Metoda ta służy do pobierania aktualnych informacji o tickerze, w tym danych o bieżących cenach, zmianach ceny i wolumenu w ciągu 24 godzin, ostatniej transakcji itp. dla wybranych aktywów.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>GetOrderBook</b>          | Ta metoda umożliwia klientowi otrzymanie aktualnej migawki arkusza zleceń dla określonej pary handlowej.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>GetCandles</b>            | Korzystając z metody Candles, klient może otrzymywać historyczne świece OHLCV różnych rozdzielczości i typów danych. Klient może wskazać dodatkowe filtry przedziału czasowego i limitu, aby odpowiedź była bardziej precyzyjna.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>GetTradeHistory</b>       | Metoda ta umożliwia klientowi pobieranie danych historycznych dotyczących zrealizowanych transakcji dla wybranej pary handlowej. Żądanie historii transakcji można uzupełnić o dodatkowe parametry filtrowania, takie jak zakres czasowy, zakres identyfikatorów transakcji, strona i inne, aby otrzymać transakcje spełniające kryteria żądania.                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>GetServerTime</b>         | Ta metoda służy do uzyskiwania bieżącego czasu na serwerze Exchange Plus. Może być przydatna w aplikacjach wymagających synchronizacji z czasem serwera.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>GetPairsInfo</b>          | Metoda Pair Info umożliwia klientowi pobieranie parametrów dla wszystkich obsługiwanych par handlowych.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>GetCurrenciesInfo</b>     | Metoda Currencies Info umożliwia klientowi uzyskanie parametrów dla wszystkich walut skonfigurowanych w Exchange Plus, a także informacji o dostępności wpłat i wypłat między Exchange Plus a portfelem CEX.IO.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>GetProcessingInfo</b>     | To żądanie umożliwia klientowi uzyskanie szczegółowych informacji o dostępnych opcjach wpłat z zewnętrznych portfeli i wypłat do zewnętrznych portfeli dla każdej obsługiwanej kryptowaluty, w tym nazwy kryptowaluty i dostępnych łańcuchów bloków dla depozytów\wypłat. Ponadto dla każdego obsługiwanego łańcucha bloków podawane są: typ kryptowaluty na danym łańcuchu, aktualna dostępność depozytów\wypłat, minimalne kwoty dla depozytów\wypłat oraz zewnętrzne opłaty za wypłaty. Informacje te zapewniają klientowi większą elastyczność w wyborze łańcucha bloków do uzyskania adresu depozytu i inicjowania zewnętrznych wypłat, co umożliwia wygodniejszy transfer aktywów kryptograficznych do lub z ekosystemu CEX.IO. |
| <b>SubscribeOrderBook</b>    | Klient subskrybujący przez WebSocket może subskrybować kanał arkusza zleceń dla wybranej pary handlowej. W odpowiedzi na żądanie subskrypcji arkusza zleceń klient otrzyma bieżącą (początkową) migawkę arkusza zleceń dla żądanej pary z podanym numerem seqId. Aby śledzić kolejne aktualizacje arkusza zleceń, klient musi subskrybować przez WebSocket wiadomości "order_book_increment", zawierające nazwę pary handlowej, numer seqId oraz zmiany poziomów cenowych ofert kupna i sprzedaży.                                                                                                                                                                                                                                    |
| <b>UnSubscribeOrder-Book</b> | Anuluj subskrypcję kanału arkusza zleceń.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>SubscribeTrade</b>        | Korzystając z metody Trade Subscribe, klient może subskrybować przez WebSocket strumień na żywo zdarzeń transakcyjnych zachodzących na żądanej parze handlowej. W odpowiedzi na żądanie Trade Subscribe klient otrzyma unikalny identyfikator subskrypcji transakcji, który powinien być następnie używany do rezygnacji z subskrypcji, gdy nie jest już ona potrzebna. Klient powinien subskrybować przez WebSocket komunikaty „tradeHistorySnapshot” i „tradeUpdate”, aby otrzymywać migawki historii transakcji oraz zdarzenia transakcji na żywo dla żądanej pary handlowej.                                                                                                                                                      |
| <b>UnSubscribeTrade</b>      | Anuluj subskrypcję kanału transakcji.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Przykład: pobierz najnowszy ticker pary BTC-USD

```

TsgcWebSocketClient* oClient;
TsgcWSAPI_CexPlus* oCexPlus;

void __fastcall OnCexPlusConnectEvent(System::TObject* Sender)
{
    oCexPlus->GetTicker("BTC-USD");
}

```

```

void __fastcall OnCexPlusMessageEvent(System::TObject* Sender, UnicodeString Event, UnicodeString Msg)
{
    ShowMessage("Ticker data: " + Msg);
}

void SetupCexPlus()
{
    oClient = new TsgcWebSocketClient(NULL);
    oCexPlus = new TsgcWSAPI_CexPlus(NULL);
    oClient->Active = true;
    oCexPlus->Client = oClient;
    oCexPlus->OnConnect = OnCexPlusConnectEvent;
    oCexPlus->OnMessage = OnCexPlusMessageEvent;
}

```

## **Prywatne wywołania API**

Exchange Plus używa kluczy API do udzielania dostępu do prywatnych API.

Klient może generować, konfigurować i zarządzać kluczami API, ustawiać poziomy uprawnień, adresy IP na białej liście dla klucza API itp. za pośrednictwem terminala Exchange Plus Web Terminal w sekcji zarządzania kluczami API.

**Limit kluczy API:** Domyślnie klient może mieć do 5 kluczy API.

Aby ograniczyć dostęp do określonych funkcji podczas korzystania z kluczy API, należy zdefiniować konkretny zestaw uprawnień dla każdego klucza API. W razie potrzeby zdefiniowany zestaw uprawnień można później edytować.

Dla kluczy API dostępne są następujące poziomy uprawnień:

- **Read:** poziom uprawnień do przeglądania danych związanych z kontem, odbierania raportów, subskrybowania danych rynkowych itp.
- **Trade:** poziom uprawnień umożliwiający składanie i anulowanie zleceń w imieniu konta.
- **Fundusze wewnętrzne:** poziom uprawnień umożliwiający transfer środków między kontami (między subkontami lub kontem głównym a subkontami) w ramach portfela CEX.IO Exchange Plus.
- **Funds Wallet:** poziom uprawnień umożliwiający transfer środków z kont portfela CEX.IO Exchange Plus (konto główne i subkonta) do CEX.IO Wallet i z powrotem.

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetCurrentFee</b>         | <p>Ta metoda wskazuje bieżące opłaty w określonym momencie czasu, z uwzględnieniem aktualnego 30-dniowego wolumenu klienta i dnia tygodnia (opłaty mogą być różne np. w weekendy).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>GetFeeStrategy</b>        | <p>Strategia opłat zwraca wszystkie dostępne opcje opłat, które mogą być zastosowane dla klienta, biorąc pod uwagę jego wolumen obrotu, dzień tygodnia, pary, grupy par itp. Metoda ta dostarcza informacji o ogólnej strategii opłat obejmującej wszystkie możliwe wartości opłat transakcyjnych. Aby otrzymać bieżące opłaty transakcyjne na podstawie wolumenu obrotu klienta z ostatnich 30 dni, należy użyć metody [Current Fee]. Aby uzyskać bieżący 30-dniowy wolumen obrotu, należy użyć metody [Volume].</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>GetVolume</b>             | <p>To żądanie umożliwia klientowi pobranie wolumenu obrotu z ostatnich 30 dni w ekwiwalencie USD.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>CreateAccount</b>         | <p>To żądanie umożliwia klientowi tworzenie nowych kont podrzędnych. Domyślnie klient może mieć do 5 kont podrzędnych, wliczając konto główne.</p> <p>Korzystając z metody Account Status V3, klient może sprawdzić bieżące saldo i jego orientacyjny odpowiednik w przeliczonej walucie (domyślnie „USD”), kwoty zablokowane w otwartych (aktywnych) zleceniach dla każdego subkonta i waluty. Jeśli dla klienta dostępne jest saldo opłaty transakcyjnej, odpowiedź zawiera również ogólne dane salda opłaty transakcyjnej, takie jak nazwa promocji, nazwa waluty, łączne saldo i data wygaśnięcia promocji dotyczącej salda opłaty transakcyjnej. Klient jest odpowiedzialny za śledzenie dostępnego salda transakcyjnego swojego subkonta, czyli bieżącego salda subkonta pomniejszonego o kwotę zablokowaną w otwartych (aktywnych) zleceniach na subkoncie.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>GetAccountStatus</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>GetOrders</b>             | <p>To żądanie umożliwia klientowi uzyskanie informacji o jego zleceniach.</p> <p>Klient może składać nowe zlecenia przez WebSocket API za pomocą żądania Do My New Order Request. Wraz z odpowiedzią na to żądanie, Exchange Plus wysyła do klienta komunikaty Account Event i Execution Report, jeśli żądanie zakończyło się sukcesem. Komunikat odpowiedzi wskazuje ostatni aktualny status zlecenia dostępny w systemie w momencie wysłania odpowiedzi. Jeśli klient nie otrzymał komunikatu odpowiedzi na Do My New Order Request, może zapytać o bieżący status zlecenia za pomocą żądania Get My Orders Request z parametrem clientOrderId. Podczas wysyłania żądania nowego zlecenia zdecydowanie zaleca się użycie parametru clientOrderId, który odpowiada konkretnemu żądaniu nowego zlecenia po stronie klienta. Exchange Plus unika wielokrotnego składania zleceń z tym samym clientOrderId. Jeśli zidentyfikowano więcej niż jedno nowe zlecenie z identycznym clientOrderId i innymi parametrami zlecenia, Exchange Plus składa tylko pierwsze zlecenie i zwraca jego status klientowi w odpowiedzi na drugie i kolejne żądania nowego zlecenia z tymi samymi parametrami. Jeśli zidentyfikowano więcej niż jedno nowe zlecenie z identycznym clientOrderId, ale z różnymi innymi parametrami zlecenia, Exchange Plus przetwarza tylko pierwsze zlecenie i odrzuca drugie i kolejne żądania nowego zlecenia z tym samym clientOrderId, ale z różnymi innymi parametrami.</p> |
| <b>NewOrder</b>              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>NewMarketOrder</b>        | <p>Składa nowe zlecenie rynkowe.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>NewLimitOrder</b>         | <p>Składa nowe zlecenie z limitem.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>CancelOrder</b>           | <p>Klient może anulować zlecenia. Wraz z odpowiedzią na to żądanie giełda Exchange Plus wysyła do klienta wiadomości Account Event i Execution Report, jeśli żądanie zakończyło się powodzeniem. Ponadto, jeśli żądanie anulowania zlecenia zostanie odrzucone, Exchange Plus wysyła wiadomość Order Cancellation Rejection.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>CancelAllOrders</b>       | <p>Klient może anulować wszystkie otwarte zlecenia przez WebSocket API. Wraz z odpowiedzią na to żądanie Exchange Plus rozpocznie proces anulowania wszystkich otwartych zleceń i wyśle odpowiednie wiadomości Account Event i Execution Report do klienta.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>GetTransactionHistory</b> | <p>To żądanie umożliwia klientowi sprawdzenie jego transakcji finansowych (wpłaty, wypłaty, transfery wewnętrzne, prowizje lub transakcje handlowe).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>GetFundingHistory</b>     | <p>Żądanie to umożliwia klientowi wyszukanie transakcji depozytowych i wypłat.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>InternalTransfer</b>      | <p>Klient może żądać transferu środków między swoimi kontami podrzędnymi lub między kontem głównym a kontem podrzędnym. Exchange Plus nie pobiera od Klienta żadnej prowizji za transfery środków między jego kontami. Wraz z odpowiedzią na to żądanie Exchange Plus wysyła do Klienta wiadomości o zdarzeniu na koncie, jeśli żądanie zakończy się pomyślnie.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|                                 |                                                                                                                                                                                                                                                                                                                                |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GetDepositAddress</b>        | Metoda ta może być używana przez klienta w celu uzyskania adresu kryptowalutowego do wpłaty środków. Adres wpłaty może być generowany dla kont głównych i podkont. Listę dostępnych sieci blockchain do generowania adresów wpłat klient może uzyskać poprzez żądanie Get Processing Info.                                     |
| <b>FundsDepositFrom-Wallet</b>  | Klient może wpłacić środki z portfela CEX.IO na konto Exchange Plus. System unika przetwarzania wielu żądań depozytów z tym samym clientTxId. Jeśli zostanie odebranych wiele żądań depozytów z identycznym clientTxId, system przetworzy tylko pierwsze żądanie i odrzuci drugie oraz kolejne żądania z tym samym clientTxId. |
| <b>FundsWithdrawal-ToWallet</b> | Klient może wypłacić środki z konta Exchange Plus do portfela CEX.IO. System zapobiega wielokrotnym żądaniom wypłaty z tym samym clientTxId. Jeśli zostanie otrzymanych wiele żądań wypłaty z identycznym clientTxId, system przetwarza tylko pierwsze żądanie i odrzuca kolejne żądania z tym samym clientTxId.               |
| <b>GetWalletBalance</b>         | Pobiera informacje o saldzie portfela CEX.IO.                                                                                                                                                                                                                                                                                  |
| <b>SubscribeAccountEvents</b>   | Subskrybuje powiadomienia o zdarzeniach na koncie w czasie rzeczywistym (zmiany salda, realizacje zleceń)                                                                                                                                                                                                                      |
| <b>UnSubscribeAccountEvents</b> | Anuluje subskrypcję powiadomień o zdarzeniach konta.                                                                                                                                                                                                                                                                           |

Przykład: pobierz zlecenia.

```

TsgcWebSocketClient* oClient;
TsgcWSAPI_CexPlus* oCexPlus;

void __fastcall OnCexPlusAuthenticatedEvent(System::TObject* Sender)
{
    oCexPlus->GetOrders();
}

void __fastcall OnCexPlusMessageEvent(System::TObject* Sender, UnicodeString Event, UnicodeString Msg)
{
    ShowMessage("Orders: " + Msg);
}

void SetupCexPlus()
{
    oClient = new TsgcWebSocketClient(NULL);
    oCexPlus = new TsgcWSAPI_CexPlus(NULL);
    oClient->Active = true;
    oCexPlus->Client = oClient;
    oCexPlus->CexPlus->ApiKey = "your-api-key";
    oCexPlus->CexPlus->ApiSecret = "your-api-secret";
    oCexPlus->OnCexPlusAuthenticated = OnCexPlusAuthenticatedEvent;
    oCexPlus->OnMessage = OnCexPlusMessageEvent;
}

```

# API Discord

## Discord

**Uwaga:** Począwszy od wersji API v10, domena API Discord zmieniła się z discordapp.com na discord.com. Komponent został zaktualizowany do wersji API 10 i nowej domeny.

Bramy (Gateways) to mechanizm Discord do komunikacji w czasie rzeczywistym za pośrednictwem bezpiecznych WebSockets. Klienci odbierają zdarzenia i dane przez bramę, z którą są połączeni, i wysyłają dane przez REST API.

## Autoryzacja

W pierwszej kolejności należy wygenerować nowego bota i skopiować token bota, który będzie używany do uwierzytelniania przez API. Następnie należy ustawić ten token w komponencie API.

```
TsgcWSAPI_Discord1->DiscordOptions->BotOptions->Token = "...bot token here...";
```

## Intencje

Utrzymywanie aplikacji ze stanem może być trudne ze względu na ilość danych do przetworzenia, zwłaszcza w dużej skali. Gateway Intents to system pomagający zmniejszyć to obciążenie obliczeniowe.

Podczas identyfikacji w bramce można określić parametr intents, który umożliwia warunkową subskrypcję predefiniowanych "intencji" — grup zdarzeń zdefiniowanych przez Discord. Nieuwzględnienie danej intencji skutkuje brakiem otrzymywania zdarzeń bramki należących do tej grupy. Prawidłowe intencje to (wartość zero oznacza odbieranie wszystkich zdarzeń):

GUILDS (1 << 0) = Integer (1)

- GUILD\_CREATE
- GUILD\_DELETE
- GUILD\_ROLE\_CREATE
- GUILD\_ROLE\_UPDATE
- GUILD\_ROLE\_DELETE
- CHANNEL\_CREATE
- CHANNEL\_UPDATE

- CHANNEL\_DELETE

- CHANNEL\_PINS\_UPDATE

GUILD\_MEMBERS (1 << 1) = Integer (2)

- GUILD\_MEMBER\_ADD

- GUILD\_MEMBER\_UPDATE

- GUILD\_MEMBER\_REMOVE

GUILD\_BANS (1 << 2) = Integer (4)

- GUILD\_BAN\_ADD

- GUILD\_BAN\_REMOVE

GUILD\_EMOJIS (1 << 3) = Integer (8)

- GUILD\_EMOJIS\_UPDATE

GUILD\_INTEGRATIONS (1 << 4) = Integer (16)

- GUILD\_INTEGRATIONS\_UPDATE

GUILD\_WEBHOOKS (1 << 5) = Integer (32)

- WEBHOOKS\_UPDATE

GUILD\_INVITES (1 << 6) = Integer (64)

- INVITE\_CREATE

- INVITE\_DELETE

GUILD\_VOICE\_STATES (1 << 7) = Integer (128)

- VOICE\_STATE\_UPDATE

GUILD\_PRESENCES (1 << 8) = Integer (256)

- PRESENCE\_UPDATE

GUILD\_MESSAGES (1 << 9) = Integer (512)

- MESSAGE\_CREATE

- MESSAGE\_UPDATE

- MESSAGE\_DELETE

GUILD\_MESSAGE\_REACTIONS (1 << 10) = Integer (1024)

- MESSAGE\_REACTION\_ADD

- MESSAGE\_REACTION\_REMOVE

- MESSAGE\_REACTION\_REMOVE\_ALL

- MESSAGE\_REACTION\_REMOVE\_EMOJI

GUILD\_MESSAGE\_TYPING (1 << 11) = Integer (2048)

- TYPING\_START

DIRECT\_MESSAGES (1 << 12) = Integer (4096)

- CHANNEL\_CREATE

- MESSAGE\_CREATE

- MESSAGE\_UPDATE

- MESSAGE\_DELETE

- CHANNEL\_PINS\_UPDATE

DIRECT\_MESSAGE\_REACTIONS (1 << 13) = Integer (8192)

- MESSAGE\_REACTION\_ADD

- MESSAGE\_REACTION\_REMOVE

- MESSAGE\_REACTION\_REMOVE\_ALL

- MESSAGE\_REACTION\_REMOVE\_EMOJI

DIRECT\_MESSAGE\_TYPING (1 << 14) = Integer (16384)

- TYPING\_START

## Heartbeat

Sygnały heartbeat są obsługiwane automatycznie przez komponent, bez konieczności ich ręcznej obsługi. Gdy klient łączy się z serwerem, serwer wysyła odpowiedź HELLO z interwałem heartbeat, a komponent odczytuje tę odpowiedź i automatycznie dostosowuje heartbeat, aby wysyłać ping co określoną liczbę sekund. Czasami serwer może wysłać ping do klienta; jest to również obsługiwane automatycznie przez klienta.

## Połączenie gotowe

Gdy połączenie jest gotowe, po pomyślnym zalogowaniu i autoryzacji przez serwer, wywoływane jest zdarzenie **OnDiscordReady**, po czym można rozpocząć odbieranie aktualizacji z serwera.

## Wznawianie połączenia

Jeśli połączenie zostaje nieoczekiwanie zamknięte i klient próbuje ponownie się połączyć, wywoływane jest zdarzenie **OnDiscordBeforeReconnect**. Komponent automatycznie zapisuje wszystkie dane potrzebne do pomyślnego wznowienia, ale parametry można zmienić w razie potrzeby. Jeśli nie chcesz ponownie łączyć się i rozpocząć nową czystą sesję, ustaw Reconnect na False.

Jeśli sesja zostanie wznowiona, wywoływane jest zdarzenie **OnDiscordResumed**. Jeśli to nowa sesja, wywoływane będzie **OnDiscordReady**.

## Dispatch Events

Zdarzenia są wysyłane przez **OnDiscordDispatch**; tutaj można odczytywać zdarzenia wysyłane przez serwer do klienta.

```
void OnDiscordDispatch(TObject *Sender, const string aEvent, const string RawData)
{
    DoLog("#discord dispatch: " + aEvent + " " + RawData);
}
```

**aEvent** parametr zawiera nazwę zdarzenia.

**RawData** zawiera pełną wiadomość JSON.

## Żądania HTTP

W celu żądania informacji o guildach, użytkownikach lub aktualizacji danych... zamiast używać wiadomości bramki WebSocket, Discord wymaga użycia żądań HTTP. Poniżej znajdziesz wszystkie metody dostępne do wykonywania żądań HTTP:

```
function GET_Request(const aPath: String): string;

function POST_Request(const aPath, aMessage: String): string;

function PUT_Request(const aPath, aMessage: String): string;

function PATCH_Request(const aPath, aMessage: String): string;

function DELETE_Request(const aPath: String): string;
```

**Przykład:** pobierz informacje o bieżącym użytkowniku

```
result = GET_Request("/users/@me");
```



przykładowa odpowiedź z serwera:

```
{
  "id": "637423922035480852",
  "username": "test",
  "avatar": null,
  "discriminator": "5125",
  "bot": true,
  "email": null,
  "verified": true,
  "locale": "en-US",
  "mfa_enabled": false,
  "flags": 0
}
```

# API | OpenAI

OpenAI Realtime API umożliwia interakcje niskolatencyjne z obsługą wielu modalności, w tym konwersację mowy na mowę oraz transkrypcję w czasie rzeczywistym.

Komponent **TsgcWSAPI\_OpenAI** implementuje RealTime OpenAI API.

## Konfiguracja

Należy użyć właściwości **method**, aby wybrać tryb **Conversation** lub **Transcription**; obecnie obsługiwany jest wyłącznie tryb Transcription.

Właściwość **InputAudio** umożliwia dostosowanie następujących danych:

- **Language:** przykład english (wartość = 'en').
- **Model:** określa, który model zostanie użyty, przykład: whisper-1
- **Prompt:** opcjonalna instrukcja dla modelu.

## OpenAI

OpenAI API używa kluczy API do uwierzytelniania. Należy odwiedzić stronę [Klucze API](#), aby pobrać klucz API używany w żądaniach.

Klucz API jest informacją poufną — nie należy udostępniać go innym ani umieszczać w kodzie po stronie klienta (przeglądarki, aplikacje). Żądania produkcyjne muszą być kierowane przez własny serwer backendowy, gdzie klucz API może być bezpiecznie ładowany ze zmiennej środowiskowej lub usługi zarządzania kluczami.

Ten **klucz API** musi być skonfigurowany we właściwości **OpenAIOptions.ApiKey** komponentu. Opcjonalnie, dla użytkowników należących do wielu organizacji, można ustawić organizację we właściwości **OpenAIOptions.Organization**, jeśli konto należy do organizacji.

Po skonfigurowaniu klucza API poniżej znajduje się lista dostępnych funkcji do interakcji z OpenAI API.

## Azure

Klient obsługuje Microsoft Azure OpenAI Services, dzięki czemu można używać konta Azure do interakcji z Azure OpenAI API. Aby skonfigurować klienta do pracy z Azure, należy wykonać następujące kroki:

1. Skonfiguruj właściwość **OpenAIOptions.Provider** = oapvAzure
2. Ustaw wartości ResourceName i DeploymentId (wartości te można znaleźć na koncie Azure)
  1. **OpenAIOptions.AzureOptions.ResourceName** = <nazwa zasobu>.
  2. **OpenAIOptions.AzureOptions.DeploymentId** = <identyfikator wdrożenia>.
3. Ustaw klucz API swojego konta Azure
  1. **OpenAIOptions.ApiKey** = <azure api key>.

## Przykład transkrypcji

Poniżej znajduje się przykład transkrypcji w czasie rzeczywistym z użyciem OpenAI API

```
TsgcWebSocketClient *WSClient = new TsgcWebSocketClient(NULL);
TsgcAudioRecorderWave *oAudio = new TsgcAudioRecorderWave(NULL);
TsgcWSAPI_OpenAI *OpenAI = new TsgcWSAPI_OpenAI(NULL);
OpenAI->Client = WSClient;
OpenAI->AudioRecorder = oAudio;

OpenAI->OpenAIOptions->APIKey = "your-api-key-here";
OpenAI->OpenAIOptions->Method = sgcoaimTranscription;
OpenAI->OpenAIOptions->Provider = sgcoaipOpenAI;
OpenAI->InputAudio->Language = "en";
OpenAI->InputAudio->Model = "whisper-1";

void __fastcall TForm1::OnOpenAIAudioTranscriptionCompleted(TObject *Sender, TsgcWSOpenAIConversation_Item_Completed *aItem)
{
    Log("#transcription_completed: " + aItem->Transcript);
}
```

# API MEXC

**MEXC** to globalna giełda kryptowalut udostępniająca dane rynkowe i konta w trybie strumieniowym za pośrednictwem bezpiecznych połączeń WebSocket.

WebSocket Spot API jest zgodne ze specyfikacjami opublikowanymi w dokumentacji [Market Streams](#) oraz [User Data Streams](#). Kanały rynkowe są dostarczane jako ramki Protocol Buffers, podczas gdy prywatne strumienie używają formatu JSON. Komponent **TsgcWSAPI\_MEXC** zarządza cyklem życia połączenia, uwierzytelnianiem oraz utrzymaniem klucza nasłuchu, dzięki czemu aplikacje mogą skupić się na przetwarzaniu przychodzących danych.

Oprócz strumieniowania przez WebSockets, zestaw narzędzi udostępnia również kompletne Spot HTTP API. Komponent **TsgcHTTP\_API\_MEXC\_Spot** (jednostka **sgcHTTP\_API\_MEXC**) opakuje każdy oficjalny punkt końcowy REST, umożliwiając aplikacjom pobieranie danych rynkowych lub handel z użyciem podpisanych żądań.

## Konfiguracja komponentu

Podstawowy punkt końcowy WebSocket to **wss://wbs-api.mexc.com/ws**. Gdy **MEXCUserDataStreams.UserStream** jest włączony i podano prawidłowy klucz API, komponent automatycznie żąda klucza nasłuchiwanie i dołącza go do URL połączenia.

Należy umieścić komponenty **TsgcWebSocketClient** i **TsgcWSAPI\_MEXC** na formularzu (lub utworzyć je w kodzie), przypisać instancję klienta do komponentu API i aktywować klienta, aby rozpocząć sesję.

```
TsgcWebSocketClient *WSClient = new TsgcWebSocketClient();
TsgcWSAPI_MEXC *MEXC = new TsgcWSAPI_MEXC();
MEXC->Client = WSClient;
MEXC->MEXCAPI->ApiKey = "YOUR_KEY";
MEXC->MEXCAPI->ApiSecret = "YOUR_SECRET";
MEXC->MEXCUserDataStreams->UserStream = true;
if (WSClient->Connect()) {
    MEXC->SubscribeTrade("BTCUSD");
}
```

## Właściwości

- **Klient:** odwołanie do transportu **TsgcWebSocketClient**. Komponent włącza bicie serca klienta (15 sekund) w celu utrzymania sesji przy życiu.
- **MEXCAPI.ApiKey / ApiSecret:** Dane uwierzytelniające Spot API. Klucz jest wymagany dla prywatnych kanałów; sekret jest potrzebny do podpisywania żądań klucza nasłuchiwanie.
- **MEXCUserDataStreams.UserStream:** gdy wartość wynosi **True** (domyślnie), komponent pobiera klucz nasłuchu przed połączeniem i odświeża go co 10 minut. Wyłączenie tej opcji umożliwia pracę wyłącznie z publicznymi kanałami danych.

- **MEXCUserDataStreams.ListenKeyOnDisconnect:** akcja wykonywana po rozłączeniu WebSocket, gdy klucz nasłuchiwanie jest aktywny:
  - **blkmxodDeleteListenKey:** unieważnia klucz nasłuchiwanie przez REST (domyślnie).
  - **blkmxodClearListenKey:** zachowuje klucz nasłuchiwanie po stronie serwera i czyści wartość z pamięci podręcznej.
  - **blkmxodDoNothing:** ponownie używa tego samego klucza nasłuchiwanie przy następnym połączeniu.
- **ListenKey:** właściwość tylko do odczytu udostępniająca aktywny klucz nasłuchu zwrócony przez API REST.
- **RawMessages:** jeśli włączone, przychodzące ramki tekstowe nie są analizowane i są przekazywane jako zwykły JSON przez dziedziczone zdarzenia WebSocket.

## Zdarzenia

- **OnConnect / OnDisconnect:** wywoływane, gdy bazowa sesja WebSocket zostaje nawiązana lub zakończona.
- **OnMEXCMarketStream:** wywoływane dla każdej ramki rynkowej Protobuf. Procedura obsługi otrzymuje obiekt **TsgcMEXCSpotProtoMessage**, którego właściwość **MessageType** identyfikuje ładunek (Trade, KLine, DiffDepth, LimitDepth, BookTicker, BookTickerBatch, MiniTicker lub MiniTickers). Do sprawdzania zdekodowanej struktury należy używać klas pomocniczych z modułu **sgcWebSocket\_API\_MEXC\_Proto**.
- **OnMEXCResponse:** odbiera potwierdzenia subskrypcji i odpowiedzi ping/pong w formacie JSON (**Id**, **Code**, **Message**).
- **OnMEXCUserDataStream:** wywoływane po nadejściu prywatnych wiadomości dotyczących konta (zlecenia, transakcje, aktualizacje salda). Dostarczana jest nazwa kanału i surowy ładunek JSON.
- **OnMEXCException:** przechwytuje wyjątki REST lub WebSocket (błędy klucza nasłuchiwanie, awarie HTTP, problemy z parsowaniem itp.).

Poniższy przykład pokazuje, jak zdekodować ramkę transakcji:

## Metody subskrypcji WebSocket

Komponent udostępnia funkcje pomocnicze dla każdego publicznego kanału Spot. Należy wywołać metodę **Subscribe\***, aby rozpocząć strumieniowanie danych, oraz odpowiednią metodę **UnSubscribe\***, aby je zatrzymać.

## Kanały publiczne

|                          |                       |                                                                                                |
|--------------------------|-----------------------|------------------------------------------------------------------------------------------------|
| SubscribeTrade           | Symbol, interwał (ms) | Zagregowane realizacje transakcji dostarczane jako Protobuf <b>publicdeals</b> .               |
| SubscribeKline           | Symbol, interwał      | Aktualizacje świec dla konfigurowalnych interwałów (od 1 minuty do 1 miesiąca).                |
| SubscribeDiffDepth       | Symbol, interwał (ms) | Przyrostowe delta głębokości arkusza zleceń do lokalnej obsługi księgi.                        |
| SubscribeBookDepth       | Symbol, Levels        | Cykliczne migawki arkusza zleceń z limitem z żadaną głębokością (domyślnie 5).                 |
| SubscribeBookTicker      | Symbol, interwał (ms) | Aktualizacje najlepszej oferty kupna/sprzedaży dla pojedynczej pary handlowej.                 |
| SubscribeBookTickerBatch | Flaga UTC             | Zagregowane aktualizacje tickera księgi zleceń dla wielu symboli.                              |
| SubscribeMiniTickers     | Flaga UTC             | Statystyki miniaturowych tikerów krocząco za 24 godziny dla wszystkich aktywnych instrumentów. |
| SubscribeMiniTicker      | Symbol, flaga UTC     | Mini ticker dla pojedynczego symbolu.                                                          |

## Kanały prywatne

Tematy prywatne wymagają ważnego klucza API. Komponent subskrybuje przy użyciu aktywnego klucza nasłuchiwanie i odnawia go cyklicznie.

|                        |                                                                      |
|------------------------|----------------------------------------------------------------------|
| SubscribeAccountUpdate | Zmiany salda konta i pozycji.                                        |
| SubscribeAccountDeals  | Raporty z realizacji zleceń wypełnionych.                            |
| SubscribeAccountOrders | Aktualizacje cyklu życia zlecenia (nowe, anulowane, odrzucone itp.). |

Należy używać odpowiednich metod **UnSubscribe\*** w celu zakończenia strumienia. Składnik automatycznie wysyła polecenia **PING** i sprawdza poprawność zwróconych odpowiedzi **PONG** w celu monitorowania łączności.

## Publiczne endpointy REST

Publicznie dostępne metody REST.

|                        |                                                                                                                                                                              |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ping                   | Sprawdzenie łączności wywołujące <b>/api/v3/ping</b> , które zwraca <b>True</b> , gdy giełda odpowie pustym obiektem.                                                        |
| GetServerTime          | Pobiera aktualny znacznik czasu giełdy z punktu końcowego <b>/api/v3/time</b> .                                                                                              |
| GetExchangeInformation | Zwraca reguły handlowe i metadane symbolu z <b>/api/v3/exchangeInfo</b> .                                                                                                    |
| GetOrderBook           | Pobiera migawki księgi zleceń z punktu końcowego <b>/api/v3/depth</b> ; opcjonalny parametr <b>limit</b> pozwala kontrolować liczbę poziomów (domyślnie 100).                |
| GetTrades              | Pobiera ostatnie transakcje publiczne z <b>/api/v3/trades</b> z opcjonalnym <b>limitem</b> (domyślnie 100).                                                                  |
| GetHistoricalTrades    | Dostarcza historyczne dane transakcji z opcjonalnymi filtrami <b>limit</b> i <b>fromId</b> .                                                                                 |
| GetAggregateTrades     | Zwraca skompresowane/zagregowane transakcje przez <b>/api/v3/aggTrades</b> z obsługą opcjonalnych filtrów <b>limit</b> , <b>fromId</b> , <b>startTime</b> i <b>endTime</b> . |
| GetKlines              | Pobiera świece z <b>/api/v3/klines</b> obsługując filtry zakresu czasu i limitu.                                                                                             |
| GetAveragePrice        | Zwraca bieżącą ważoną średnią cenę z punktu końcowego <b>/api/v3/avgPrice</b> .                                                                                              |
| Get24hrTicker          | Pobiera 24-godzinne statystyki ticker dla jednego lub wszystkich symboli za pośrednictwem <b>/api/v3/ticker/24hr</b> .                                                       |
| GetPriceTicker         | Pobiera najnowszą cenę za pomocą <b>/api/v3/ticker/price</b> ; podanie pustego symbolu skutkuje pobraniem cen dla wszystkich par handlowych.                                 |
| GetBookTicker          | Pobiera najlepsze oferty kupna/sprzedaży z <b>/api/v3/ticker/bookTicker</b> z obsługą żądań dla pojedynczego symbolu lub całej giełdy.                                       |

## Prywatne punkty końcowe REST

Prywatne endpointy wymagają prawidłowych kluczy API i podpisanego ciągu zapytania. Komponent automatycznie dołącza znacznik czasu, recvWindow oraz podpis HMAC.

|                        |                                                                                                                                                                                                                                                                                   |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetAccountInformation  | Zwraca salda i uprawnienia z <b>/api/v3/account</b> .                                                                                                                                                                                                                             |
| GetOpenOrders          | Wyświetla bieżące otwarte zlecenia (opcjonalnie filtrowane według symbolu).                                                                                                                                                                                                       |
| GetAllOrders           | Pobiera historyczne zlecenia z opcjonalnymi filtrami czasu i limitu.                                                                                                                                                                                                              |
| GetOrder               | Pobiera status konkretnego zlecenia, podając <b>orderId</b> lub <b>origClientOrderId</b> .                                                                                                                                                                                        |
| GetMyTrades            | Wyświetla prywatne realizacje transakcji z opcjonalnymi filtrami <b>limit</b> , <b>fromId</b> , <b>startTime</b> i <b>endTime</b> .                                                                                                                                               |
| NewOrder               | Składa aktywne zlecenie na <b>/api/v3/order</b> (rynkowe, z limitem, stop itp.) z opcjonalnymi parametrami <b>timeInForce</b> , <b>quantity</b> , <b>quoteOrderQty</b> , <b>price</b> , <b>newClientOrderId</b> , <b>stopPrice</b> , <b>icebergQty</b> i dodatkowymi parametrami. |
| TestNewOrder           | Wysyła żądanie wyłącznie weryfikacyjne do <b>/api/v3/order/test</b> , przyjmując ten sam zestaw parametrów co <b>NewOrder</b> .                                                                                                                                                   |
| CancelOrder            | Anuluje konkretne zlecenie w <b>/api/v3/order</b> używając <b>orderId</b> lub <b>origClientOrderId</b> .                                                                                                                                                                          |
| CancelAllOrders        | Masowo anuluje wszystkie otwarte zlecenia dla danego symbolu przez <b>/api/v3/openOrders</b> .                                                                                                                                                                                    |
| GetSubAccounts         | Wyświetla zarządzane konta podrzędne ( <b>/api/v3/sub-account/list</b> ).                                                                                                                                                                                                         |
| GetSubAccountAssets    | Zwraca salda dla określonego subkonta.                                                                                                                                                                                                                                            |
| TransferSubAccount     | Transferuje aktywa między subkontami przez <b>/api/v3/sub-account/transfer</b> ; należy podać kwotę i opcjonalnie <b>typ</b> transferu.                                                                                                                                           |
| GetDepositAddress      | Pobiera adresy depozytów z opcjonalnym filtrem <b>sieci</b> .                                                                                                                                                                                                                     |
| GetDepositHistory      | Pobiera rekordy depozytów z opcjonalnymi filtrami <b>coin</b> , <b>status</b> , <b>startTime</b> i <b>endTime</b> .                                                                                                                                                               |
| GetWithdrawHistory     | Zwraca historię wypłat filtrowaną według <b>waluty</b> , <b>statusu</b> i opcjonalnego zakresu czasowego.                                                                                                                                                                         |
| Wypłata                | Przesyła żądanie wypłaty przez <b>/api/v3/capital/withdraw/apply</b> , zawierające wymagane pola <b>coin</b> , <b>address</b> i <b>amount</b> , a także opcjonalne <b>network</b> , <b>addressTag</b> , <b>withdrawOrderId</b> i dodatkowe parametry.                             |
| BatchOrders            | Złóż zlecenia wsadowe dla wielu symboli w jednym żądaniu.                                                                                                                                                                                                                         |
| GetTradeFee            | Pobierz stawkę opłaty transakcyjnej dla określonego symbolu.                                                                                                                                                                                                                      |
| GetDepositHistory      | Pobiera rekordy depozytów z opcjonalnymi filtrami <b>coin</b> , <b>status</b> , <b>startTime</b> i <b>endTime</b> .                                                                                                                                                               |
| CancelWithdraw         | Anuluj oczekujące żądanie wypłaty według identyfikatora wypłaty.                                                                                                                                                                                                                  |
| CreateInternalTransfer | Transferuj aktywa między kontami wewnętrznie.                                                                                                                                                                                                                                     |
| GetTransferHistory     | Pobierz historię wewnętrznych transferów między kontami.                                                                                                                                                                                                                          |

# API MEXC Futures

**MEXC** kontrakty wieczyste i terminowe udostępniają dedykowane API strumieniowania opisane w oficjalnej [specyfikacji Futures WebSocket](#). Komponent **TsgcWSAPI\_MEXC\_Futures** hermetyzuje połączenie, uzgadnianie logowania i zarządzanie tematami wymagane do konsumowania danych dotyczących instrumentów pochodnych w czasie rzeczywistym.

MEXC utrzymuje również interfejs REST do obrotu instrumentami pochodnymi. Komponent **TsgcHTTP\_API\_MEXC\_Futures** zawarty w pakiecie **sgcHTTP\_API\_MEXC** uzupełnia kanały WebSocket o pomocnicze funkcje HTTP mapowane jeden do jednego na oficjalne punkty końcowe.

## Konfiguracja komponentu

Punkt końcowy Futures WebSocket to **wss://contract.mexc.com/edge**. Wiadomości są kodowane jako obiekty JSON, a giełda wymaga okresowych ramek **ping/pong**, które są obsługiwane automatycznie przez komponent.

Aby otrzymywać prywatne powiadomienia (dane zleceń i konta), konieczne jest uwierzytelnienie. Należy ustawić **MEXCAPI.ApiKey** i **MEXCAPI.ApiSecret** przed aktywowaniem klienta WebSocket. Gdy dane uwierzytelniające są obecne, komponent podpisuje żądanie logowania (HMAC SHA256) natychmiast po nawiązaniu połączenia gniazda.

```
TsgcWebSocketClient *WSClient = new TsgcWebSocketClient();
TsgcWSAPI_MEXC_Futures *Futures = new TsgcWSAPI_MEXC_Futures();
Futures->Client = WSClient;
Futures->MEXCAPI->ApiKey = "YOUR_KEY";
Futures->MEXCAPI->ApiSecret = "YOUR_SECRET";
if (WSClient->Connect()) {
    Futures->SubscribeDepth("BTC_USDT", true);
}
```

## Właściwości

- **Client:** Transport WebSocket używany przez komponent API. Obsługa heartbeat jest włączona w celu utrzymania aktywności gniazda.
- **MEXCAPI.ApiKey / ApiSecret:** wymagane do podpisania żądania logowania. Bez danych uwierzytelniających dostępne są wyłącznie kanały publiczne.
- **RawMessages:** po włączeniu pomija parser JSON, dzięki czemu surowe ramki tekstowe są udostępniane przez podstawowe zdarzenia.

## Zdarzenia

- **OnConnect / OnDisconnect:** powiadomienia o cyklu życia WebSocket.
- **OnMEXCLogin:** wywoływane po pomyślnym uwierzytelnionym logowaniu.



- **OnMEXCSubscribed / OnMEXCUnsubscribed:** potwierdzenie zmian subskrypcji. Zdarzenie udostępnia nazwę kanału zwróconą przez MEXC.
- **OnMEXCMessage:** dostarcza każdą aktualizację rynkową lub dotyczącą konta wraz z identyfikatorem kanału.
- **OnMEXCError:** wywoływane gdy serwer odpowiada ładunkiem **rs.error** (nieprawidłowe parametry, błąd uwierzytelnienia itp.).
- **OnMEXCException:** zgłasza lokalne wyjątki raised podczas przetwarzania danych JSON lub wykonywania operacji sieciowych.

Poniższy moduł obsługi rejestruje potwierdzenia subskrypcji i wyświetla przychodzące migawki głębokości rynku:

## Metody subskrypcji WebSocket

Każda metoda pomocnicza opakuje żądanie **sub/unsb** zgodnie z opisem w oficjalnym API. Do anulowania strumienia należy użyć odpowiadającej metody **UnSubscribe\***.

|                      |                  |                                                                                                                              |
|----------------------|------------------|------------------------------------------------------------------------------------------------------------------------------|
| SubscribeDeal        | Symbol           | Transakcje wykonane na kontrakcie (kanał: <b>deal</b> ).                                                                     |
| SubscribeTickers     | –                | Globalne statystyki dla wszystkich kontraktów.                                                                               |
| SubscribeTicker      | Symbol           | Podsumowanie notowań dla pojedynczego instrumentu.                                                                           |
| SubscribeDepth       | Symbol, Compress | Przyrostowe aktualizacje arkusza zleceń.                                                                                     |
| SubscribeDepthFull   | Symbol, Level    | Pełne migawki głębokości z konfigurowalną liczbą poziomów (domyślnie 20).                                                    |
| SubscribeKline       | Symbol, interwał | Dane świecowe dla obsługiwanych przedziałów czasowych (Min1, Min5, Min15, Min30, Min60, Hour4, Hour12, Day1, Week1, Month1). |
| SubscribeFundingRate | Symbol           | Najnowsze ogłoszenia dotyczące stopy finansowania.                                                                           |
| SubscribeIndexPrice  | Symbol           | Strumień bazowej ceny indeksu.                                                                                               |
| SubscribeFairPrice   | Symbol           | Aktualizacje ceny mark (godziwej) przesyłane przez giełdę.                                                                   |

Gdy serwer potwierdzi żądanie, wywoływane jest zdarzenie **OnMEXCSubscribed** z nazwą kanału (na przykład **rs.sub.depth.BTC\_USDT**). Błędy są przekazywane przez **OnMEXCError** wraz z komunikatem serwera ułatwiającym diagnostykę.

## Kanały prywatne

Kanały prywatne wymagają ważnego klucza API. Komponent uwierzytelnia się automatycznie po podaniu danych uwierzytelniających.

|                            |                                                         |
|----------------------------|---------------------------------------------------------|
| SubscribePersonalOrder     | Osobiste aktualizacje zleceń.                           |
| SubscribePersonalOrderDeal | Aktualizacje osobistych transakcji (realizacji) zleceń. |
| SubscribePersonalPosition  | Aktualizacje pozycji osobistych.                        |
| SubscribePersonalPlanOrder | Aktualizacje zleceń osobistego planu (wyzwalacza).      |
| SubscribePersonalStopOrder | Aktualizacje osobistych zleceń stop.                    |

|                                |                                                                 |
|--------------------------------|-----------------------------------------------------------------|
| SubscribePersonalStopPlanOrder | Osobiste aktualizacje zleceń stop plan.                         |
| SubscribePersonalRiskLimit     | Aktualizacje osobistego limitu ryzyka.                          |
| SubscribePersonalADLLevel      | Osobiste aktualizacje poziomu ADL (automatyczna delewaryzacja). |
| SubscribePersonalAsset         | Aktualizacje zasobów osobistych.                                |

## Publiczne endpointy REST

Publicznie dostępne metody REST.

|                       |                                                                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| GetPing               | Sprawdza dostępność API za pomocą <b>/api/v1/ping</b> .                                                                                            |
| GetServerTime         | Zwraca znacznik czasu serwera z <b>/api/v1/time</b> .                                                                                              |
| GetContracts          | Dostarcza listę dostępnych kontraktów ( <b>/api/v1/contract/detail</b> ).                                                                          |
| GetDepth              | Pobiera głębokość księgi zleceń z <b>/api/v1/contract/depth</b> ; opcjonalny parametr <b>limit</b> (domyślnie 50) pozwala wybrać liczbę poziomów.  |
| GetDeals              | Pobiera ostatnie transakcje przy użyciu <b>/api/v1/contract/deals</b> z opcjonalnym parametrem <b>limit</b> (domyślnie 100).                       |
| GetKlines             | Zwraca dane świecowe przez <b>/api/v1/contract/kline</b> z opcjonalnymi filtrami <b>startTime</b> , <b>endTime</b> i <b>limit</b> (domyślnie 200). |
| GetIndexPrice         | Pobiera bazową cenę indeksu dla żadanego instrumentu ( <b>/api/v1/contract/indexPrice</b> ).                                                       |
| GetFairPrice          | Zwraca cenę godziwą (mark) dla kontraktu z <b>/api/v1/contract/fairPrice</b> .                                                                     |
| GetFundingRate        | Raportuje ostatnią stopę finansowania dla symbolu, korzystając z <b>/api/v1/contract/fundingRate</b> .                                             |
| GetAllTickers         | Pobierz wszystkie tickery kontraktów futures z aktualnymi podsumowaniami cen w czasie rzeczywistym.                                                |
| GetFundingRateHistory | Pobierz historyczne rekordy stóp finansowania dla kontraktu.                                                                                       |
| GetFairPriceKline     | Pobiera dane świecowe kline dla ceny godziwej (mark price) dla kontraktu.                                                                          |
| GetIndexPriceKline    | Pobierz dane świecowe indeksu cen dla kontraktu.                                                                                                   |

## Prywatne punkty końcowe REST

Prywatne punkty końcowe instrumentów pochodnych wymagają poświadczeń API. Komponent automatycznie podpisuje każde żądanie znacznikiem czasu i wartością recvWindow.

|                     |                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------|
| GetAccountAssets    | Zwraca salda depozytów zabezpieczających z <b>/api/v1/private/account/assets</b> .                            |
| GetPositionList     | Wyświetla bieżące pozycje przez <b>/api/v1/private/position/list</b> , opcjonalnie filtrowane według symbolu. |
| SetPositionLeverage | Aktualizuje dźwignię dla symbolu oraz, jeśli podano, tryb marży.                                              |

|                        |                                                                                                                                                                                                                         |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PlaceOrder             | Składa zlecenie futures na <b>/api/v1/private/order</b> , podając <b>symbol</b> , <b>stronę</b> , <b>positionSide</b> , <b>typ</b> i <b>ilość</b> oraz opcjonalnie <b>cenę</b> , <b>clientId</b> i dodatkowe parametry. |
| CancelOrder            | Anuluje określone zlecenie za pomocą metody <b>/api/v1/private/order/cancel</b> na podstawie parametru <b>orderId</b> lub <b>clientId</b> .                                                                             |
| CancelAllOrders        | Anuluje wszystkie otwarte zlecenia dla symbolu przez <b>/api/v1/private/order/cancel-all</b> .                                                                                                                          |
| GetOpenOrders          | Wyświetla bieżące otwarte zlecenia przez <b>/api/v1/private/order/list/open</b> z opcjonalnym filtrem symbolu.                                                                                                          |
| GetOrderHistory        | Pobiera historię zleceń ( <b>/api/v1/private/order/list/history</b> ) z opcjonalnymi parametrami: symbol, zakres czasowy i limit.                                                                                       |
| GetFundingHistory      | Zwraca historyczne płatności finansowania z <b>/api/v1/private/account/funding</b> z opcjonalnymi filtrami <b>startTime</b> , <b>endTime</b> oraz <b>limit</b> .                                                        |
| GetOpenPositions       | Pobierz wszystkie aktualnie otwarte pozycje.                                                                                                                                                                            |
| ChangeMargin           | Zmień kwotę depozytu zabezpieczającego dla pozycji.                                                                                                                                                                     |
| GetPositionMode        | Pobiera bieżący tryb pozycji (tryb jednokierunkowy lub zabezpieczający).                                                                                                                                                |
| ChangePositionMode     | Zmień tryb pozycji między trybem jednostronnym a trybem zabezpieczenia.                                                                                                                                                 |
| PlaceBatchOrder        | Składa wiele zleceń futures w jednym żądaniu wsadowym.                                                                                                                                                                  |
| GetOrderDetail         | Pobierz szczegółowe informacje o konkretnym zleceniu.                                                                                                                                                                   |
| GetOrderDealDetails    | Pobierz szczegóły realizacji/transakcji dla konkretnego zlecenia.                                                                                                                                                       |
| PlaceTriggerOrder      | Złożenie zlecenia wyzwalającego (plan), które zostanie wykonane po spełnieniu określonych warunków.                                                                                                                     |
| CancelTriggerOrder     | Anuluj określone zlecenie wyzwalające.                                                                                                                                                                                  |
| CancelAllTriggerOrders | Anuluj wszystkie otwarte zlecenia wyzwalające.                                                                                                                                                                          |
| GetTriggerOrders       | Pobierz listę zleceń wyzwalanych (plan).                                                                                                                                                                                |
| GetStopOrders          | Pobierz listę zleceń stop.                                                                                                                                                                                              |
| CancelStopOrder        | Anuluj określone zlecenie stop.                                                                                                                                                                                         |
| CancelAllStopOrders    | Anuluj wszystkie otwarte zlecenia stop.                                                                                                                                                                                 |

# API Bitget

Bitget

## Obsługiwane API

- **WebSocket API:** łączy się z serwerem websocket i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta i inne informacje.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

## Właściwości

Użyj właściwości **BitgetChannel**, aby wybrać typ kanału:

- **bgcSpot:** połącz się ze Spot WebSocket API.
- **bgcFutures:** połącz z API WebSocket Futures.

Następujące właściwości można skonfigurować we właściwości Bitget.

- **ApiKey:** nowy klucz API można wygenerować na koncie Bitget, a jego wartość należy skopiować do tej właściwości. Jeśli ApiKey jest ustawiony, klient połączy się z prywatnym serwerem websocket. Jeśli jest pusty, połączy się z publicznym serwerem WebSocket.
- **ApiSecret:** to tajny klucz API.
- **Passphrase:** niestandardowy ciąg znaków zdefiniowany podczas tworzenia nowego klucza API.

## Połączenie

Gdy klient pomyślnie połączy się z serwerami Bitget, wywoływane jest zdarzenie **OnConnect**. Po wywołaniu zdarzenia **OnConnect** można rozpocząć **wysyłanie i odbieranie wiadomości** do i z serwerów Bitget. Jeśli nawiązywane jest połączenie z prywatnym kanałem websocket, należy poczekać na wywołanie zdarzenia **OnBitgetAuthentication** i sprawdzić, czy parametr success ma wartość true, zanim nastąpi subskrypcja jakiegokolwiek kanału.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWS_API_Bitget *oBitget = new TsgcWS_API_Bitget();
oBitget->Client = oClient;
oBitget->Bitget->ApiKey = "your_api_key";
oBitget->Bitget->ApiSecret = "your_api_secret";
oBitget->Bitget->Passphrase = "your_passphrase";
oBitget->BitgetChannel = bgcSpot;
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#Bitget Connected");
}
```

## Zdarzenia

Klient Bitget implementuje następujące zdarzenia do kontrolowania przepływu połączenia i odbierania danych wysłanych z serwera WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnBitgetAuthentication:** wywoływane, gdy klient uwierzytelnia się na prywatnym serwerze WebSocket.

- **OnBitgetSubscribe:** gdy klient subskrybuje kanał websocket.
- **OnBitgetUnSubscribe:** wywoływane, gdy klient anuluje subskrypcję kanału websocket.
- **OnBitgetData:** wywoływane, gdy klient odbiera dane z serwera.
- **OnBitgetError:** wywoływane w przypadku wystąpienia błędu podczas połączenia WebSocket z serwisem Bitget.
- **OnBitgetHTTPException:** wywoływane w przypadku błędu podczas żądania REST.

## WebSocket API

Kanał WebSocket dostarcza aktualizacje rynku w czasie rzeczywistym dotyczące zleceń i transakcji.

Można subskrybować następujące **kanały**:

|                           |           |                                                                                                          |
|---------------------------|-----------|----------------------------------------------------------------------------------------------------------|
| <b>SubscribeTicker</b>    | Publiczny | Subskrybuj strumień tickera dla instrumentu.                                                             |
| <b>SubscribeTrade</b>     | Publiczny | Subskrybuj strumień ostatnich transakcji.                                                                |
| <b>SubscribeCandle</b>    | Publiczny | Subskrybuje strumień wykresów świecowych. Obsługuje wiele interwałów (np. candle1m, candle5m, candle1H). |
| <b>SubscribeOrderBook</b> | Publiczny | Subskrybuj strumień arkusza zleceń. Obsługuje różne poziomy głębokości.                                  |
| <b>SubscribeOrders</b>    | Prywatny  | Subskrypcja aktualizacji zleceń. Wymaga uwierzytelnienia.                                                |
| <b>SubscribePositions</b> | Prywatny  | Subskrybuj aktualizacje pozycji. Wymaga uwierzytelnienia.                                                |
| <b>SubscribeAccount</b>   | Prywatny  | Subskrybuj aktualizacje salda konta. Wymagane uwierzytelnianie.                                          |

Poniżej znajduje się przykład subskrybowania prywatnych kanałów WebSocket po pomyślnym uwierzytelnieniu.

```
void OnBitgetAuthentication(TObject *Sender, bool aSuccess, const string aError, const string aRawMessage)
{
    if (aSuccess == true)
    {
        oBitget->SubscribeOrders("BTCUSDT");
        oBitget->SubscribeAccount();
    }
}
```

## REST API

REST API udostępnia publiczne i prywatne metody do żądania danych z rynków i prywatnych kont. Dostęp do REST API uzyskuje się przez właściwość **REST\_API** komponentu.

|                        |           |                                                                   |
|------------------------|-----------|-------------------------------------------------------------------|
| <b>GetServerTime</b>   | Publiczny | Pobierz czas serwera.                                             |
| <b>GetTickers</b>      | Publiczny | Pobiera informacje o tickerze dla jednego lub wszystkich symboli. |
| <b>GetOrderBook</b>    | Publiczny | Pobierz arkusz zleceń dla symbolu.                                |
| <b>GetCandles</b>      | Publiczny | Pobierz dane świec/kline dla symbolu.                             |
| <b>GetRecentTrades</b> | Publiczny | Pobierz najnowsze transakcje dla danego symbolu.                  |
| <b>PlaceOrder</b>      | Prywatny  | Złóż nowe zamówienie.                                             |
| <b>CancelOrder</b>     | Prywatny  | Anuluje istniejące zlecenie.                                      |
| <b>GetOpenOrders</b>   | Prywatny  | Pobiera listę otwartych zleceń.                                   |
| <b>GetOrderDetail</b>  | Prywatny  | Pobierz szczegóły określonego zlecenia.                           |
| <b>GetOrderHistory</b> | Prywatny  | Pobierz historię zleceń.                                          |

|                         |          |                                      |
|-------------------------|----------|--------------------------------------|
| <b>GetAccountAssets</b> | Prywatny | Pobierz informacje o aktywach konta. |
| <b>GetAccountInfo</b>   | Prywatny | Pobierz informacje o koncie.         |

Poniżej znajduje się przykład użycia REST API.

```
TsgcWS_API_Bitget *oBitget = new TsgcWS_API_Bitget();
oBitget->Bitget->ApiKey = "your_api_key";
oBitget->Bitget->ApiSecret = "your_api_secret";
oBitget->Bitget->Passphrase = "your_passphrase";
// Get tickers
oBitget->REST_API->GetTickers("SPOT");
// Place an order
oBitget->REST_API->PlaceOrder("BTCUSDT", "buy", "market", "normal", "0.01");
```

# API GateIO

[Gate.io](https://gate.io)

## Obsługiwane API

- **WebSocket API:** łączy się z serwerem websocket i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta i inne informacje.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

## Właściwości

Użyj właściwości **GateIOChannel**, aby wybrać typ kanału:

- **giocSpot:** połącz z WebSocket API Spot.
- **giocFutures:** nawiązuje połączenie z WebSocket API Futures.

Następujące właściwości można skonfigurować we właściwości GateIO.

- **ApiKey:** Nowy klucz API można wygenerować na koncie Gate.io, a następnie skopiować jego wartość do tej właściwości. Jeśli właściwość ApiKey jest ustawiona, klient uwierzytliczy się na prywatnym serwerze. Jeśli jest pusta, dostępne będą wyłącznie kanały publiczne.
- **ApiSecret:** to tajny klucz API.

## Połączenie

Gdy klient pomyślnie połączy się z serwerami Gate.io, wywoływane jest zdarzenie **OnConnect**. Po wywołaniu zdarzenia **OnConnect** można rozpocząć **wysyłanie i odbieranie wiadomości** do i z serwerów Gate.io. Jeśli nawiązywane jest połączenie z prywatnym kanałem websocket, należy poczekać na wywołanie zdarzenia **OnGateIOAuthentication** i sprawdzić, czy parametr success ma wartość true, zanim nastąpi subskrypcja jakiegokolwiek prywatnego kanału.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWS_API_GateIO *oGateIO = new TsgcWS_API_GateIO();
oGateIO->Client = oClient;
oGateIO->GateIO->ApiKey = "your_api_key";
oGateIO->GateIO->ApiSecret = "your_api_secret";
oGateIO->GateIOChannel = giocSpot;
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#GateIO Connected");
}
```

## Zdarzenia

Klient GateIO implementuje następujące zdarzenia do kontrolowania przepływu połączeń i pobierania danych wysyłanych przez serwer WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnGateIOAuthentication:** wywoływane, gdy klient uwierzytliczy się na prywatnym serwerze WebSocket.
- **OnGateIOSubscribe:** gdy klient subskrybuje kanał websocket.
- **OnGateIOUnsubscribe:** wywoływane, gdy klient anuluje subskrypcję kanału WebSocket.

- **OnGateIOData:** wywoływane, gdy klient odbiera dane z serwera.
- **OnGateIOError:** wywoływane w przypadku wystąpienia błędu podczas połączenia WebSocket z GateIO.
- **OnGateIOHTTPException:** wywoływane w przypadku błędu podczas żądania REST.

## WebSocket API

Kanał WebSocket dostarcza aktualizacje rynku w czasie rzeczywistym dotyczące zleceń i transakcji.

Można subskrybować następujące **kanały**:

|                              |           |                                                                                                             |
|------------------------------|-----------|-------------------------------------------------------------------------------------------------------------|
| <b>SubscribeTicker</b>       | Publiczny | Subskrybuje strumień tickera dla symbolu.                                                                   |
| <b>SubscribeTrades</b>       | Publiczny | Subskrybuj strumień ostatnich transakcji.                                                                   |
| <b>SubscribeCandlesticks</b> | Publiczny | Subskrybuj strumień świec. Obsługuje wiele interwałów (np. 1m, 5m, 15m, 30m, 1h).                           |
| <b>SubscribeOrderBook</b>    | Publiczny | Subskrybuj strumień arkusza zamówień. Obsługuje konfigurowalne poziomy głąbokości i interwały aktualizacji. |
| <b>SubscribeOrders</b>       | Prywatny  | Subskrypcja aktualizacji zleceń. Wymaga uwierzytelnienia.                                                   |
| <b>SubscribeBalances</b>     | Prywatny  | Subskrypcja aktualizacji salda. Wymaga uwierzytelnienia.                                                    |

Poniżej znajduje się przykład subskrybowania publicznych kanałów WebSocket.

```
void OnConnect(TsgcWSConnection *Connection)
{
    oGateIO->SubscribeTicker("BTC_USDT");
    oGateIO->SubscribeOrderBook("BTC_USDT", "20", "100ms");
}
```

## REST API

REST API udostępnia publiczne i prywatne metody do żądania danych z rynków i prywatnych kont. Dostęp do REST API uzyskuje się przez właściwość **REST\_API** komponentu.

|                         |           |                                                                         |
|-------------------------|-----------|-------------------------------------------------------------------------|
| <b>GetCurrencyPairs</b> | Publiczny | Pobierz informacje o dostępnych parach walutowych.                      |
| <b>GetTickers</b>       | Publiczny | Pobierz informacje o tickerze dla jednej lub wszystkich par walutowych. |
| <b>GetOrderBook</b>     | Publiczny | Pobierz księgę zleceń dla pary walutowej.                               |
| <b>GetCandlesticks</b>  | Publiczny | Pobieranie danych świecowych/kline dla pary walutowej.                  |
| <b>GetTrades</b>        | Publiczny | Pobierz ostatnie transakcje dla pary walutowej.                         |
| <b>PlaceOrder</b>       | Prywatny  | Składa nowe zlecenie. Obsługuje zlecenia limit i market.                |
| <b>CancelOrder</b>      | Prywatny  | Anuluje istniejące zlecenie.                                            |
| <b>GetOrder</b>         | Prywatny  | Pobierz szczegóły określonego zlecenia.                                 |
| <b>GetOpenOrders</b>    | Prywatny  | Pobiera listę otwartych zleceń.                                         |
| <b>GetSpotAccounts</b>  | Prywatny  | Pobierz informacje o koncie spot.                                       |

Poniżej znajduje się przykład użycia REST API.

```
TsgcWS_API_GateIO *oGateIO = new TsgcWS_API_GateIO();
oGateIO->GateIO->ApiKey = "your_api_key";
oGateIO->GateIO->ApiSecret = "your_api_secret";
```



```
// Get tickers
oGateIO->REST_API->GetTickers("BTC_USDT");
// Place an order
oGateIO->REST_API->PlaceOrder("BTC_USDT", "buy", "0.001", "50000", "limit");
```

# API Deribit

## Deribit

Deribit to giełda instrumentów pochodnych na kryptowaluty, oferująca handel kontraktami futures i opcjami na Bitcoin i Ethereum.

## Obsługiwane API

- **WebSocket API:** łączy się z serwerem websocket i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta i inne informacje.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

## Właściwości

Poniższe właściwości można skonfigurować we właściwości Deribit.

- **ApiKey:** nowy klucz API można wygenerować na koncie Deribit; wystarczy skopiować wartość do tej właściwości. Jeśli ApiKey jest ustawiony, klient uwierzytłni się wobec prywatnego serwera. Jeśli jest pusty, dostępne będą wyłącznie kanały publiczne.
- **ApiSecret:** to tajny klucz API.
- **TestNet:** po włączeniu klient łączy się ze środowiskiem testowym Deribit TestNet (domyślnie wyłączone). Przydatne do testowania bez ryzyka utraty rzeczywistych środków.

## Połączenie

Po pomyślnym połączeniu klienta z serwerami Deribit wywoływane jest zdarzenie **OnConnect**. Po wywołaniu zdarzenia **OnConnect** można rozpocząć **wysyłanie i odbieranie wiadomości** do/od serwerów Deribit. W przypadku połączenia z prywatnym kanałem WebSocket należy poczekać na wywołanie zdarzenia **OnDeribitAuthentication** i sprawdzić, czy parametr success ma wartość true, przed subskrypcją dowolnego prywatnego kanału.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWS_API_Deribit *oDeribit = new TsgcWS_API_Deribit();
oDeribit->Client = oClient;
oDeribit->Deribit->ApiKey = "your_api_key";
oDeribit->Deribit->ApiSecret = "your_api_secret";
oDeribit->Deribit->TestNet = false;
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#Deribit Connected");
}
```

## Zdarzenia

Klient Deribit implementuje następujące zdarzenia w celu kontroli przepływu połączeń i odbioru danych wysyłanych przez serwer WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnDeribitAuthentication:** wywoływane, gdy klient uwierzytłnia się na prywatnym serwerze WebSocket.
- **OnDeribitSubscribe:** wywoływane, gdy klient subskrybuje kanał websocket.
- **OnDeribitUnSubscribe:** wywoływane, gdy klient anuluje subskrypcję kanału WebSocket.

- **OnDeribitData:** wywoływane, gdy klient odbiera dane z serwera.
- **OnDeribitError:** wywoływane w przypadku wystąpienia błędu podczas połączenia WebSocket z Deribit.
- **OnDeribitHTTPException:** gdy wystąpi błąd podczas żądania REST.

## WebSocket API

Kanał WebSocket dostarcza aktualizacje rynku w czasie rzeczywistym dotyczące zleceń i transakcji.

Możliwa jest subskrypcja następujących **kanałów publicznych**:

|                            |                                                                                                                                                |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SubscribeTicker</b>     | Subskrybuj strumień tickera dla instrumentu. Dostarcza dane o cenie i wolumenie w czasie rzeczywistym.                                         |
| <b>SubscribeTrades</b>     | Subskrybuj strumień ostatnich transakcji dla instrumentu.                                                                                      |
| <b>SubscribeOrderBook</b>  | Subskrybuje strumień książki zleceń. Obsługuje konfigurowalne grupowanie, poziomy głębokości i interwały aktualizacji.                         |
| <b>SubscribeCandle</b>     | Subskrypcja strumienia wykresów świecowych. Obsługuje wiele rozdzielczości (np. 1, 3, 5, 10, 15, 30, 60, 120, 180, 360, 720, 1D).              |
| <b>SubscribePerpetual</b>  | Subskrybuje dane kontraktu wieczystego dla instrumentu. Dostarcza stopę finansowania i inne informacje specyficzne dla kontraktów wieczystych. |
| <b>SubscribeBookChange</b> | Subskrybuje powiadomienia o zmianach w księdze zleceń dla instrumentu.                                                                         |

Można subskrybować następujące **kanały prywatne** (wymagają uwierzytelnienia):

|                                |                                                                                             |
|--------------------------------|---------------------------------------------------------------------------------------------|
| <b>SubscribeOrders</b>         | Subskrybuj aktualizacje zleceń dla instrumentu.                                             |
| <b>SubscribePositions</b>      | Subskrybuj aktualizacje pozycji. Obsługuje filtrowanie według waluty i rodzaju instrumentu. |
| <b>SubscribeAccountChanges</b> | Subskrybuj zmiany salda konta dla waluty.                                                   |
| <b>SubscribeUserTrades</b>     | Subskrybuje aktualizacje transakcji użytkownika dla instrumentu.                            |

Poniżej znajduje się przykład subskrypcji kanałów WebSocket.

```
void OnDeribitAuthentication(TObject *Sender, bool aSuccess, const string aError, const string aRawMessage)
{
    if (aSuccess == true)
    {
        oDeribit->SubscribeOrders("BTC-PERPETUAL");
        oDeribit->SubscribePositions("BTC");
    }
}
```

## REST API

REST API udostępnia publiczne i prywatne metody do pobierania danych z rynków, handlu, konta i portfela. Dostęp do REST API uzyskuje się przez właściwość **REST\_API** komponentu.

### Dane rynkowe (publiczne)

|                       |                                                      |
|-----------------------|------------------------------------------------------|
| <b>GetInstruments</b> | Pobierz dostępne instrumenty dla danej waluty.       |
| <b>GetTicker</b>      | Pobierz informacje o instrumencie z arkusza notowań. |
| <b>GetOrderBook</b>   | Pobierz arkusz zleceń dla instrumentu.               |
| <b>GetTrades</b>      | Pobiera ostatnie transakcje dla instrumentu.         |

|                                   |                                                                              |
|-----------------------------------|------------------------------------------------------------------------------|
| <b>GetCurrencies</b>              | Pobierz listę obsługiwanych walut.                                           |
| <b>GetIndexPrice</b>              | Pobierz bieżącą cenę indeksu dla podanej nazwy indeksu.                      |
| <b>GetFundingRateHistory</b>      | Pobierz historię stopy finansowania dla instrumentu.                         |
| <b>GetFundingRateValue</b>        | Pobierz bieżącą wartość stopy finansowania dla instrumentu.                  |
| <b>GetBookSummaryByCurrency</b>   | Pobiera podsumowanie księgi zleceń dla wszystkich instrumentów danej waluty. |
| <b>GetBookSummaryByInstrument</b> | Pobierz podsumowanie arkusza dla określonego instrumentu.                    |

## Trading (prywatny)

|                              |                                                                         |
|------------------------------|-------------------------------------------------------------------------|
| <b>Kup</b>                   | Złóż zlecenie kupna. Obsługuje zlecenia rynkowe i limitowane.           |
| <b>Sprzedaj</b>              | Składa zlecenie sprzedaży. Obsługuje zlecenia rynkowe i z limitem ceny. |
| <b>CancelOrder</b>           | Anuluj istniejące zlecenie według identyfikatora zlecenia.              |
| <b>CancelAllOrders</b>       | Anuluj wszystkie otwarte zlecenia.                                      |
| <b>CancelAllByInstrument</b> | Anuluj wszystkie otwarte zlecenia dla określonego instrumentu.          |
| <b>EditOrder</b>             | Edytuje istniejące zlecenie (zmiana kwoty lub ceny).                    |
| <b>GetOpenOrders</b>         | Pobierz listę otwartych zleceń dla waluty.                              |
| <b>GetOrderState</b>         | Pobiera stan konkretnego zlecenia.                                      |
| <b>GetOrderHistory</b>       | Pobierz historię zleceń dla waluty.                                     |

## Konto (prywatne)

|                          |                                              |
|--------------------------|----------------------------------------------|
| <b>GetPositions</b>      | Pobierz pozycje dla danej waluty.            |
| <b>GetAccountSummary</b> | Pobierz podsumowanie konta dla danej waluty. |
| <b>GetSubAccounts</b>    | Pobierz listę subkont.                       |
| <b>GetTransactionLog</b> | Pobiera dziennik transakcji dla waluty.      |

## Portfel (prywatny)

|                       |                                               |
|-----------------------|-----------------------------------------------|
| <b>GetDeposits</b>    | Pobierz historię wpłat dla danej waluty.      |
| <b>GetWithdrawals</b> | Pobierz historię wypłat dla danej waluty.     |
| <b>GetTransfers</b>   | Pobierz historię transferów dla danej waluty. |

Poniżej znajduje się przykład użycia REST API.

```
TsgcWS_API_Deribit *oDeribit = new TsgcWS_API_Deribit();
oDeribit->Deribit->ApiKey = "your_api_key";
oDeribit->Deribit->ApiSecret = "your_api_secret";
// Get ticker
oDeribit->REST_API->GetTicker("BTC-PERPETUAL");
// Place a buy order
oDeribit->REST_API->Buy("BTC-PERPETUAL", 10, "market");
// Get account summary
oDeribit->REST_API->GetAccountSummary("BTC");
```

# API Crypto.com

[Crypto.com](https://crypto.com)

## Obsługiwane API

- **WebSocket API:** łączy się z serwerem websocket i zapewnia aktualizacje danych rynkowych w czasie rzeczywistym, zmiany konta i inne informacje.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

## Właściwości

Użyj właściwości **Channel**, aby wybrać typ kanału:

- **cccMarket:** połączenie z Market WebSocket API w celu uzyskania publicznych danych rynkowych.
- **cccUser:** połącz się z WebSocket API użytkownika w celu uzyskania prywatnych danych konta (wymagane uwierzytelnienie).

Poniżej przedstawiono właściwości, które można skonfigurować we właściwości CryptoCom.

- **ApiKey:** nowy klucz API można uzyskać na koncie Crypto.com; należy skopiować jego wartość do tej właściwości. Wymagany dla kanałów prywatnych i uwierzytelnionych wywołań REST API.
- **ApiSecret:** to tajny klucz API.

## Połączenie

Po pomyślnym połączeniu klienta z serwerami Crypto.com wywoływane jest zdarzenie **OnConnect**. Po wywołaniu zdarzenia **OnConnect** można zacząć **wysyłać i odbierać wiadomości** do/z serwerów Crypto.com. Jeśli następuje połączenie z kanałem użytkownika, należy poczekać na wywołanie zdarzenia **OnCryptoComAuthentication** i sprawdzić, czy parametr sukcesu wynosi True, przed subskrybowaniem jakiegokolwiek prywatnego kanału.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWS_API_CryptoCom *oCryptoCom = new TsgcWS_API_CryptoCom();
oCryptoCom->Client = oClient;
oCryptoCom->CryptoCom->ApiKey = "your_api_key";
oCryptoCom->CryptoCom->ApiSecret = "your_api_secret";
oCryptoCom->Channel = cccMarket;
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#CryptoCom Connected");
}
```

## Zdarzenia

Klient Crypto.com implementuje następujące zdarzenia do sterowania przepływem połączenia i odbierania danych wysyłanych przez serwer WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnCryptoComAuthentication:** wywoływane, gdy klient uwierzytelnia się na prywatnym serwerze WebSocket.
- **OnCryptoComSubscribe:** wywoływane, gdy klient subskrybuje kanał websocket.

- **OnCryptoComUnSubscribe:** wywoływane, gdy klient anuluje subskrypcję kanału WebSocket.
- **OnCryptoComData:** wywoływane, gdy klient odbiera dane z serwera.
- **OnCryptoComError:** wywoływane w przypadku wystąpienia błędu podczas połączenia WebSocket z Crypto.com.
- **OnCryptoComHTTPException:** wywoływane w przypadku błędu podczas żądania REST.

## WebSocket API

Kanał WebSocket dostarcza aktualizacje rynku w czasie rzeczywistym dotyczące zleceń i transakcji.

Możesz subskrybować następujące **kanały rynkowe** (publiczne):

|                             |                                                                                                        |
|-----------------------------|--------------------------------------------------------------------------------------------------------|
| <b>SubscribeTicker</b>      | Subskrybuj strumień tickera dla instrumentu. Dostarcza dane o cenie i wolumenie w czasie rzeczywistym. |
| <b>SubscribeTrade</b>       | Subskrybuj strumień ostatnich transakcji dla instrumentu.                                              |
| <b>SubscribeCandlestick</b> | Subskrybuj strumień świec. Obsługiwane są różne interwały (np. 5m, 15m, 30m, 1h, 4h, 1D).              |
| <b>SubscribeBook</b>        | Subskrybuje strumień arkusza zleceń. Obsługuje konfigurowalne poziomy głębokości (np. 10, 50).         |

Można subskrybować następujące **kanały użytkownika** (prywatne, wymagają uwierzytelnienia):

|                            |                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------|
| <b>SubscribeOrders</b>     | Subskrybuje aktualizacje transakcji użytkownika dla instrumentu.                              |
| <b>SubscribeUserTrades</b> | Subskrybuj aktualizacje transakcji użytkownika. Opcjonalnie filtruj według nazwy instrumentu. |
| <b>SubscribeBalance</b>    | Subskrybuj aktualizacje salda dla swojego konta.                                              |

Poniżej znajduje się przykład subskrypcji kanałów rynkowych.

```
void OnConnect(TsgcWSConnection *Connection)
{
    oCryptoCom->SubscribeTicker("BTC_USDT");
    oCryptoCom->SubscribeBook("BTC_USDT", "10");
}
```

## REST API

REST API udostępnia publiczne i prywatne metody do żądania danych z rynków i prywatnych kont. Dostęp do REST API uzyskuje się przez właściwość **REST\_API** komponentu.

|                        |           |                                                                        |
|------------------------|-----------|------------------------------------------------------------------------|
| <b>GetInstruments</b>  | Publiczny | Pobierz listę dostępnych instrumentów.                                 |
| <b>GetTicker</b>       | Publiczny | Pobierz informacje o tickerze dla jednego lub wszystkich instrumentów. |
| <b>GetOrderBook</b>    | Publiczny | Pobierz arkusz zleceń dla instrumentu.                                 |
| <b>GetTrades</b>       | Publiczny | Pobiera ostatnie transakcje dla instrumentu.                           |
| <b>GetCandlestick</b>  | Publiczny | Pobierz dane świec/kline dla instrumentu.                              |
| <b>CreateOrder</b>     | Prywatny  | Tworzy nowe zlecenie. Obsługuje zlecenia z limitem ceny i rynkowe.     |
| <b>CancelOrder</b>     | Prywatny  | Anuluje istniejące zlecenie.                                           |
| <b>CancelAllOrders</b> | Prywatny  | Anuluj wszystkie otwarte zlecenia dla instrumentu.                     |
| <b>GetOpenOrders</b>   | Prywatny  | Pobiera listę otwartych zleceń.                                        |
| <b>GetOrderDetail</b>  | Prywatny  | Pobierz szczegóły określonego zlecenia.                                |

|                          |          |                                           |
|--------------------------|----------|-------------------------------------------|
| <b>GetOrderHistory</b>   | Prywatny | Pobierz historię zleceń.                  |
| <b>GetAccountSummary</b> | Prywatny | Pobierz podsumowanie informacji o koncie. |

Poniżej znajduje się przykład użycia REST API.

```
TsgcWS_API_CryptoCom *oCryptoCom = new TsgcWS_API_CryptoCom();
oCryptoCom->CryptoCom->ApiKey = "your_api_key";
oCryptoCom->CryptoCom->ApiSecret = "your_api_secret";
// Get instruments
oCryptoCom->REST_API->GetInstruments();
// Create an order
oCryptoCom->REST_API->CreateOrder("BTC_USDT", "BUY", "MARKET", "", "0.001");
// Get account summary
oCryptoCom->REST_API->GetAccountSummary();
```

# API HTX

## HTX

HTX (dawniej Huobi) to międzynarodowa wielojęzyczna giełda kryptowalut. Komponent HTX API udostępnia HTTP REST API uzupełniające istniejące Huobi WebSocket API.

## Obsługiwane API

- **WebSocket API:** łączenie z serwerem WebSocket i udostępnianie aktualizacji danych rynkowych w czasie rzeczywistym, zmian na koncie i nie tylko. Zapoznaj się z dokumentacją Huobi API, aby uzyskać informacje o metodach subskrypcji WebSocket.
- **REST API:** wysyłanie żądań HTTP w celu pobierania danych rynkowych, składania zleceń, danych konta...

## Właściwości

Można skonfigurować następujące właściwości we właściwości Huobi (dla WebSocket) lub HTXOptions (dla REST API).

- **ApiKey:** nowy klucz API można zamówić na koncie HTX — wystarczy skopiować wartość do tej właściwości. Jeśli ApiKey jest ustawiony, klient połączy się z prywatnym serwerem WebSocket. Jeśli jest pusty, połączy się z publicznym serwerem WebSocket.
- **ApiSecret:** to tajny klucz API.

## Połączenie

Składnik HTX rozszerza klienta Huobi WebSocket API. Gdy klient pomyślnie połączy się z serwerami HTX, wywoływane jest zdarzenie **OnConnect**. Jeśli ApiKey nie jest pusty, klient spróbuje połączyć się z prywatnym serwerem WebSocket i dostępne będą wyłącznie metody prywatne. Jeśli ApiKey jest pusty, klient połączy się z publicznym serwerem WebSocket i dostępne będą wyłącznie metody publiczne.

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWS_API_Huobi *oHTX = new TsgcWS_API_Huobi();
oHTX->Client = oClient;
oHTX->Huobi->ApiKey = "your_api_key";
oHTX->Huobi->ApiSecret = "your_api_secret";
oClient->Active = true;
void OnConnect(TsgcWSConnection *Connection)
{
    DoLog("#HTX Connected");
}
```

## Zdarzenia

Klient HTX implementuje następujące zdarzenia w celu kontroli przepływu połączeń i odbioru danych wysyłanych przez serwer WebSocket:

- **OnConnect:** wywoływane, gdy klient websocket łączy się z serwerem WebSocket.
- **OnDisconnect:** wywoływane, gdy klient WebSocket rozłącza się z serwerem WebSocket.
- **OnHuobiSubscribed:** zdarzenie wywoływane po pomyślnej subskrypcji.
- **OnHuobiUnsubscribed:** zdarzenie wywoływane po pomyślnym anulowaniu subskrypcji.
- **OnHuobiUpdate:** zdarzenie jest wywoływane za każdym razem, gdy nastąpi aktualizacja danych (kline, głębokość rynku...).
- **OnHuobiError:** zdarzenie wywoływane, gdy wystąpi błąd w HTX API. Dostarcza informacji o błędzie.



## WebSocket API

WebSocket API zapewnia aktualizacje danych rynkowych w czasie rzeczywistym. Pełna lista metod subskrypcji WebSocket, w tym kanałów publicznych (SubscribeKLine, SubscribeMarketDepth, SubscribeTradeDetail itp.) i prywatnych (SubscribeOrderUpdates, SubscribeTradeClearing, SubscribeAccountChange), znajduje się w dokumentacji Huobi API.

## REST API

REST API zapewnia publiczne i prywatne metody żądania danych z rynków i prywatnych kont. Dostęp do REST API odbywa się przez komponent **TsgcHTTP\_API\_HTX**.

### Dane rynkowe (publiczne)

|                              |                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>GetServerTime</b>         | Pobierz czas serwera.                                                                                                  |
| <b>GetSymbols</b>            | Pobierz listę dostępnych symboli handlowych.                                                                           |
| <b>GetCurrencys</b>          | Pobierz listę obsługiwanych walut.                                                                                     |
| <b>GetMarketTickers</b>      | Pobierz tickery dla wszystkich symboli handlowych.                                                                     |
| <b>GetMarketDetail</b>       | Pobierz 24-godzinne szczegóły rynkowe dla symbolu.                                                                     |
| <b>GetMarketDepth</b>        | Pobierz głębokość arkusza zleceń dla symbolu. Obsługuje konfigurowalny typ głębokości i poziomy głębokości.            |
| <b>GetMarketTrade</b>        | Pobierz najnowszą transakcję dla danego symbolu.                                                                       |
| <b>GetMarketHistoryTrade</b> | Pobierz najnowszą historię transakcji dla danego symbolu.                                                              |
| <b>GetMarketHistoryKline</b> | Pobierz dane świec/klines. Obsługuje wiele okresów (1min, 5min, 15min, 30min, 60min, 4hour, 1day, 1mon, 1week, 1year). |

### Trading (prywatny)

|                        |                                                                                                                  |
|------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>PlaceOrder</b>      | Składa nowe zlecenie. Obsługuje typy: kupno-rynkowe, sprzedaż-rynkowa, kupno-limitowe, sprzedaż-limitowa i inne. |
| <b>CancelOrder</b>     | Anuluj istniejące zlecenie według identyfikatora zlecenia.                                                       |
| <b>GetOrder</b>        | Pobierz szczegóły określonego zlecenia.                                                                          |
| <b>GetOpenOrders</b>   | Pobierz listę otwartych zleceń dla konta.                                                                        |
| <b>GetOrderHistory</b> | Pobierz historię zleceń dla symbolu.                                                                             |

### Konto (prywatne)

|                                 |                                                                    |
|---------------------------------|--------------------------------------------------------------------|
| <b>GetAccounts</b>              | Pobiera listę kont.                                                |
| <b>GetAccountBalance</b>        | Pobiera saldo określonego konta.                                   |
| <b>GetAccountAssetValuation</b> | Pobiera całkowitą wycenę aktywów dla danego typu konta (np. spot). |

Poniżej znajduje się przykład użycia REST API.

```
TsgcHTTP_API_HTX *oHTX = new TsgcHTTP_API_HTX();
oHTX->HTXOptions->ApiKey = "your_api_key";
oHTX->HTXOptions->ApiSecret = "your_api_secret";
// Get market tickers
oHTX->GetMarketTickers();
// Get kline data
oHTX->GetMarketHistoryKline("btcusdt", "1min", 150);
```

```
// Place an order
oHTX->PlaceOrder("12345678", "btcusdt", "buy-limit", "0.001", "50000");
// Get account balance
oHTX->GetAccountBalance("12345678");
```

# WhatsApp Cloud API

## Whatsapp

**Wysyłaj i odbieraj wiadomości** za pomocą hostowanej w chmurze wersji **platformy WhatsApp Business. Cloud API** umożliwia implementację interfejsów API WhatsApp Business bez konieczności hostowania własnych serwerów, a także ułatwia skalowanie komunikacji biznesowej. Cloud API obsługuje do 80 wiadomości na sekundę w trybie łączonym wysyłania i odbierania (łącznie wiadomości tekstowych i multimedialnych).

WhatsApp Business API umożliwia średnim i dużym firmom komunikację z klientami na dużą skalę. Korzystając z API, firmy mogą budować systemy łączące tysiące klientów z agentami lub botami, co umożliwia zarówno programową, jak i ręczną komunikację. Ponadto API można integrować z wieloma systemami zaplecza, takimi jak platformy CRM i marketingowe.

## Funkcje

Firmy będą szybciej otrzymywać wszystkie nowe funkcje w Cloud API. Obecnie WhatsApp Business Cloud API zawiera wszystkie funkcje dostępne w WhatsApp Business API.

Przydatne funkcje WhatsApp Cloud API:

- **Integracja** wiadomości WhatsApp z narzędziami takimi jak **CRM, analityka** i aplikacje **innych firm**
- **Zielony znacznik**, zweryfikowany profil WhatsApp Business
- WhatsApp **rozsyłanie & masowe wysyłanie wiadomości**
- Brak aplikacji ani interfejsu; dostęp przez BSP lub CRM
- **WhatsApp Chatbot** i **automatyzacja** czatu przy użyciu aplikacji innych firm
- **Planowanie** wiadomości WhatsApp na dużą skalę
- **Interaktywne przesyłanie wiadomości** obejmuje wiadomości z listą, przyciski odpowiedzi, wiadomości CTA

## Najczęstsze zastosowania

- **Konfiguracja**
  - [WhatsApp Create App](#)
  - [WhatsApp Phone Number Id](#)
  - [WhatsApp Token](#)
  - [WhatsApp Webhook](#)
  - [WhatsApp Security](#)

- **Wiadomości**
  - [Wysyłanie wiadomości WhatsApp](#)
  - [Wysyłanie interaktywnych wiadomości WhatsApp](#)
  - [Wysyłanie wiadomości z szablonu WhatsApp](#)
  - [Odbieranie wiadomości i powiadomień o statusie WhatsApp](#)
  - [Wysyłanie plików WhatsApp](#)
  - [Pobieranie mediów WhatsApp](#)

## Pierwsze kroki

Aby wysłać i odebrać pierwszą wiadomość przy użyciu numeru testowego, wykonaj następujące kroki:

### 1. Skonfiguruj zasoby deweloperskie i dostęp do platformy

- [Zarejestruj się jako deweloper Meta](#)
- [Włącz uwierzytelnianie dwuskładnikowe dla swojego konta](#)
- [Utwórz aplikację Meta](#): Przejdź do **developers.facebook.com** > **Moje aplikacje** > **Utwórz aplikację**. Wybierz typ „Biznes” i postępuj zgodnie z instrukcjami wyświetlanymi na ekranie.

Z poziomu pulpitu aplikacji należy kliknąć aplikację, którą chcesz połączyć z WhatsApp. Przewiń w dół, aby znaleźć produkt „WhatsApp” i kliknij **Konfiguruj**.

Następnie zostanie wyświetlona opcja wyboru istniejącego Business Managera (jeśli użytkownik go posiada) lub — opcjonalnie — automatycznego utworzenia nowego podczas procesu wdrażania (konfigurację firmy można dostosować później). Należy dokonać wyboru i kliknąć **Kontynuuj**.

Po kliknięciu przycisku **Continue** proces wdrożenia wykonuje następujące działania:

- Aplikacja jest powiązana z wybranym menedżerem Business Manager lub tym, który został utworzony automatycznie.
- Testowy numer telefonu WhatsApp jest dodawany do Twojej firmy. Można go używać do eksplorowania platformy WhatsApp Business Platform bez konieczności rejestracji ani migracji rzeczywistego numeru telefonu. Testowe numery telefonu mogą wysyłać nieograniczoną liczbę wiadomości do maksymalnie 5 odbiorców (w dowolnym miejscu na świecie).

### 2. Wyślij wiadomość testową

Teraz można otworzyć środowisko IDE i utworzyć nowy projekt. Należy upuścić komponent `TsgcWhatsapp_Client` i wypełnić następujące właściwości:

- **WhatsappOptions.PhoneNumberId:** identyfikator numeru telefonu używanego do wysyłania wiadomości.
- **WhatsappOptions.Token:** jest tymczasowym tokenem dostępu ważnym przez 24 godziny.

Po prawidłowym skonfigurowaniu obu właściwości należy wywołać metodę **SendTest**, aby wysłać **pierwszą wiadomość** na numer telefonu za pośrednictwem **platformy WhatsApp Business**.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendTest("34605889421");
```

### 3. Skonfiguruj webhook

Aby otrzymywać powiadomienia o odebraniu wiadomości lub zmianie jej statusu, należy skonfigurować punkt końcowy Webhooks dla aplikacji. Konfiguracja Webhooks nie wpływa na status numeru telefonu i nie zakłóca wysyłania ani odbierania wiadomości.

Aby rozpocząć, należy najpierw utworzyć punkt końcowy, konfigurując właściwość **ServerOptions** komponentu klienta WhatsApp i ustawiając następujące właściwości:

- **ServerOptions:** tutaj można skonfigurować adres IP do powiązania, port nasłuchiwania, czy używany jest SSL (WebHook musi działać na bezpiecznym serwerze; można skonfigurować serwer SSL lub przekierować żądania WebHook do serwera bez HTTPS). Serwer oparty jest na [TsgcWebSocketHTTPServer](#).
  - **WebhookOptions:** ta właściwość umożliwia ustawienie właściwości Webhook, które zostaną później skonfigurowane na koncie dewelopera Facebook.
    - **Endpoint:** nazwa punktu końcowego, domyślnie /webhook. Przykład: jeśli serwer nasłuchuje pod adresem `https://www.esegece.com`, punkt końcowy to `https://www.esegece.com/webhook`
    - **Token:** ciąg zabezpieczający, który może zawierać dowolną wartość zdefiniowaną przez użytkownika. Służy do weryfikacji poprawności rejestracji Webhook.

Po skonfigurowaniu serwera można użyć metody **StartServer**, aby uruchomić serwer i zacząć akceptować przychodzące żądania.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->ServerOptions->WebhookOptions->PhoneNumberId = "/webhook";
oClient->ServerOptions->WebhookOptions->Token = "MySecretToken";
oClient->StartServer();
```

Po przygotowaniu punktu końcowego należy przejść do panelu aplikacji.

Na pulpicie nawigacyjnym aplikacji znajdź produkt WhatsApp i kliknij **Konfiguracja**. Następnie znajdź sekcję webhooków i kliknij **Skonfiguruj webhook**. Po kliknięciu na ekranie pojawi się okno dialogowe, w którym należy podać dwie informacje:

- Adres URL wywołania zwrotnego: to adres URL, pod który Meta będzie wysyłać zdarzenia.
- Verify Token: Ten ciąg jest ustawiany przez użytkownika podczas tworzenia punktu końcowego webhooka.

Po dodaniu informacji należy kliknąć **Verify and Save**.

Po powrocie do Panelu aplikacji kliknij **WhatsApp > Konfiguracja** w lewym panelu. W sekcji Webhooks kliknij **Zarządzaj**. Otworzy się okno dialogowe ze wszystkimi obiektami, o których można otrzymywać powiadomienia. Aby odbierać wiadomości od użytkowników, kliknij **Subskrybuj** dla opcji **wiadomości**.

#### 4. Odbierz wiadomość testową

Za każdym razem, gdy zostanie odebrana nowa wiadomość, wywoływane jest zdarzenie klienta **OnMessageReceived**.

```
void OnMessageReceived(TObject *Sender, TsgcWhatsapp_Receive_Message *aMessage, ref bool aMarkAsRead)
{
    DoLog("Received: " + aMessage->Messages->_Message[0]->Id);
}
```

Po skonfigurowaniu webhooka wyślij wiadomość na numer testowy, który został użyty. Powiadomienie z webhookiem zawierające treść wiadomości powinno dotrzeć natychmiast!

*WhatsApp API nie zezwala na wysyłanie wiadomości tekstowych swobodnych do numerów telefonów, które nie nawiązały kontaktu z użytkownikiem w ciągu ostatnich 24 godzin. Jedynym sposobem wysłania wiadomości tekstowej do numeru, który nigdy nie kontaktował się z kontem deweloperskim, jest wysłanie szablonu (wcześniej zatwierdzonego przez Meta). Aby ominąć to ograniczenie podczas testowania wiadomości tekstowych swobodnych, należy najpierw wysłać wiadomość WhatsApp z numeru docelowego na numer konta deweloperskiego, a następnie przez 24 godziny będzie możliwe wysyłanie wiadomości tekstowych swobodnych.*

## **Zdarzenia**

### **OnBeforeSendMessage**

To zdarzenie jest wywoływane przed wysłaniem wiadomości do serwerów WhatsApp. Dostęp do wewnętrznej wiadomości można uzyskać przez parametr `RawMessage`.

### **OnBeforeSubscribe**

To zdarzenie jest wywoływane przed subskrypcją przez serwer danego tematu. Parametr `Accept` pozwala zezwolić lub odrzucić subskrypcję. Domyślnie serwer subskrybuje wszystkie żądane zdarzenia.

### **OnRawMessage**

Zdarzenie jest wywoływane, gdy serwer odbiera nową wiadomość, która nie została jeszcze przetworzona, dzięki czemu uzyskuje się dostęp do surowej wiadomości.

### **OnMessageReceived**

To zdarzenie jest wywoływane po odebraniu i przeanalizowaniu nowej wiadomości przez serwer. Ustawienie parametru `MarkAsRead` na `True` spowoduje, że nadawca otrzyma podwójny znacznik odczytu.

### **OnMessageSent**

Zdarzenie jest wywoływane za każdym razem, gdy serwer odbiera nową wiadomość statusu dotyczącą wcześniej wysłanej wiadomości. Należy odczytać właściwość `Status`, aby sprawdzić, czy wiadomość została wysłana, dostarczona lub przeczytana.





# WhatsApp Tworzenie aplikacji

Przejdź do [developers.facebook.com](https://developers.facebook.com) i kliknij **Utwórz aplikację**.

Wybierz **Business Type** jako typ aplikacji i kontynuuj.



Należy podać nazwę aplikacji (należy unikać zastrzeżonych nazw, takich jak "WhatsApp" lub "Facebook").



Po utworzeniu aplikacji kliknij przycisk **WhatsApp** na następnym ekranie, aby dodać do aplikacji możliwość wysyłania wiadomości przez WhatsApp.



Na następnym ekranie wymagane będzie połączenie aplikacji WhatsApp z kontem firmowym na Facebooku. Dostępna będzie również opcja utworzenia nowego konta firmowego, jeśli użytkownik jeszcze go nie posiada.

# Identyfikator numeru telefonu WhatsApp

Po rejestracji w WhatsApp Cloud API Facebook udostępnia testowy numer telefonu WhatsApp, który będzie domyślnym adresem nadawczym aplikacji. W przypadku odbiorców istnieje możliwość dodania maksymalnie 5 numerów telefonu w fazie rozwoju bez konieczności dokonywania płatności.

Później można zarejestrować własny numer telefonu, aby uniknąć ograniczenia do 5 numerów telefonów.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();  
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
```

# Token WhatsApp

WhatsApp Cloud API wymaga ważnego tokenu do wysyłania jakichkolwiek wiadomości przy użyciu Cloud API.

Facebook udostępnia testowy numer telefonu WhatsApp, który umożliwia wysyłanie wiadomości do maksymalnie 5 numerów telefonów. Ograniczenie to można później usunąć, rejestrując własny numer telefonu.

WhatsApp API udostępnia **tymczasowy token dostępu** ważny przez 23 godziny. Token ten musi zostać skonfigurowany w komponencie TsgcWhatsApp\_Client, aby umożliwić wysyłanie wiadomości.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();  
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
```

Jeśli potrzebny jest token o długim czasie ważności, można utworzyć (lub zaktualizować) użytkownika systemowego i wygenerować nowy token z uprawnieniem **whatsapp\_business\_messaging**. Umożliwi to wysyłanie i odbieranie wiadomości WhatsApp bez konieczności aktualizowania tokena co 23 godziny.

# Webhook WhatsApp

Subskrybuj Webhooki, aby otrzymywać powiadomienia o wiadomościach otrzymywanych przez Twoją firmę oraz aktualizacjach profilu klienta.

## Utwórz endpoint

Przed rozpoczęciem odbierania powiadomień należy utworzyć endpoint na serwerze do ich odbioru.

Punkt końcowy musi obsługiwać dwa typy żądań HTTPS: żądania weryfikacji i powiadomienia o zdarzeniach. Ponieważ oba żądania używają HTTPS, serwer musi posiadać prawidłowy certyfikat TLS lub SSL poprawnie skonfigurowany i zainstalowany. Certyfikaty z podpisem własnym nie są obsługiwane.

Podczas konfigurowania webhooka w ustawieniach WhatsApp należy zdefiniować endpoint, pod którym nasłuchuje serwer, oraz token (może być dowolną wartością). Token jest używany podczas rejestrowania endpointu webhooka i weryfikowania, czy subskrybent jest prawidłowy.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->ServerOptions->WebhookOptions->PhoneNumberId = "/webhook";
oClient->ServerOptions->WebhookOptions->Token = "MySecretToken";
oClient->StartServer();
```

Po skonfigurowaniu Webhooka należy subskrybować pola Webhooka **Messages**, aby otrzymywać powiadomienia za każdym razem, gdy nadejdzie nowa wiadomość.

Więcej informacji na temat konfigurowania [serwera SSL](#).

## Bezpieczeństwo WhatsApp

Za każdym razem, gdy zostanie odebrana nowa wiadomość lub zmieni się status wiadomości, serwer otrzymuje powiadomienie w punkcie końcowym skonfigurowanym w [Webhooku](#). Aby upewnić się, że żądanie pochodzi z serwerów WhatsApp Cloud API, żądanie zawiera nagłówek z podpisem. Można skonfigurować klienta WhatsApp do weryfikacji podpisów przed przetworzeniem wiadomości.

W tym celu należy najpierw ustawić Application Secret we właściwości **ServerOptions.Application.Secret** i włączyć właściwość **VerifySignature**.

Po skonfigurowaniu, przy każdym odebraniu nowej wiadomości, najpierw weryfikowany jest podpis, a jeśli jest nieprawidłowy, zwracany jest błąd 500 i wiadomość nie jest przetwarzana.

# Wysyłanie wiadomości WhatsApp

Wszystkie wywołania API muszą być uwierzytelnione za pomocą tokenu dostępu. Deweloperzy mogą uwierzytelnić swoje wywołania API za pomocą tokenu dostępu wygenerowanego w **App Dashboard > WhatsApp > Getting Started**

Wywołania API zwracają identyfikator wiadomości jako ciąg znaków.

## Wiadomości tekstowe

Wywołaj metodę **SendMessageText** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aText:** treść wiadomości.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageText("34605889421", "Hello from sgcWebSockets!!!");
```

## Wiadomości graficzne

Wywołaj metodę **SendMessageImage** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aLink:** adres URL, pod którym znajduje się obraz do wysłania
- **aCaption:** tytuł obrazu (opcjonalnie).

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageImage("34605889421", "https://www.media.com/image.png", "logo");
```

## Wiadomości dokumentowe

Należy wywołać metodę **SendMessageDocument** i przekazać następujące parametry:

- **aTo:** numer telefonu
- **aLink:** adres URL dokumentu do wysłania
- **aCaption:** tytuł dokumentu (opcjonalny).

- **aFileName:** nazwa pliku (opcjonalna).

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageDocument("34605889421", "https://www.documents.com/file.txt", "Document", "file.txt");
```

## Wiadomości audio

Wywołaj metodę **SendMessageAudio** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aLink:** adres URL audio do wysłania

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageAudio("34605889421", "https://www.audio.com/audio.mp3");
```

## Wiadomości wideo

Wywołaj metodę **SendMessageVideo** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aLink:** adres URL wskazujący na wideo do wysłania

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageVideo("34605889421", "https://www.video.com/audio.mp4");
```

## Wiadomości naklejkowe

Należy wywołać metodę **SendMessageSticker** i przekazać następujące parametry:

- **aTo:** numer telefonu
- **aLink:** adres URL naklejki do wysłania

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageSticker("34605889421", "https://www.stickers.com/sticker");
```

## Wiadomości lokalizacyjne

Wywołaj metodę **SendMessageLocation** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aLongitude:** Długość geograficzna lokalizacji.
- **aLatitude:** Szerokość geograficzna lokalizacji.
- **aName:** Nazwa lokalizacji.
- **aAddress:** Adres lokalizacji. Wyświetlany wyłącznie, gdy ustawiono aName.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageLocation("34605889421", "50.159305", "9.762686", "My Location", "My Address");
```

## Wiadomości kontaktowe

Wywołaj metodę **SendMessageContact** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aName:** Pełna nazwa, w formie normalnie używanej (wymagane).
- **aPhone:** numer telefonu (opcjonalnie).
- **aEmail:** adres e-mail (opcjonalny).

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageLocation("34605889421", "John Smith", "15550386570", "john@mail.com");
```



# WhatsApp — wysyłanie wiadomości interaktywnych

Wiadomości interaktywne oferują użytkownikom prostszy sposób znajdowania i wybierania tego, czego potrzebują od Twojej firmy na WhatsApp. Podczas testów chatboty korzystające z interaktywnych funkcji wiadomości osiągały znacznie wyższe wskaźniki odpowiedzi i konwersji w porównaniu z tymi opartymi wyłącznie na tekście.

Następujące wiadomości są uznawane za interaktywne:

- **Lista wiadomości:** Wiadomości zawierające menu z maksymalnie 10 opcjami. Ten typ wiadomości oferuje prostszy i bardziej spójny sposób dokonywania wyborów przez użytkowników podczas interakcji z firmą.
- **Przyciski odpowiedzi:** Wiadomości zawierające do 3 opcji, z których każda jest przyciskiem. Ten typ wiadomości oferuje użytkownikom szybszy sposób dokonania wyboru z menu podczas interakcji z firmą. Przyciski odpowiedzi mają takie samo działanie jak interaktywne szablony z przyciskami.

## Specyfikacje wiadomości interaktywnych

- Interaktywne wiadomości można łączyć w tym samym przepływie.
- Użytkownicy nie mogą jednocześnie wybrać więcej niż jednej opcji z listy lub komunikatu z przyciskami, ale mogą wrócić i ponownie otworzyć poprzedni komunikat.
- Wiadomości z listą lub przyciskami odpowiedzi nie mogą być używane jako powiadomienia. Obecnie mogą być wysyłane wyłącznie w ciągu 24 godzin od ostatniej wiadomości wysłanej przez użytkownika. Próba wysłania wiadomości poza oknem 24-godzinnym skutkuje komunikatem błędu.

## Kiedy należy tego używać

Wiadomości z listą najlepiej nadają się do prezentowania kilku opcji, takich jak:

- Menu obsługi klienta lub FAQ

- Menu na wynos
- Wybór pobliskich sklepów lub lokalizacji
- Dostępne terminy rezerwacji
- Wybieranie ostatniego zlecenia do powtórzenia

Przyciski odpowiedzi najlepiej sprawdzają się do oferowania szybkich odpowiedzi z ograniczonego zestawu opcji, takich jak:

- Doładowanie środków na konto
- Zmiana danych osobowych
- Ponowne zamawianie poprzedniego zlecenia
- Żądanie zwrotu
- Dodawanie opcjonalnych dodatków do zamówienia jedzenia
- Wybór metody płatności

Przyciski odpowiedzi są szczególnie przydatne w przypadkach „spersonalizowanych”, gdy ogólna odpowiedź jest niewystarczająca.

## **Lista interaktywna**

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageInteractiveList("34605889421", "What Would you like to do today?", "To begin, Tap Main Menu and choose
```



## Przyciski odpowiedzi

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageInteractiveButtons("34605889421", "Select an option", "Which number would you like to add airtime to?",
```



# WhatsApp Wysyłanie Wiadomości Szablonowych

Wywołaj metodę **SendMessageTemplate** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aTemplate:** identyfikator szablonu.
- **aLanguageCode:** język szablonu.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendMessageTemplate("34605889421", "hello_world", "en_US");
```

## Parametry wiadomości szablonu

Szablony mogą zawierać parametry; poniżej znajduje się przykład domyślnego szablonu z parametrami.

```
void SendSamplePurchaseFeedbackTemplate(string aName)
{
    TsgcWhatsApp_Send_Message_Template *oTemplate = new TsgcWhatsApp_Send_Message_Template();
    Try
    {
        oTemplate->Language->Code = "en_US";
        oTemplate->TemplateName = "sample_purchase_feedback";
        // ... header
        TsgcWhatsApp_Send_Message_Template_Component *oComponent = new TsgcWhatsApp_Send_Message_Template_Component();
        oComponent->_Type = wapctHeader;
        oTemplate->Components->Add(oComponent);
        TsgcWhatsApp_Send_Message_Template_Parameter *oParameter = new TsgcWhatsApp_Send_Message_Template_Parameter();
        oParameter->Image->Link = "https://www.esegece.com/images/esegece.png";
        oParameter->_Type = wapptImage;
        oComponent->Parameters->Add(oParameter);
        // ... body
        TsgcWhatsApp_Send_Message_Template_Component *oComponent = new TsgcWhatsApp_Send_Message_Template_Component();
        oComponent->_Type = wapctBody;
        oTemplate->Components->Add(oComponent);
        TsgcWhatsApp_Send_Message_Template_Parameter *oParameter = new TsgcWhatsApp_Send_Message_Template_Parameter();
        oParameter->Text = aName;
        oParameter->_Type = wapptText;
        oComponent->Parameters->Add(oParameter);
        whatsapp->SendMessageTemplate("107809351952205", oTemplate);
    }
    __Finally
    {
        oTemplate->Free();
    }
    End;
}
```

## Przesłany obraz szablonu wiadomości

Poniżej znajduje się przykład szablonu, w którym zamiast użycia łącza do obrazu najpierw przesyła się obraz na serwer, a następnie ustawia identyfikator dokumentu.

```

oClient->OnMessage(TsgcWSConnection *Connection, const string Text)
void SendSamplePurchaseFeedbackTemplate(string aName)
{
    TsgcWhatsApp_Send_Message_Template *oTemplate = new TsgcWhatsApp_Send_Message_Template();
    Try
    // ... first upload the file
    string vId = whatsapp->UploadMedia("c:\\images\\file.png", "image/png");

    // ... send message
    oTemplate->Language->Code = "en_US";
    oTemplate->TemplateName = "sample_purchase_feedback";
    // ... header
    TsgcWhatsApp_Send_Message_Template_Component *oComponent = new TsgcWhatsApp_Send_Message_Template_Component();
    oComponent->_Type = wapctHeader;
    oTemplate->Components->Add(oComponent);
    TsgcWhatsApp_Send_Message_Template_Parameter *oParameter = new TsgcWhatsApp_Send_Message_Template_Parameter();
    oParameter->Image->id = vId;
    oParameter->_Type = wapptImage;
    oComponent->Parameters->Add(oParameter);
    // ... body
    TsgcWhatsApp_Send_Message_Template_Component *oComponent = new TsgcWhatsApp_Send_Message_Template_Component();
    oComponent->_Type = wapctBody;
    oTemplate->Components->Add(oComponent);
    TsgcWhatsApp_Send_Message_Template_Parameter *oParameter = new TsgcWhatsApp_Send_Message_Template_Parameter();
    oParameter->Text = aName;
    oParameter->_Type = wapptText;
    oComponent->Parameters->Add(oParameter);
    whatsapp->SendMessageTemplate("107809351952205", oTemplate);
    __Finally
    oTemplate->Free();
End;
}

```

# WhatsApp — odbieranie wiadomości i powiadomień o statusie

Subskrybuj [Webhooki](#), aby otrzymywać powiadomienia o wiadomościach odbieranych przez firmę oraz aktualizacjach profilu klienta.

Za każdym razem, gdy nastąpi zdarzenie wyzwalające, platforma WhatsApp Business wykrywa je i wysyła powiadomienie na uprzednio podany adres URL webhooka. Możliwe są dwa typy powiadomień:

- **Odebrane wiadomości:** Ten alert informuje o odebraniu wiadomości.
- **Powiadomienia o statusie wiadomości i cenach:** Ten alert informuje o zmianie statusu wiadomości — na przykład o jej przeczytaniu lub dostarczeniu.

## Odebrane wiadomości

Za każdym razem, gdy zostanie odebrana nowa wiadomość, wywoływane jest zdarzenie **OnMessageReceived**, w którym można uzyskać dostęp do treści wiadomości i oznaczyć ją jako przeczytaną.

Poniżej znajduje się przykład: po otrzymaniu nowej wiadomości tekstowej jest ona odsyłana do użytkownika, który ją wysłał.

```
void OnWhatsAppMessageReceived(TObject *Sender, const TsgcWhatsApp_Receive_Message *aMessage, ref bool aMarkAsRead)
{
    if (aMessage->Contacts->Count > 0)
    {
        string vTo = aMessage->Contacts->Contact[0]->WaID;
        if (aMessage->Messages->Count > 0)
        {
            if (aMessage->Messages->_Message[0]->_Type = wapmrtText)
            {
                vText = "ECHO ==> " + aMessage->Messages->_Message[0]->Text->Body;
                WhatsApp->SendMessageText(vTo, vText);
                aMarkAsRead = true;
            }
        }
    }
}
```

## Wysłane wiadomości

Platforma WhatsApp Business wysyła powiadomienia informujące o statusie wiadomości między użytkownikiem a innymi użytkownikami. Gdy wiadomość zostanie pomyślnie wysłana, otrzymuje się powiadomienie o jej wysłaniu, dostarczeniu i przeczytaniu. Kolejność tych powiadomień w aplikacji może nie odzwierciedlać rzeczywistego czasu zmiany statusu wiadomości. W razie potrzeby należy sprawdzić znacznik czasu, aby określić moment zdarzenia.

- **sent:** Następujące powiadomienie jest odbierane, gdy firma wysyła wiadomość w ramach rozmowy zainicjowanej przez użytkownika (jeśli ta rozmowa nie rozpoczęła się w punkcie wejścia bezpłatnego):
- **delivered:** Następujące powiadomienie jest odbierane, gdy wiadomość firmy zostaje dostarczona i jest częścią konwersacji zainicjowanej przez użytkownika (jeśli ta konwersacja nie pochodzi z bezpłatnego punktu wejścia):
- **read:** Następujące powiadomienie jest odbierane, gdy użytkownik przeczyta wiadomość.

Za każdym razem, gdy zostanie odebrany nowy status, wywoływane jest zdarzenie **OnMessageSent**.

```
void OnWhatsAppMessageSent(TObject *Sender, const TsgcWhatsApp_Receive_Message *aMessage, TsgcWhatsAppSendMessageStatusType  
{  
    string vPhone := aMessage.MetaData.DisplayPhoneNumber  
    if (aStatus = wapsmstSent) {  
        DoLog("Message to " + vPhone + " sent.");  
    } else if (aStatus = wapsmstDelivered) {  
        DoLog("Message to " + vPhone + " delivered.");  
    } else if (aStatus = wapsmstRead) {  
        DoLog("Message to " + vPhone + " read.");  
    } else {  
        DoLog("Message to " + vPhone + " unknown status.");  
    }  
}
```

# WhatsApp — wysyłanie plików

Wszystkie wywołania API muszą być uwierzytelnione za pomocą tokenu dostępu. Deweloperzy mogą uwierzytelnić swoje wywołania API za pomocą tokenu dostępu wygenerowanego w **App Dashboard** > **WhatsApp** > **Getting Started**

Wywołania API zwracają identyfikator wiadomości jako ciąg znaków.

W przypadku wysyłania pliku przy użyciu API WhatsApp plik jest najpierw przesyłany na serwery WhatsApp, a następnie wysyłana jest nowa wiadomość zawierająca identyfikator obiektu zwrócony po przesłaniu pliku.

## Wiadomości graficzne

Wywołaj metodę **SendMessageImage** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aFileName:** pełna nazwa pliku (ze ścieżką) obrazu do wysłania.
- **aFileType:**
  - image/jpeg
  - image/png
- **aCaption:** tytuł obrazu (opcjonalnie).

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendFileImage("34605889421", "c:\\images\\image.png", "image/png");
```

## Wiadomości dokumentowe

Należy wywołać metodę **SendMessageDocument** i przekazać następujące parametry:

- **aTo:** numer telefonu
- **aFileName:** pełna nazwa pliku (ze ścieżką) dokumentu do wysłania.
- **aFileType:**
  - text/plain
  - application/pdf
  - application/vnd.ms-powerpoint
  - application/msword
  - application/vnd.ms-excel
  - application/vnd.openxmlformats-officedocument.wordprocessingml.document
  - application/vnd.openxmlformats-officedocument.presentationml.presentation



- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
- **aCaption:** tytuł dokumentu (opcjonalny).

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendFileDocument("34605889421", "c:\\MyDocuments\\invoice.pdf", "application/pdf");
```

## Wiadomości audio

Wywołaj metodę **SendMessageAudio** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aFileName:** pełna nazwa pliku (ze ścieżką) pliku audio do wysłania.
- **aFileType:**
  - audio/aac
  - audio/mp4
  - audio/mpeg
  - audio/amr
  - audio/ogg

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendFileAudio("34605889421", "c:\\Music\\audio.mp3", "audio/mp4");
```

## Wiadomości wideo

Wywołaj metodę **SendMessageVideo** i przekaz następujące parametry:

- **aTo:** numer telefonu
- **aFileName:** pełna nazwa pliku (ze ścieżką) pliku wideo do wysłania.
- **aFileType:**
  - video/mp4
  - video/3gp

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendFileVideo("34605889421", "c:\\Videos\\video.mp4", "video/mp4");
```

## Wiadomości naklejkowe

Należy wywołać metodę **SendMessageSticker** i przekazać następujące parametry:

- **aTo:** numer telefonu
- **aFileName:** pełna nazwa pliku (ze ścieżką) naklejki do wysłania.
- **aFileType:**
  - image/webp

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->SendFileSticker("34605889421", "c:\\Stickers\\MySicker.webp", "image/webp");
```

## WhatsApp — pobieranie multimediów

Jeśli zostanie odebrana wiadomość zawierająca łącze do pliku multimedialnego, plik ten można pobrać przy użyciu metody **DownloadMedia**.

```
TsgcWhatsapp_Client oClient = new TsgcWhatsapp_Client();
oClient->WhatsappOptions->PhoneNumberId = "107809351952205";
oClient->WhatsappOptions->Token = "EAA040pgZAs98BAGj3nCFGr...ZB2t8mmLB2LRXJkte2Y5PMNh2";
oClient->DownloadMedia("38923878928822", "c:\\whatsapp\\media\\image.png");
```

Aby usunąć wcześniej przesłany plik multimedialny, należy wywołać metodę **DeleteMedia** i przekazać identyfikator obiektu jako argument.

# API Telegram

## Telegram

Telegram oferuje dwa rodzaje API: **Bot API** umożliwia tworzenie programów korzystających z botów i protokołu HTTPs, natomiast **Telegram API i TDLib** pozwalają budować niestandardowych klientów Telegram i oferują znacznie więcej możliwości niż Bot API.

sgcWebSockets **obsługuje TDLib za pośrednictwem biblioteki tdjson**, co oznacza, że można zbudować własnego klienta Telegram. TDLib zajmuje się wszystkimi szczegółami implementacji sieciowej, szyfrowaniem i lokalnym przechowywaniem danych. TDLib obsługuje wszystkie funkcje Telegram.

## Zalety TDLib (Telegram Database Library)

- **Wieloplatformowy:** może być używany w systemach Windows, Android, iOS, MacOS, Linux...
- **Łatwość użycia:** do komunikacji między aplikacją a Telegram używane są wiadomości JSON.
- **Wysoka wydajność:** W Telegram Bot API każda instancja TDLib obsługuje ponad 24 000 botów.
- **Consistent:** TDLib gwarantuje, że wszystkie aktualizacje zostaną dostarczone we właściwej kolejności.
- **Niezawodność:** TDLib pozostaje stabilny przy wolnych i zawodnych połączeniach internetowych.
- **Secure:** Wszystkie lokalne dane są szyfrowane przy użyciu klucza szyfrowania podanego przez użytkownika.
- **W pełni asynchroniczny:** Żądania do TDLib nie blokują się nawzajem. Odpowiedzi są wysyłane, gdy są dostępne.

## Konfiguracja

### Windows

TDLib wymaga innych bibliotek firm trzecich: OpenSSL i ZLib. Biblioteki te muszą być wdrożone razem z biblioteką tdjson.

\* Wersje Windows wymagają VCRuntime, który można pobrać od firmy Microsoft: <https://www.microsoft.com/en-us/download/details.aspx?id=52685>. Jeśli problem nadal występuje po instalacji,

należy skopiować następujące biblioteki DLL do folderu, w którym znajduje się aplikacja: VCRUNTIME140.dll i VCRUNTIME140\_1.dll.

Skopiuj następujące biblioteki do tego samego katalogu, w którym znajduje się aplikacja:

|                   |                       |
|-------------------|-----------------------|
| tdjson.dll        | tdjson.dll            |
| libcrypto-1_1.dll | libcrypto-1_1-x64.dll |
| libssl-1_1.dll    | libssl-1_1-x64.dll    |
| zlib1.dll         | zlib1.dll             |

## **OSX64**

-

Wdróż bibliotekę libtdjson.dylib na urządzeniu i możesz ustawić lokalizację biblioteki za pomocą SetTDJsonPath, na przykład:

Jeśli wdrożenie następuje do katalogu „Contents\MacOS\", należy ustawić ścieżkę w folderze TPath.GetDirectoryName(ParamStr(0)).

## **OSXARM64**

-

Wdróż bibliotekę libtdjson.dylib na urządzeniu i możesz ustawić lokalizację biblioteki za pomocą SetTDJsonPath, na przykład:

Jeśli wdrożenie następuje do katalogu „Contents\MacOS\", należy ustawić ścieżkę w folderze TPath.GetDirectoryName(ParamStr(0)).

## **Linux64**

-

Wdróż bibliotekę libtdjson.so na urządzeniu i ustaw ścieżkę biblioteki, wywołując metodę SetTDJsonPath.

## **Android**

-

Należy wdrożyć bibliotekę libtdjsonandroid.so na urządzeniu. Przykład: jeśli wdrażana jest biblioteka Android64, należy ustawić RemotePath w Project/Deployment na "library\lib\arm64-v8a\". Dla wersji Android32 należy ustawić RemotePath na "library\lib\armeabi-v7a\".

## iOS64

-

Skopiuj bibliotekę libtdjson.a do następujących katalogów:

- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\debug
- C:\Program Files (x86)\Embarcadero\Studio\<IDE Version>\lib\iosDevice64\release

## Tworzenie aplikacji Telegram

Aby uzyskać identyfikator API i opracować własną aplikację korzystając z Telegram API, należy wykonać następujące czynności:

- Zarejestruj się w Telegram za pomocą dowolnej aplikacji.
- Zaloguj się do rdzenia Telegram: <https://my.telegram.org>.
- Przejdź do **narzędzi deweloperskich API** i wypełnij formularz.
- Zostaną uzyskane podstawowe adresy, a także parametry **api\_id** i **api\_hash** wymagane do autoryzacji użytkownika.
- Na chwilę obecną każdy numer może mieć podłączony tylko jeden **api\_id**.

Wartości te należy ustawić we właściwości **Telegram.API** komponentu Telegram. W celu uwierzytelnienia można zalogować się jako użytkownik lub jako bot; dostępne są 2 właściwości, które można ustawić, aby zalogować się do Telegrama:

- **PhoneNumber:** w przypadku logowania jako użytkownik należy podać **numer telefonu** (z kodem kierunkowym kraju), na przykład: +34699123456
- **BotToken:** jeśli logujesz się jako bot, ustaw tutaj swój token (zgodnie z informacją od Telegrama).
- **DatabaseDirectory:** umożliwia określenie lokalizacji bazy danych tdlb. Pozostaw puste, aby użyć domyślnej konfiguracji.

Można skonfigurować następujące parametry:

- **ApplicationVersion:** wersja aplikacji, przykład: 1.0
- **DeviceModel:** model urządzenia, przykład: desktop
- **LanguageCode:** kod języka użytkownika, przykład: en.
- **SystemVersion:** wersja systemu operacyjnego, przykład: windows.

Opcjonalnie można skonfigurować ścieżkę, w której znajduje się biblioteka tdjson, używając metody **SetTDJsonPath**. Wystarczy przekazać ścieżkę przed rozpoczęciem nowej sesji Telegram.

Po skonfigurowaniu komponentu Telegram można ustawić właściwość **Active** na **true**, a program podejmie próbę połączenia z Telegram.

### Przykładowy kod

```
TsgcTDLib_Telegram oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "your api hash";
oTelegram->Telegram->API->ApiId = "your api id";
oTelegram->PhoneNumber = "your phone number";
oTelegram->ApplicationVersion = "1.0";
oTelegram->DeviceModel = "Desktop";
oTelegram->LanguageCode = "en";
oTelegram->SystemVersion = "Windows";
oTelegram->Active = true;
```

### Autoryzacja

Istnieją dwa zdarzenia, które mogą być wywoływane przez bibliotekę w celu uzyskania kodu uwierzytelniającego (dostarczanego w aplikacji Telegram, nie SMS-em) lub podania hasła.

### OnAuthenticationCode

To zdarzenie jest wywoływane, gdy Telegram wyśle kod autoryzacyjny do aplikacji Telegram i użytkownik musi skopiować ten kod i wprowadzić go w argumencie **Code** tego zdarzenia.

```
void OnAuthenticationCode(TObject *Sender, ref string Code)
{
```

```
Code = InputBox("Telegram Code", "Introduce code", "");  
}
```

## OnAuthenticationPassword

To zdarzenie jest wywoływane, gdy Telegram wymaga od użytkownika ustawienia hasła.

## Stan autoryzacji

Po rozpoczęciu autoryzacji można sprawdzić jej status w zdarzeniu **OnAuthorizationStatus**, wywoływanym za każdym razem, gdy nastąpi zmiana statusu autoryzacji. Niektóre wartości parametru Status to:

- authorizationStateWaitTdlibParameters
- authorizationStateWaitEncryptionKey
- authorizationStateWaitPhoneNumber
- authorizationStateWaitCode
- authorizationStateLoggingOut
- authorizationStateClosed
- authorizationStateReady

## Stan połączenia

Po rozpoczęciu połączenia można sprawdzić jego status za pomocą zdarzenia **OnConnectionStatus**, które jest wywoływane za każdym razem, gdy zmienia się stan połączenia. Przykładowe wartości Status:

- connectionStateConnecting
- connectionStateUpdating
- connectionStateReady

## Główne metody

Komponent API TsgcTDLib\_Telegram obsługuje kilka metod Telegrama; poniżej przedstawiono najczęściej używane.





|                            |                                                                                                                                                                                              |                                                                                                                                                                                                                                                            |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SendTextMessage</b>     | <p><b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość <b>aText:</b> Treść wiadomości.</p>                                                                            | Wysyła wiadomość tekstową do czatu.                                                                                                                                                                                                                        |
|                            | <p><b>InlineKeyboard:</b> Opcjonalne przyciski (wyłącznie boty).</p>                                                                                                                         |                                                                                                                                                                                                                                                            |
| <b>SendRichTextMessage</b> | <p><b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość <b>aText:</b> Treść wiadomości.</p>                                                                            | Wysyła sformatowaną wiadomość tekstową do czatu. Składnia Markdown:                                                                                                                                                                                        |
|                            | <p><b>InlineKeyboard:</b> Opcjonalne przyciski (wyłącznie boty).</p>                                                                                                                         | <ul style="list-style-type: none"> <li>• Pogrubienie: <b>**bold**</b></li> <li>• Kursywa: <i>__italic__</i></li> <li>• Przekreślenie: <del>--strike--</del></li> <li>• Podkreślenie: <u>~~underline~~</u></li> <li>• Kod: <code>##code##</code></li> </ul> |
| <b>SendDocumentMessage</b> | <p><b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość <b>aFilePath:</b> pełna ścieżka pliku dokumentu <b>aInlineKeyboard:</b> Opcjonalne przyciski (tylko boty).</p> | Wysyła dokument do czatu.                                                                                                                                                                                                                                  |
| <b>SendPhotoMessage</b>    | <p><b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość <b>aFilePath:</b> pełna ścieżka pliku zdjęcia</p>                                                              |                                                                                                                                                                                                                                                            |
|                            | <b>Width:</b> szerokość zdjęcia.                                                                                                                                                             | Wysyła zdjęcie do czatu.                                                                                                                                                                                                                                   |
|                            | <b>Height:</b> szerokość zdjęcia.                                                                                                                                                            |                                                                                                                                                                                                                                                            |
|                            | <p><b>InlineKeyboard:</b> Opcjonalne przyciski (wyłącznie boty).</p>                                                                                                                         |                                                                                                                                                                                                                                                            |
| <b>SendVideoMessage</b>    | <p><b>aChatId:</b> identyfikator czatu, do którego zostanie wysłana wiadomość <b>aFilePath:</b> pełna ścieżka pliku wideo <b>aWidth:</b> szerokość wideo.</p>                                |                                                                                                                                                                                                                                                            |
|                            | <b>Wysokość:</b> szerokość wideo.                                                                                                                                                            | Wysyła wideo do czatu.                                                                                                                                                                                                                                     |
|                            | <b>aDuration:</b> czas trwania wideo w sekundach.                                                                                                                                            |                                                                                                                                                                                                                                                            |
|                            | <p><b>aInlineKeyboard:</b> Opcjonalne przyciski (tylko dla botów).</p>                                                                                                                       |                                                                                                                                                                                                                                                            |
| <b>SendInvoiceMessage</b>  | <p><b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość <b>aInvoice:</b> Treść wiadomości.</p>                                                                         | Wysyła fakturę (dostępne tylko gdy jest botem i w kanałach prywatnych).                                                                                                                                                                                    |
|                            | <p><b>aInlineKeyboard:</b> Opcjonalne przyciski (tylko dla botów).</p>                                                                                                                       |                                                                                                                                                                                                                                                            |

|                                |                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                 |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                | <b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                 |
|                                | <b>aMessageId:</b> Identyfikator wiadomości do zmodyfikowania                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                 |
| <b>EditTextMessage</b>         | <b>Text:</b> Treść wiadomości.                                                                                                                                                                                                                                                       | Edytuje tekst wiadomości (lub tekst wiadomości gry)                                                                                                                                                                                                                                                             |
|                                | <b>InlineKeyboard:</b> Opcjonalne przyciski (wyłącznie boty).                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                 |
|                                | <b>ShowKeyboard:</b> Opcjonalne przyciski (tylko boty).                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                 |
| <b>AddChatMember</b>           | <b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość. <b>aUserId:</b> Identyfikator użytkownika. <b>aForwardLimit:</b> Liczba wcześniejszych wiadomości z czatu do przekazania nowemu członkowi; maksymalnie 100. Ignorowane w przypadku supergroup i kanałów. | Dodaje nowego członka do czatu. Nie można dodawać członków do prywatnych lub tajnych czatów. Członkowie nie zostaną dodani, dopóki stan czatu nie zostanie zsynchronizowany z serwerem.                                                                                                                         |
| <b>AddChatMembers</b>          | <b>aChatId:</b> Identyfikator czatu, do którego zostanie wysłana wiadomość. <b>aUserIds:</b> Identyfikatory użytkowników, którzy mają zostać dodani do czatu.                                                                                                                        | Dodaje wielu nowych członków do czatu. Opcja ta jest obecnie dostępna tylko dla supergrup i kanałów. Nie można jej używać do dołączania do czatu. Nie można dodawać członków do kanału, jeśli liczy on więcej niż 200 członków. Członkowie nie będą dodani do momentu zsynchronizowania stanu czatu z serwerem. |
| <b>GetChatMember</b>           | <b>aChatId:</b> Identyfikator czatu. <b>aUserId:</b> Identyfikator użytkownika.                                                                                                                                                                                                      | Zwraca informacje o pojedynczym członku czatu.                                                                                                                                                                                                                                                                  |
| <b>GetBasicGroupFullInfo</b>   | <b>aGroupId:</b> Identyfikator grupy podstawowej                                                                                                                                                                                                                                     | Zwraca pełne informacje o grupie podstawowej na podstawie jej identyfikatora.                                                                                                                                                                                                                                   |
|                                | <b>aSuperGroupId:</b> Identyfikator supergrupy lub kanału.                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                 |
| <b>GetSupergroupMembers</b>    | <b>aSupergroupMembersFilter:</b> Typ użytkowników do zwrócenia. Domyślnie null. <b>aOffset:</b> Liczba użytkowników do pominięcia.                                                                                                                                                   | Zwraca informacje o członkach lub zbanowanych użytkownikach supergrupy lub kanału.                                                                                                                                                                                                                              |
|                                | <b>aLimit:</b> Maksymalna liczba zwracanych użytkowników; do 200.                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                 |
| <b>JoinChatByInviteLink</b>    | <b>aLink:</b> Link zaproszenia do zaimportowania;                                                                                                                                                                                                                                    | Używa łącza z zaproszeniem, aby dodać bieżącego użytkownika do czatu, jeśli jest to możliwe. Nowy członek nie zostanie dodany do chwili zsynchronizowania stanu czatu z serwerem.                                                                                                                               |
| <b>CreateNewSecretChat</b>     | <b>aUserId:</b> Identyfikator użytkownika.                                                                                                                                                                                                                                           | Tworzy nowy tajny czat.                                                                                                                                                                                                                                                                                         |
| <b>CreateNewBasicGroupChat</b> | <b>aUserIds:</b> Identyfikatory użytkowników, którzy mają zostać dodani do czatu. <b>aTitle:</b> Tytuł nowej podstawowej grupy                                                                                                                                                       | Tworzy nową grupę podstawową                                                                                                                                                                                                                                                                                    |

|                                |                                                                                                                                                                                                        |                                                                                                                                                         |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CreateNewSupergroupChat</b> | <b>aTitle:</b> Tytuł nowej SuperGrupy                                                                                                                                                                  | Tworzy nową supergrupę lub kanał.                                                                                                                       |
|                                | <b>aIsChannel:</b> True, jeśli ma zostać utworzony czat kanałowy. <b>aDescription:</b> Opis czatu.                                                                                                     |                                                                                                                                                         |
| <b>CreatePrivateChat</b>       | <b>aUserId:</b> Identyfikator użytkownika.                                                                                                                                                             | Zwraca istniejący czat odpowiadający danemu użytkownikowi                                                                                               |
|                                | <b>aForce:</b> Jeśli true, czat zostanie utworzony bez żądania sieciowego. W takim przypadku wszystkie informacje o czacie z wyjątkiem jego typu, tytułu i zdjęcia mogą być nieprawidłowe.             |                                                                                                                                                         |
| <b>GetChats</b>                | <b>aOffsetOrder:</b> Kolejność czatów, od której należy zwracać czaty <b>aOffsetChatId:</b> Identyfikator czatu, od którego należy zwracać czaty <b>aLimit:</b> Maksymalna liczba czatów do zwrócenia. | Zwraca uporządkowaną listę czatów. Czaty są posortowane według pary (order, chat_id) w kolejności malejącej (nie można użyć, jeśli zalogowany jako Bot) |
| <b>GetChat</b>                 | <b>aChatId:</b> Identyfikator czatu                                                                                                                                                                    | Zwraca informacje o czacie na podstawie jego identyfikatora                                                                                             |
| <b>GetChatHistory</b>          | <b>aChatId:</b> Identyfikator czatu                                                                                                                                                                    | Zwraca wiadomości w czacie. Wiadomości są zwracane w odwrotnym porządku chronologicznym                                                                 |
|                                | <b>aFromMessageId:</b> Identyfikator wiadomości, od której należy pobrać historię; użyj 0, aby pobrać wyniki od ostatniej wiadomości.                                                                  |                                                                                                                                                         |
| <b>GetUser</b>                 | <b>aOffset:</b> Podaj 0, aby uzyskać wyniki dokładnie od from_message_id, lub ujemne przesunięcie do 99, aby dodatkowo pobrać nowsze wiadomości.                                                       | Zwraca informacje o użytkowniku na podstawie jego identyfikatora.                                                                                       |
|                                | <b>aLimit:</b> Maksymalna liczba wiadomości do zwrócenia                                                                                                                                               |                                                                                                                                                         |
| <b>AddProxyHTTP</b>            | <b>aUserId:</b> Identyfikator użytkownika                                                                                                                                                              | Dodaje serwer proxy HTTP dla żądań sieciowych. Można wywołać przed autoryzacją.                                                                         |
|                                | <b>aServer:</b> Nazwa serwera proxy.                                                                                                                                                                   |                                                                                                                                                         |
|                                | <b>aPort:</b> Numer portu proxy.                                                                                                                                                                       |                                                                                                                                                         |
|                                | <b>aUserName:</b> nazwa użytkownika do logowania; może być pusta.                                                                                                                                      |                                                                                                                                                         |
| <b>AddProxyHTTP</b>            | <b>aPassword:</b> Hasło do logowania; może być puste.                                                                                                                                                  |                                                                                                                                                         |
|                                | <b>aHTTPOnly:</b> Przekaz true, jeśli proxy obsługuje tylko żądania HTTP i nie obsługuje przezroczystych połączeń TCP przez metodę HTTP CONNECT.                                                       |                                                                                                                                                         |

|                                |                                                                                                                                                                                                        |                                                                                                                                                         |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AddProxyMT-Proto</b>        | <b>aServer:</b> Nazwa serwera proxy.<br><b>aPort:</b> Numer portu proxy. <b>aSecret:</b> Sekret proxy w kodowaniu szesnastkowym.                                                                       | Dodaje serwer proxy MTProto dla żądań sieciowych. Może być wywołane przed autoryzacją.                                                                  |
| <b>AddProxy-Socks5</b>         | <b>aServer:</b> Nazwa serwera proxy.<br><b>aPort:</b> Numer portu proxy.<br><b>aUserName:</b> nazwa użytkownika do logowania; może być pusta.<br><b>aPassword:</b> Hasło do logowania; może być puste. | Dodaje serwer proxy Socks5 dla żądań sieciowych. Może być wywołane przed autoryzacją.                                                                   |
| <b>EnableProxy</b>             | <b>aId:</b> Identyfikator proxy                                                                                                                                                                        | Włącza serwer proxy. Jednocześnie może być włączony tylko jeden serwer proxy. Może być wywoływane przed autoryzacją.                                    |
| <b>DisableProxy</b>            |                                                                                                                                                                                                        | Wyłącza aktualnie włączone proxy. Można wywołać przed autoryzacją.                                                                                      |
| <b>RemoveProxy</b>             | <b>aId:</b> Identyfikator proxy                                                                                                                                                                        | Usuwa serwer proxy. Może być wywołana przed autoryzacją.                                                                                                |
| <b>GetProxies</b>              |                                                                                                                                                                                                        | Zwraca listę aktualnie skonfigurowanych serwerów proxy. Można wywołać przed autoryzacją.                                                                |
| <b>getChatSponsoredMessage</b> | <b>aChatId:</b> Identyfikator czatu                                                                                                                                                                    | Zwraca sponsorowaną wiadomość do wyświetlenia na czacie; tylko dla czatów kanałowych. Zwraca błąd 404, jeśli na czacie nie ma sponsorowanej wiadomości. |
| <b>ViewMessage</b>             | <b>aSponsorChatId:</b> Identyfikator czatu sponsora<br><b>aMessageId:</b> identyfikator wiadomości                                                                                                     | Informuje TDLib, że użytkownik aktualnie przegląda wiadomości. Wiele przydatnych operacji zależy od tego, czy wiadomości są w danej chwili wyświetlane  |
| <b>Wylogowanie</b>             |                                                                                                                                                                                                        | Wylogowuje z Telegrama.                                                                                                                                 |
| <b>TDLibSend</b>               | <b>aRequest:</b> Żądanie JSON.                                                                                                                                                                         | Wyślij dowolne żądanie w protokole JSON.                                                                                                                |

## Przykład wysyłania wiadomości tekstowej

```

TsgcTDLib_Telegram oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "your api hash";
oTelegram->Telegram->API->ApiId = "your api id";
oTelegram->PhoneNumber = "your phone number";
oTelegram->Active = true;
...
oTelegram->SendTextMessage("1234", "My First Message from sgcWebSockets");

```

## Przykład: wysyłanie odpowiedzi „metoda nie zaimplementowana”

Dowolną wiadomość JSON można wysłać za pomocą metody `TDLibSend`, na przykład w celu dołączenia do czatu Telegram.

```
TsgcTDLib_Telegram oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "your api hash";
oTelegram->Telegram->API->ApiId = "your api id";
oTelegram->PhoneNumber = "your phone number";
oTelegram->Active = true;
...
oTelegram->TDLibSend("{\"@type\": \"joinChat\", \"chat_id\": \"1234\"}");
```

## Zdarzenia

### OnBeforeReadEvent

Zdarzenie jest wywoływane, gdy komponent API Telegram odbierze wiadomość JSON, która nie została jeszcze przetworzona. Należy ustawić właściwość `Handled` na `True`, jeśli zdarzenie jest przetwarzane ręcznie lub nie ma być obsługiwane przez komponent. Zdarzenia można również używać do rejestrowania wszystkich wiadomości.

### OnMessageText

Zdarzenie wywoływane po odebraniu nowej wiadomości tekstowej; należy odczytać parametr `MessageText`, aby uzyskać dostęp do właściwości tekstu wiadomości.

- **ChatId:** identyfikator czatu.
- **MessageId:** Identyfikator wiadomości.
- **SenderId:** Identyfikator nadawcy.
- **Text:** Treść wiadomości.

### OnMessageDocument

Zdarzenie jest wywoływane po odebraniu nowej wiadomości dokumentu. Dostęp do właściwości dokumentu można uzyskać przez `MessageDocument`.

- **ChatId:** identyfikator czatu.
- **MessageId:** Identyfikator wiadomości.
- **SenderId:** Identyfikator nadawcy (odczytaj `SenderChat` i `SenderUser` z `tdlib 1.7.+`).
- **FileName:** Nazwa dokumentu.
- **DocumentId:** Identyfikator dokumentu.
- **LocalPath:** pełna ścieżka do lokalnego pliku, jeśli istnieje.
- **MimeType:** Typ MIME dokumentu.
- **Size:** Rozmiar dokumentu.
- **RemoteDocumentId:** Identyfikator dokumentu zdalnego.

## OnMessagePhoto

Zdarzenie jest wywoływane po odebraniu nowej wiadomości ze zdjęciem. Dostęp do właściwości zdjęcia uzyskuje się za pośrednictwem obiektu MessagePhoto.

- **ChatId:** identyfikator czatu.
- **MessageId:** Identyfikator wiadomości.
- **SenderId:** Identyfikator nadawcy (odczytaj SenderChat i SenderUser z tdlb 1.7.+).
- **PhotoId:** identyfikator zdjęcia.
- **LocalPath:** pełna ścieżka do lokalnego pliku, jeśli istnieje.
- **Size:** Rozmiar zdjęcia.
- **RemotePhotoId:** Zdalny identyfikator zdjęcia.

## OnVideoPhoto

To zdarzenie jest wywoływane po odebraniu nowej wiadomości wideo. Dostęp do MessageVideo umożliwia odczyt właściwości wideo.

- **ChatId:** identyfikator czatu.
- **MessageId:** Identyfikator wiadomości.
- **SenderId:** Identyfikator nadawcy (odczytaj SenderChat i SenderUser z tdlb 1.7.+).
- **VideoId:** Identyfikator zdjęcia.
- **LocalPath:** pełna ścieżka do lokalnego pliku, jeśli istnieje.
- **Width:** szerokość obrazu wideo.
- **Height:** wysokość obrazu wideo.
- **Duration:** czas trwania wideo w sekundach.
- **Size:** Rozmiar wideo.
- **RemoteVideoId:** Zdalny identyfikator zdjęcia.

## OnMessageSponsored

To zdarzenie jest wywoływane po odebraniu nowej sponsorowanej wiadomości (po wywołaniu metody getChatSponsoredMessage)

- **SponsorChatId:** Identyfikator czatu sponsora.
- **MessageId:** Identyfikator wiadomości.
- **Text:** Treść wiadomości.

## OnNewChat

To zdarzenie jest wywoływane po odebraniu nowego czatu.

- **ChatId:** identyfikator czatu.
- **ChatType:** Typ czatu (chatTypeSupergroup, chatTypePrivate...)
- **Title:** Nazwa czatu.

- **SuperGroupId:** Identyfikator grupy, jeśli jest to SuperGrupa.
- **IsChannel:** zwraca informację, czy jest to kanał, czy nie.

## OnNewCallbackQuery

To zdarzenie jest wywoływane po odebraniu nowego przychodzącego zapytania zwrotnego; tylko dla botów.

- **Id:** Unikalny identyfikator zapytania.
- **SenderId:** Identyfikator użytkownika, który wysłał zapytanie.
- **ChatId:** identyfikator czatu, w którym zostało wysłane zapytanie.
- **MessageId:** Identyfikator wiadomości, z której pochodzi zapytanie.
- **ChatInstance:** Identyfikator jednoznacznie odpowiadający czatowi, do którego wysłano wiadomość.
- **PayloadData:** ładunek z ogólnego przycisku wywołania zwrotnego.
  - **Data:** Dane dołączone do przycisku wywołania zwrotnego.

## OnEvent

Zdarzenie to jest wywoływane, gdy komponent API odbierze nowe zdarzenie. Może być używane do przetwarzania zdarzeń niezaimplementowanych przez komponent API.

- **Zdarzenie:** Nazwa zdarzenia (zdarzenia takie jak: updateOption, updateUser...)
- **Text:** pełna wiadomość JSON

## OnException

To zdarzenie jest wywoływane, gdy podczas przetwarzania danych Telegram API wystąpi wyjątek.

## Właściwości

**MyId:** zwraca identyfikator bieżącego użytkownika.

## Pełny przykład kodu

Zapoznaj się z poniższym przykładem kodu, który pokazuje, jak połączyć się z Telegram API, poprosić użytkownika o wprowadzenie kodu (jeśli jest to wymagane przez Telegram API), wysłać wiadomość po nawiązaniu połączenia oraz rejestrować odbierane wiadomości tekstowe.

```
TsgcTDLib_Telegram oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "your api hash";
oTelegram->Telegram->API->ApiId = "your api id";
oTelegram->PhoneNumber = "your phone number";
oTelegram->ApplicationVersion = "1.0";
oTelegram->DeviceModel = "Desktop";
```

```
oTelegram->LanguageCode = "en";
oTelegram->SystemVersion = "Windows";
oTelegram->Active = true;

void OnAuthenticationCode(TObject *Sender, ref string Code)
{
    Code = InputBox("Telegram Code", "Introduce code", "");
}

void OnMessageText(TObject *Sender, TsgcTelegramMessageText *MessageText)
{
    Log("Message Received: " + MessageText->Text);
}

void OnConnectionStatus(TObject *Sender, const string Status)
{
    if (Status == "connectionStateReady")
    {
        oTelegram->SendTextMessage("1234", "Hello Telegram!");
    }
}
```



# Telegram | Wysyłanie wiadomości Telegram z przyciskami inline

Telegram API umożliwia wysyłanie wiadomości z przyciskami inline do wyboru opcji jako odpowiedzi (opcja dostępna wyłącznie dla botów).

Przed wysłaniem wiadomości należy utworzyć instancję klasy **TsgcTelegramReplyMarkupInlineKeyboard** i wywołać metodę **AddButtonTypeCallback** lub **AddButtonTypeUrl** dla każdego przycisku, który ma zostać utworzony.

## Przykład

Utwórz nową wiadomość pytającą użytkownika, czy podoba mu się wiadomość, wraz z linkiem do ankiety. Przetwórz odpowiedź za pomocą zdarzenia **OnNewCallbackQuery**.

```
TsgcTelegramReplyMarkupInlineKeyboard *oReplyMarkup = new TsgcTelegramReplyMarkupInlineKeyboard();
try
{
    oReplyMarkup->AddButtonTypeCallback("Yes", "I like it");
    oReplyMarkup->AddButtonTypeCallback("No", "I hate it");
    oReplyMarkup->AddButtonTypeUrl("Poll", "https://www.yoursite.com/telegram/poll");
    sgctelegram->SendTextMessage("123456", "Do you like the message?", oReplyMarkup);
}
__finally
{
    oReplyMarkup->Free();
}

void OnNewCallbackQuery(TObject *Sender, TsgcTelegramCallbackQuery *CallbackQuery)
{
    if (CallbackQuery->PayloadData->Data == "I like it") then
    {
        ShowMessage("yes")
    }
    else
    {
        ShowMessage("no");
    }
}
```

## Telegram | Wyślij wiadomość bota z przyciskami

API Telegram umożliwia wysyłanie wiadomości z przyciskami służącymi do żądania danych od użytkownika (opcja dostępna wyłącznie dla botów).

Przed wysłaniem wiadomości należy utworzyć instancję klasy **TsgcTelegramReplyMarkupShowKeyboard** i wywołać metodę **AddButtonTypeRequestLocation**, **AddButtonTypeRequestPhoneNumber** lub **AddButtonTypeText** dla każdego przycisku, który ma zostać utworzony.

### Przykład

Utwórz nową wiadomość z prośbą do użytkownika o podanie numeru telefonu

```
oReplyMarkup = new TsgcTelegramReplyMarkupShowKeyboard();
oReplyMarkup->AddButtonTypeRequestPhoneNumber("Give me your phone");
sgcTelegram->SendTextMessage("123456", "Please provide the information below", null, oReplyMarkup);
oReplyMarkup->Free();
```

# Telegram | Wyślij pogrubioną wiadomość Telegram

Możliwe jest wyróżnianie tekstu przy użyciu pogrubienia, kursywy i innych stylów. Należy użyć metody **SendRichTextMessage**, aby wysłać wiadomość tekstową z możliwościami stylistycznymi; metoda ta automatycznie analizuje wiadomość i dodaje wymagane encje do API Telegram.

## Składnia Markdown

- Bold [ \* ]

**\*\*This is Bold\*\***

- Kursywa [ \_ ]

*\_\_This is Italic\_\_*

- Strike [ - ]

~~--This is Strike--~~

- Podkreślenie [ ~ ]

~~This is Underline~~

- Kod [ # ]

`##This is Monospace##`

## Telegram | Czat nie znaleziony jako Bot

Przy **logowaniu jako bot** metody GetChats nie można użyć, dlatego nie są dostępne wszystkie czaty. Jeśli jest to **pierwsze logowanie jako bot** i nastąpi próba **wysłania wiadomości do znanego czatu**, wystąpi następujący błąd:

```
{"@type":"error","code":5,"message":"Chat not found"}
```

Rozwiązaniem jest wywołanie metody **GetChat** przed wysłaniem wiadomości w Telegram i przekazanie **ChatId** jako parametru. Po uzyskaniu danych czatu można wysyłać wiadomości w Telegram w standardowy sposób.

Należy zauważyć, że wywołanie **GetChat** jest potrzebne **wyłącznie za PIERWSZYM RAZEM** przed wysłaniem wiadomości, jeśli nie otrzymano jeszcze żadnej wiadomości od bota z tego czatu. Po zamknięciu i ponownym uruchomieniu aplikacji nie ma potrzeby ponownego wywoływania GetChat, ponieważ czat jest już zapisany w bazie danych Telegram.

## Telegram | Sponsorowane wiadomości

Za każdym razem, gdy użytkownik otwiera kanał, należy wywołać `channels.getSponsoredMessages`, aby pobrać sponsorowane wiadomości dostępne dla tego kanału. Wynik powinien być buforowany przez 5 minut.

### Wyświetlanie sponsorowanych wiadomości

Sponsorowane wiadomości muszą być wyświetlane poniżej wszystkich innych wpisów na kanale, po tym, jak użytkownik przewinie dalej w dół, za ostatnią wiadomością. Promowany kanał lub bot określony w polu `from_id` musi być wyświetlany jako autor wiadomości. Wiadomość powinna również zawierać jeden z następujących przycisków na dole:

- **View Bot:** jeśli promowany jest bot, naciśnięcie przycisku musi otworzyć czat z botem. Jeśli podano parametr `start_param`, aplikacja musi użyć mechanizmu deep linking, aby otworzyć bota.
- **Wyświetl kanał:** jeśli kanał jest promowany. Naciśnięcie przycisku musi otworzyć dany kanał.
- **Wyświetl wpis:** jeśli kanał jest promowany i podano `channel_post`. Naciśnięcie przycisku musi otworzyć dany wpis kanału.

Po wyświetleniu całego tekstu na ekranie (z wyłączeniem przycisku) należy wywołać metodę **ViewMessage** z parametrem `random_id` tej sponsorowanej wiadomości.

### Pobierz wiadomości sponsorowane

Wyślij żądanie do kanału z pytaniem o dostępne wiadomości sponsorowane, wywołując metodę **GetChatSponsoredMessage**.

```
TsgcTDLib_Telegram *oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "ABCDEFGHijklmn";
oTelegram->Telegram->API->ApiId = "1234";
oTelegram->PhoneNumber = "008745744155";
oTelegram->Active = true;
oTelegram->getChatSponsoredMessage("100");
```

Jeśli czat zawiera sponsorowane wiadomości, zdarzenie **OnMessageSponsored** jest wywoływane z treścią sponsorowanej wiadomości. Jeśli nie ma żadnych wiadomości, zwracany jest błąd 404.

```
private void(TObject *Sender, TsgcTelegramMessageSponsored *MessageSponsored)
{
    DoLog(MessageSponsored->Text);
}
```

Należy wywołać metodę **ViewMethod** po wyświetleniu użytkownikowi wiadomości sponsorowanych.

```
oTelegram->ViewMessage("100", "54653256245");
```

## Telegram | Wyślij fakturę Telegram

Jeśli bot obsługuje tryb inline, użytkownicy mogą również wysyłać faktury do innych czatów za pośrednictwem bota, w tym do prywatnych rozmów z innymi użytkownikami.

Wiadomości fakturowe zawierają zdjęcie i opis produktu wraz z widocznym przyciskiem Zapłać. Dotknięcie tego przycisku otwiera specjalny interfejs płatności w aplikacji Telegram

Boty mogą wysyłać faktury jako wiadomości przy użyciu metody **SendInvoiceMessage**.

```
private void SendInvoice()
{
    TsgcTelegramSendInvoice *oInvoice = new TsgcTelegramSendInvoice();
    Try
    {
        oInvoice->Title = "Invoice Title Test";
        oInvoice->Description = "Description Invoice Test";
        oInvoice->Invoice->Currency = 'EUR';
        oInvoice->Invoice->Total = 800;
        oInvoice->Invoice->IsTest = True;
        oInvoice->Invoice->Payload := "payload";
        oInvoice->Invoice->ProviderToken := "provider_token";
        oInvoice->Invoice->ProviderData := "provider_data";

        sgcTelegram->SendInvoiceMessage("3284239872", oInvoice);
    }__finally
    {
        oInvoice->Free();
    }
}
```

## Telegram | Pobierz członków supergrupy

API Telegram umożliwia pobieranie informacji o członkach SuperGroup. Użyj metody **GetSuperGroupMembers**, aby uzyskać informacje o członkach lub zbanowanych użytkownikach w supergroup lub kanale. Można używać tylko wtedy, gdy SuperGroupFullInfo.can\_get\_members wynosi true; ponadto w przypadku niektórych filtrów mogą być wymagane uprawnienia administratora.

Domyślnie metoda zwraca wszystkich członków grupy, ale można filtrować zwracanych członków za pomocą parametru Filter. Dostępne są następujące parametry:

### **tsgmFilterNone**

Wartość domyślna oznacza, że członkowie nie są filtrowani.

### **tsgmFilterAdministrators**

Zwraca twórcę i administratorów.

### **tsgmFilterBanned**

Zwraca użytkowników zablokowanych w supergrupie lub kanale; może być używane wyłącznie przez administratorów.

Argumentu aSuperGroupMembersQuery można używać do wyszukiwania za pomocą zapytania.

### **tsgmFilterBots**

Zwraca członków będących botami w supergroup lub kanale.

### **tsgmFilterContacts**

Zwraca kontakty użytkownika, które są członkami supergrupy lub kanału.

Argumentu aSuperGroupMembersQuery można używać do wyszukiwania za pomocą zapytania.

### **tsgmFilterMention**

Zwraca użytkowników, których można wspomnieć w supergrupie.

### **tsgmFilterRecent**

Zwraca ostatnio aktywnych użytkowników w odwrotnej kolejności chronologicznej.

### **tsgmFilterRestricted**

Zwraca ograniczonych członków supergrupy; może być używane wyłącznie przez administratorów.

Argumentu aSuperGroupMembersQuery można używać do wyszukiwania za pomocą zapytania.

### **tsgmFilterSearch**

Służy do wyszukiwania członków supergrupy lub kanału za pomocą zapytania (ciągu znaków).

Argumentu aSuperGroupMembersQuery można używać do wyszukiwania za pomocą zapytania.

Wynik można odczytać za pomocą wywołania zwrotnego OnEvent, filtrując po event = "chatMembers".

```
Telegram->GetSupergroupMembers(1452979380);
```

```
private void OnTelegramEvent(TObject *Sender, const string Event, const string Text)
{
    if (Event == "chatMembers")
    {
        ReadJSON(Text);
    }
}
```



# Telegram | Dodawanie proxy Telegram

Klient Telegram można skonfigurować do korzystania z serwera proxy. Aktualnie Telegram obsługuje 3 typy serwerów proxy:

1. HTTP
2. MTProto
3. Socks5

## Dodaj proxy

Aby skonfigurować serwer proxy HTTP, należy najpierw dodać proxy do konfiguracji Telegram — w tym celu należy wywołać **AddProxyHTTP**, a w przypadku powodzenia zostanie zwrócona wiadomość z nowo dodanym proxy. Po dodaniu proxy do listy należy wywołać **EnableProxy** i przekazać **identyfikator proxy** otrzymany w wiadomości potwierdzającej.

```
Telegram->AddProxyHTTP("8.8.8.8", 8080, "", "", true);  
// ... read the confirmation message and save the ID of the proxy.  
Telegram->EnableProxy(2);
```

## Usuń proxy

Wywołaj metodę **RemoveProxy** i przekaz identyfikator serwera proxy, który ma zostać usunięty.

# Telegram | Rejestracja użytkownika Telegram

Proces rejestracji nowego użytkownika w Telegram jest bardzo prosty: potrzebne jest API Id, API Hash oraz numer telefonu nowego konta.

Skonfiguruj klienta Telegram:

- Identyfikator API
- API Hash
- Numer telefonu nowego konta Telegram.

Uruchom klienta, a na telefon zostanie wysłany nowy kod. Klient poprosi o podanie kodu Telegram, a jeśli będzie on poprawny, zostanie wywołane zdarzenie OnRegisterUser. W tym zdarzeniu należy ustawić imię i nazwisko użytkownika, co zakończy rejestrację.

```
TsgcTDLib_Telegram *oTelegram = new TsgcTDLib_Telegram();
oTelegram->Telegram->API->ApiHash = "ABCDEFGHijklmn";
oTelegram->Telegram->API->ApiId = "1234";
oTelegram->PhoneNumber = "008745744155";
oTelegram->Active = true;

void OnTelegramAuthenticationCode(TObject *Sender, ref string Code)
{
    Code = "code sent to phone";
}

void OnTelegramRegisterUser(TObject *Sender, ref string FirstName, ref string LastName)
{
    FirstName = "first name";
    LastName = "last name";
}
```

# RCON

## RCON

Source RCON Protocol to protokół komunikacyjny oparty na TCP/IP używany przez Source Dedicated Server, który umożliwia wydawanie poleceń konsolowych serwerowi przez „zdalną konsolę” (RCON). Najczęstszym zastosowaniem RCON jest umożliwienie właścicielom serwerów zarządzania serwerami gier bez bezpośredniego dostępu do maszyny, na której działa serwer.

## Konfiguracja

Właściwość **RCON\_Options** umożliwia skonfigurowanie następujących właściwości:

- **Host:** zdalny adres serwera.
- **Port:** port nasłuchiwania serwera.
- **Password:** jest tajnym ciągiem używanym do uwierzytelniania względem serwera

## Połącz

Użyć właściwości **Active**, aby połączyć się / rozłączyć z serwerem.

Gdy właściwość Active jest ustawiona na True, klient próbuje połączyć się z serwerem; jeśli połączenie się powiedzie, spróbuje się uwierzytelnić przy użyciu podanego hasła.

Serwer wyśle odpowiedź na żądanie uwierzytelnienia. Zdarzenie **OnAuthenticate** zostanie wywołane i można odczytać, czy uwierzytelnianie się powiodło, używając parametru Authenticate.

## Wyślij komendy

Użyj metody **ExecCommand**, aby wysyłać polecenia do serwera. Odpowiedzi będą dostępne przez zdarzenie **OnResponse**.

```
TsgcLib_RCON oRCON = new TsgcLib_RCON();
oRCON->RCON_Options->Host = "127.0.0.1";
oRCON->RCON_Options->Port = 25575;
oRCON->RCON_Options->Password = "test";
oRCON->Active = true;
```

```
void OnAuthenticate(TObject *Sender, bool Authenticated, const TsgcRCON_Packet *aPacket)
{
    if (Authenticated == true)
    {
```

```
        DoLog("#authenticated");
    }
    else
    {
        DoLog("#not authenticated");
    }
}
```

```
void OnResponse(Object *Sender, const string aResponse, const TsgcRCON_Packet *aPacket)
{
    DoLog(aResponse);
}
```

# CryptoHopper

## CryptoHopper

CryptoHopper to zautomatyzowany bot do handlu kryptowalutami, który umożliwia automatyzację handlu i zarządzania portfelem dla Bitcoin, Ethereum, Litecoin i innych.

## Konfiguracja

Wymagane jest **Konto dewelopera**; po jego zatwierdzeniu można przystąpić do tworzenia nowej aplikacji. API używa OAuth2 do uwierzytelniania, więc można pobrać **client\_id** i **client\_secret** z aplikacji.

```
TsgcHTTP_Cryptohopper oCryptoHopper = new TsgcHTTP_Cryptohopper();
oCryptoHopper->CryptoHopperOptions->OAuth2->ClientId = "client_id";
oCryptoHopper->CryptoHopperOptions->OAuth2->ClientSecret = "client_secret";
oCryptoHopper->CryptoHopperOptions->OAuth2->LocalIP = "127.0.0.1";
oCryptoHopper->CryptoHopperOptions->OAuth2->LocalPort = 8080;
oCryptoHopper->CryptoHopperOptions->OAuth2->Scope->Text = "read,notifications,manage,trade";
```

## Metody

CryptoHopper używa HTTPS jako protokołu do wysyłania żądań do API. Niektóre metody wymagają uwierzytelnienia (składanie zleceń, pobieranie danych użytkownika...), a inne są publiczne (np. pobieranie danych giełdy).

Funkcje zwracają odpowiedź CryptoHopper, a w przypadku wystąpienia błędu zostaje zgłoszony wyjątek.

## Hoppers

Zarządzanie podstawowymi operacjami bunkra.

|                     |                                                                  |
|---------------------|------------------------------------------------------------------|
| <b>GetHoppers</b>   | Pobierz Hoppery użytkowników.                                    |
| <b>CreateHopper</b> | <b>aBody:</b> tekst<br>JSON konfiguracji.<br>Tworzy nowy Hopper. |

|                     |                                                                             |                   |
|---------------------|-----------------------------------------------------------------------------|-------------------|
| <b>GetHopper</b>    | <b>aId:</b> identyfikator hoppera                                           | Retrieve Hopper   |
| <b>DeleteHopper</b> | <b>aId:</b> identyfikator hoppera                                           | Usuń Hopper       |
| <b>UpdateHopper</b> | <b>aId:</b> identyfikator hoppera<br><b>aBody:</b> tekst JSON konfiguracji. | Aktualizuj Hopper |

## Zamówienia

Zarządzaj zleceniami swojego Hoppera.

|                         |                                                                                                                                                                                            |                                                                                                                            |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>GetOpenOrders</b>    | <b>aId:</b> identyfikator hoppera                                                                                                                                                          | Pobierz wszystkie otwarte zlecenia hoppera.                                                                                |
| <b>CreateNewOrder</b>   | <b>aId:</b> identyfikator zasobnika<br><b>aOrder:</b> instancja TsgcHT-TPCTHOrder                                                                                                          | Utwórz nowe zlecenie kupna lub sprzedaży. W przypadku sprzedaży zaleca się użycie dedykowanego punktu końcowego sprzedaży. |
| <b>PlaceMarketOrder</b> | <b>aId:</b> identyfikator hoppera<br><b>aOrderSide:</b> cthosBuy lub cthosSell. <b>aCoin:</b> nazwa monety, przykład: EOS <b>aAmount:</b> wielkość zlecenia.                               | Złóż zlecenie rynkowe.                                                                                                     |
| <b>PlaceLimitOrder</b>  | <b>aId:</b> identyfikator hoppera<br><b>aOrderSide:</b> cthosBuy lub cthosSell. <b>aCoin:</b> nazwa monety, przykład: EOS <b>aAmount:</b> wielkość zlecenia.<br><b>aPrice:</b> cena limit. | Złóż zlecenie z limitem ceny                                                                                               |

|                        |                                                                              |                                                          |
|------------------------|------------------------------------------------------------------------------|----------------------------------------------------------|
| <b>DeleteOrder</b>     | <b>aId:</b> identyfikator hoppera<br><b>aOrderId:</b> identyfikator zlecenia | Usuwa zlecenie dla wybranego hoppera.                    |
| <b>DeleteAllOrders</b> | <b>aId:</b> identyfikator hoppera                                            | Usuwa wszystkie otwarte zlecenia dla wybranego hoppera.  |
| <b>GetOpenOrder</b>    | <b>aId:</b> identyfikator hoppera<br><b>aOrderId:</b> identyfikator zlecenia | Pobiera otwarte zlecenie w hopper według identyfikatora. |
| <b>CancelOrder</b>     | <b>aId:</b> identyfikator hoppera<br><b>aOrderId:</b> identyfikator zlecenia | Anuluj otwarte zlecenie.                                 |

## Pozycja

Zarządzaj pozycjami Hopper.

|                    |                                   |                                  |
|--------------------|-----------------------------------|----------------------------------|
| <b>GetPosition</b> | <b>aId:</b> identyfikator hoppera | Pobierz otwarte pozycje hoppera. |
|--------------------|-----------------------------------|----------------------------------|

## Transakcja

Historia transakcji z Twojego hopera.

|                            |                                                                                |                                                   |
|----------------------------|--------------------------------------------------------------------------------|---------------------------------------------------|
| <b>GetTradeHistory</b>     |                                                                                | Pobierz historię transakcji bota.                 |
| <b>GetTradeHistoryById</b> | <b>aId:</b> identyfikator hoppera<br><b>aTradeId:</b> identyfikator transakcji | Pobierz transakcję według identyfikatora hoppera. |

## Giełda

Pobieranie informacji o dostępnych giełdach w CryptoHopper

|                        |                                                                                                                          |                                                                              |
|------------------------|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| <b>GetExchange</b>     |                                                                                                                          | Pobierz wszystkie dostępne giełdy na platformie Cryptohopper.                |
| <b>GetAllTickers</b>   | <b>aExchange:</b> nazwa wymiany                                                                                          | Pobierz kurs dla wszystkich par                                              |
| <b>GetMarketTicker</b> | <b>aExchange:</b> nazwa giełdy<br><b>aPair:</b> nazwa pary<br><b>aExchange:</b> nazwa giełdy<br><b>aPair:</b> nazwa pary | Pobiera ticker dla pary handlowej.                                           |
| <b>GetOrderBook</b>    | <b>aDepth:</b> głębokość arkusza zleceń                                                                                  | Pobiera księgę zleceń dla wybranej giełdy, rynku i głębokości księgi zleceń. |

## Webhooki

Historia transakcji z Twojego hopera.

|                      |                                                                                                  |                                       |
|----------------------|--------------------------------------------------------------------------------------------------|---------------------------------------|
| <b>CreateWebhook</b> | <b>aURL:</b> adres URL webhooka<br><b>aMessageTypes:</b> typy wiadomości oddzielone przecinkami. | Aktualizowanie lub tworzenie webhooka |
| <b>DeleteWebhook</b> | <b>aURL:</b> adres URL webhooka                                                                  | Usuwa istniejący Webhook.             |



## Sygnały

Wysyłanie sygnałów do API CryptoHopper.

|                       |                                                                                       |                                      |
|-----------------------|---------------------------------------------------------------------------------------|--------------------------------------|
| <b>SendSignal</b>     | <b>aSignal:</b> klasa zawierająca wszystkie pola wymagane do wysłania sygnału.        | Wysyła sygnał                        |
| <b>SendTestSignal</b> | <b>aSignal:</b> klasa zawierająca wszystkie pola wymagane do wysłania sygnału.        | Wysyła sygnał testowy                |
| <b>GetSignalStats</b> | <b>aSignalId:</b> identyfikator sygnału. <b>aExchange:</b> opcjonalnie, nazwa giełdy. | Pobiera część statystyk sygnałowych. |

## Jak zaktualizować konfigurację Cryptohopper

Użyj metody UpdateHopper, aby zaktualizować konfigurację Hopper. Metoda jest przeciążona, więc można przekazać ciąg JSON lub użyć obiektu TsgcHTTPCTHopper i skorzystać z właściwości, aby włączyć lub wyłączyć właściwości Hopper.

```
public string EnableHopper()
{
    TsgcHTTPCTHopper *oHopper = new TsgcHTTPCTHopper();
    try
    {
        oHopper->Enabled = 1;
        result = Cryptohopper->UpdateHopper("1234", oHopper);
    }
    __finally
    {
        oHopper->Free();
    }
}
```

## Jak skonfigurować Webhook

Webhook umożliwia otrzymywanie powiadomień o zdarzeniach w hopperze. Webhooki wymagają publicznego serwera HTTPS nasłuchującego pod określonym adresem URL na wiadomości wysyłane przez

Cryptohopper. Serwer publiczny musi być chroniony certyfikatem SSL (certyfikaty z własnym podpisem nie są dozwolone).

Najpierw należy utworzyć webhook: skonfigurować właściwość Webhook klienta Cryptohopper, ustawiając Host i Port, na którym serwer będzie nasłuchiwał, a następnie skonfigurować certyfikat we właściwości SSLOptions.

**Przykład:** Publiczny adres IP będzie wynosił 1.1.1.1, a port nasłuchiwania 443. Certyfikat jest przechowywany jako plik PEM o nazwie sgc.pem bez hasła.

```
/* OAuth2 */
cryptohopper->CryptohopperOptions->OAuth2->ClientId = "client_id";
cryptohopper->CryptohopperOptions->OAuth2->ClientSecret = "client_secret";
cryptohopper->CryptohopperOptions->OAuth2->LocalIP = "127->0->0->1";
cryptohopper->CryptohopperOptions->OAuth2->LocalPort = 8080;
/* Webhook */
cryptohopper->CryptohopperOptions->Webhook->Enabled = True;
cryptohopper->CryptohopperOptions->Webhook->Host = "1.1.1.1";
cryptohopper->CryptohopperOptions->Webhook->Port = 443;
cryptohopper->CryptohopperOptions->Webhook->ValidationCode = "1234";
cryptohopper->CryptohopperOptions->Webhook->SSLOptions->CertFile = "sgc->pem";
cryptohopper->CryptohopperOptions->Webhook->SSLOptions->KeyFile = "sgc->pem";
cryptohopper->CryptohopperOptions->Webhook->SSLOptions->RootCertFile = "sgc->pem";
cryptohopper->CryptohopperOptions->Webhook->SSLOptions->Password = "";
cryptohopper->StartWebhook();
```

# RTCMultiConnection

## RTCMultiConnection

RTCMultiConnection to biblioteka JavaScript WebRTC przeznaczona do aplikacji peer-to-peer (udostępnianie ekranu, konferencje audio/wideo, udostępnianie plików, strumieniowanie mediów itp.)

## Konfiguracja

RTCMultiConnection wymaga serwera WebSocket do sygnalizacji, dlatego należy przypisać właściwość `server` komponentu RTCMultiConnection do serwera WebSocket (np. [TsgcWebSocketHTTPServer](#)). Poniżej znajdują się właściwości, które należy skonfigurować.

### Serwer

Host: publiczny adres IP lub nazwa DNS serwera WebSocket.

Port: port nasłuchiwania serwera WebSocket.

### IceServers

Jest to konfiguracja serwerów ICE (STUN/TURN) umożliwiająca komunikację między równorzędnymi punktami. Przykład:

```
[
{
  "urls": "stun:www.yourstun.com"},
{
  "urls": "turn:www.yourturn.com",
  "username": "user",
  "credential": "secret"
}
```

## VideoResolution

Tutaj można skonfigurować rozdzielczość wideo wideokonferencji; im wyższa rozdzielczość, tym większa przepustowość wymagana przez połączenie.

## HTMLDocuments

Dla każdej aplikacji należy skonfigurować nazwę strony HTML obsługującej tę zawartość.

Przykład: jeśli serwer działa na witrynie `www.webrtc.com` na porcie 8443, a właściwość `HTMLDocuments.VideoConferencing` = `/RTCMultiConnection-VideoConferencing.html`, adres URL do dostępu do VideoConferencing będzie następujący:

`https://www.webrtc.com:8443/RTCMultiConnection-VideoConferencing.html`

WebRTC wymaga bezpiecznego połączenia (HTTPS), dlatego konieczne jest użycie certyfikatów. Więcej informacji można znaleźć w sekcji [SSL serwera](#).

## Aplikacje

|                   |                                                                                   |
|-------------------|-----------------------------------------------------------------------------------|
| Wideokonferencje  | Wieloosobowy czat wideo (wiele do wielu) z wykorzystaniem modelu sieci mesh.      |
| ScreenSharing     | Udostępnianie ekranu dla wielu użytkowników (jeden do wielu) z topologią gwiazdy. |
| VideoBroadcasting | Wieloużytkownikowe (jeden do wielu) nadawanie wideo z topologią gwiazdową.        |

# WebPush

[RFC 8030](#)

[RFC 8291](#)

Protokół WebPush jest zdefiniowany przez **RFC 8030** (Delivery using HTTP Push) oraz **RFC 8291** (Message Encryption).

Web Push to **ustandaryzowany protokół** służący do **dostarczania powiadomień push do przeglądarek internetowych**. Wykorzystuje Push API, standardowe API webowe umożliwiające witrynom rejestrowanie i odbieranie wiadomości push. Push API pozwala witrynom wysyłać wiadomości push do przeglądarki użytkownika, nawet gdy użytkownik aktywnie nie przegląda danej witryny.

Aby korzystać z Web Push, witryna internetowa musi najpierw **uzyskać subskrypcję push od przeglądarki użytkownika**. Subskrypcja składa się z unikalnego adresu URL punktu końcowego oraz klucza szyfrowania. Adres URL punktu końcowego jest używany przez witrynę do wysyłania wiadomości push do przeglądarki użytkownika, natomiast klucz szyfrowania służy do szyfrowania i odszyfrowywania wiadomości push.

Po **uzyskaniu subskrypcji push** przez witrynę internetową może ona **wysyłać wiadomości push** do przeglądarki użytkownika poprzez żądanie HTTP do adresu URL punktu końcowego. Wiadomość push jest wysyłana w specjalnym formacie zwanym Web Push Protocol Message, który składa się z zestawu nagłówek i ładunku. Nagłówki zawierają informacje takie jak klucz szyfrowania i TTL (czas życia) wiadomości, natomiast ładunek zawiera właściwą treść wiadomości.

Gdy **przeglądarka użytkownika odbiera wiadomość push**, najpierw **odszyfrowuje wiadomość** przy użyciu klucza szyfrowania. Następnie **wyświetla powiadomienie użytkownikowi** wraz z ewentualnymi dodatkowymi akcjami, takimi jak odrzucenie powiadomienia lub otwarcie witryny internetowej.

Aby zapewnić bezpieczeństwo i prywatność wiadomości push, Web Push używa szyfrowania end-to-end i wymaga, aby subskrypcje push były uzyskiwane przez bezpieczne połączenie (np. HTTPS). Ponadto protokół zapewnia mechanizmy uwierzytelniania nadawcy wiadomości push i zapobiegania nadużyciom (np. poprzez ograniczenie liczby wiadomości push, które witryna może wysyłać do użytkownika).

## Komponenty

Istnieją 2 składniki obsługujące WebPush:

- **TsgcWSAPIServer\_WebPush:** implementuje protokół WebPush po stronie serwera, umożliwiając żądanie uprawnień od użytkowników, rejestrowanie subskrypcji, wysyłanie powiadomień i wiele więcej. Komponent zawiera już wbudowanego klienta WebPush do wysyłania powiadomień.
- **TsgcWebPush\_Client:** implementuje protokół WebPush po stronie klienta, umożliwiając wysyłanie powiadomień do użytkowników przez przeglądarkę na komputerze i urządzeniu mobilnym. Jest to przydatne, gdy posiadasz już klucze i punkt końcowy i chcesz jedynie publikować wiadomości WebPush do subskrybujących klientów.

# TsgcWSAPIServer\_WebPush

Składnik TsgcWSAPIServer\_WebPush.

## Wprowadzenie

**TsgcWSServer\_API\_WebPush** to komponent zapewniający funkcjonalność obsługi subskrypcji WebPush. WebPush to protokół dostarczania powiadomień w czasie rzeczywistym do aplikacji internetowych działających w przeglądarce. Komponent ten służy do zarządzania subskrypcjami i wysyłania powiadomień do subskrybowanych klientów. Poniżej znajdują się właściwości, zdarzenia i metody udostępniane przez klasę **TsgcWSServer\_API\_WebPush** wraz z przykładami kodu demonstrującymi ich użycie.

## Konfiguracja

1. Dołącz **TsgcWSServer\_API\_WebPush** do serwera WebSocket przy użyciu właściwości **Server**.
2. Skonfiguruj **klucze publiczny i prywatny** we właściwości **WebPush.VAPID**. (Zarejestrowani użytkownicy mogą pobrać plik wykonywalny generujący klucze VAPID dla systemu Windows).
3. Wymaga wdrożenia **wersji OpenSSL 3.0.0**
4. We właściwości **WebPush.Endpoints** można zdefiniować własne punkty końcowe do obsługi subskrypcji WebPush; domyślnie dostęp do punktu końcowego „/sgcWebPush.html” powoduje wyświetlenie prostej strony internetowej umożliwiającej subskrypcję powiadomień WebPush.
5. Uruchom serwer i uzyskaj dostęp do skonfigurowanego punktu końcowego, aby go przetestować.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 3 metody publiczne.
- Zdarzenia — 3 zdarzenia do obsługi.

# TsgcWebPush\_Client

TsgcWebPush\_Client to klasa umożliwiająca wysłanie powiadomienia po uzyskaniu szczegółów subskrypcji.

## Wprowadzenie

TsgcWebPush\_Client to klasa umożliwiająca wysłanie powiadomienia po uzyskaniu szczegółów subskrypcji.

Poniżej znajduje się przykład użycia klienta WebPush do wysłania powiadomienia dla danego punktu końcowego, klucza publicznego i sekretu uwierzytelniania z subskrypcji WebPush.

```
void SendWebPushNotification()
{
    TsgcHTTP_API_WebPush_PushSubscription* oSubscription = new TsgcHTTP_API_WebPush_PushSubscription();
    try
    {
        oSubscription->Endpoint = "endpoint";
        oSubscription->PublicKey = "public key";
        oSubscription->AuthSecret = "authentication secret";
        TsgcHTTP_API_WebPush_Client* oWebPush = new TsgcHTTP_API_WebPush_Client(NULL);
        try
        {
            oWebPush->VAPID->PEM->PrivateKey->Text = "private_key_pem";
            oWebPush->VAPID->DER->PrivateKey = "private_key";
            oWebPush->VAPID->DER->PublicKey = "public_key";
            oWebPush->SendNotification(oSubscription, "{\"title\": \"eSeGeCe Notification\", \"body\": \"Hello from eSeGeCe\"}");
        }
        __finally
        {
            oWebPush->Free();
        }
    }
    __finally
    {
        oSubscription->Free();
    }
}
```

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 1 publiczna metoda.
- Zdarzenia — 0 zdarzeń do obsługi.



# Rozszerzenia

Protokół WebSocket jest zaprojektowany z myślą o rozszerzalności. Klienci WebSocket mogą żądać rozszerzeń, a serwery WebSocket mogą akceptować niektóre lub wszystkie rozszerzenia żądane przez klientów.

Obsługiwane rozszerzenia:

1. [Deflate-Frame](#): kompresja ramek WebSocket.
2. [PerMessage-Deflate](#): kompresja wiadomości WebSocket.

## Rozszerzenia | PerMessage-Deflate

PerMessage jest rozszerzeniem protokołu WebSocket; jeśli rozszerzenie jest obsługiwane przez serwer i klienta, obaj mogą kompresować przesyłane wiadomości:

- Używa metody kompresji Deflate.
- Kompresja dotyczy wyłącznie danych aplikacji (ramki sterujące i nagłówki nie są uwzględniane).
- Serwer i klient mogą wybrać, które wiadomości będą kompresowane.

### Max Window Bits

To rozszerzenie umożliwia dostosowanie rozmiaru okna przesuwającego po stronie serwera i klienta stosowanego przez algorytm LZ77 (od 8 do 15). Im większa ta wartość, tym większe prawdopodobieństwo znalezienia i eliminacji duplikatów, ale wiąże się to z większym zużyciem pamięci i cykli procesora. Domyślna wartość to 15.

### Brak przejęcia kontekstu

Domyślnie poprzednie wiadomości są używane do kompresji i dekompresji. Jeśli wiadomości są podobne, poprawia to współczynnik kompresji. Jeśli opcja jest włączona, każda wiadomość jest kompresowana wyłącznie przy użyciu własnych danych. Domyślnie jest wyłączona.

### MemLevel

Wartość ta nie jest negocjowana między serwerem a klientem. Wartość 1 oznacza najmniejsze zużycie pamięci, ale spowalnia algorytm kompresji i zmniejsza współczynnik kompresji; wartość 9 oznacza największe zużycie pamięci i najlepszą wydajność. Domyślnie jest ustawiona na 1.

*\* Wersja Indy dostarczona z Rad Studio XE2 zgłasza wyjątek z powodu niezgodności wersji zlib z funkcjami inicjalizacji. Aby to naprawić, wystarczy zaktualizować wersję Indy do najnowszej.*

## Rozszerzenia | Deflate-Frame

Jest to rozszerzenie protokołu WebSocket umożliwiające kompresję ramek wysyłanych przy użyciu protokołu WebSocket. Jest obsługiwane przez przeglądarki oparte na WebKit, takie jak Chrome i Safari. Rozszerzenie to jest obsługiwane zarówno przez komponenty serwera, jak i klienta.

To rozszerzenie zostało wycofane.

*\* Wersja Indy dostarczona z Rad Studio XE2 zgłasza wyjątek z powodu niezgodności wersji zlib z funkcjami inicjalizacji. Aby to naprawić, wystarczy zaktualizować wersję Indy do najnowszej.*

# MCP

**MCP (Model Context Protocol)** to ustandaryzowany protokół zaprojektowany w celu definiowania i zarządzania **kontekstową wymianą informacji** między modelami językowymi a zewnętrznymi systemami.

Umożliwia to ustrukturyzowaną, bezpieczną i dynamiczną komunikację, która pozwala modelom AI **rozumieć, rozszerzać** oraz **wchodzić w interakcje** z zewnętrznymi narzędziami lub źródłami danych poza ich natywnym środowiskiem.

MCP zapewnia **jednolity interfejs** do przekazywania kontekstowych metadanych, umożliwiając modelom interpretację instrukcji, pobieranie danych i wykonywanie zadań w przewidywalny i interoperacyjny sposób, podobny w duchu do tego, jak *WebAuthn* standaryzuje przepływy uwierzytelniania.

## Cel

MCP zostało opracowane w celu rozwiązania problemów z **fragmentacją kontekstu i niespójnością integracji** między modelami, klientami i narzędziami.

Dzięki MCP dowolny zgodny klient może komunikować się z dowolnym modelem lub usługą zgodną z MCP za pośrednictwem wspólnej, dobrze zdefiniowanej struktury.

Typowe przypadki użycia obejmują:

- Dołączanie **metadanych kontekstowych** (sesja użytkownika, stan aplikacji, uprawnienia itp.) do zapytań modelu.
- Dynamiczna wymiana **definicji funkcji lub narzędzi**.
- Zarządzanie **interakcjami stanowymi** między sesjami modelu.
- Ustanawianie **bezpiecznej współpracy** między modelami LLM, zestawami SDK i usługami backendowymi.

## Komponenty

- **TsgcWSAPIServer\_MCP**: Komponent zapewnia proste, lecz zaawansowane rozwiązanie do implementacji serwera MCP z funkcjami takimi jak żądania Tools, Prompts i Resources, uwierzytelnianie, kontrola przepływu za pomocą zdarzeń i wiele więcej.
- **TsgcWSAPIClient\_MCP**: Komponent zapewnia proste, ale wydajne rozwiązanie do implementacji klienta MCP z takimi funkcjami jak żądania Tools, Prompts i Resources oraz innymi.

# TsgcWSAPIServer\_MCP

Komponent TsgcWSAPIServer\_MCP.

## Wprowadzenie

**TsgcWSAPIServer\_MCP** udostępnia Model Context Protocol (MCP) przez punkt końcowy serwera HTTP `sgcWebSockets`. Komponent łączy przychodzące żądania HTTP z silnikiem `TsgcAI_MCP_Server`, dzięki czemu klienci zgodne z MCP mogą negocjować sesje, wyliczać monity/zasoby/narzędzia i wywoływać narzędzia za pomocą wywołań w stylu JSON-RPC.

## Konfiguracja

- **Umieść komponenty:** Komponent `TsgcWSAPIServer_MCP` musi być dołączony do serwera HTTP, `TsgcWebSocketHTTP_Server` lub `TsgcWebSocketServer_HTTPAPI` za pomocą właściwości `Server`.
- **Wybierz punkt końcowy:** Dostosuj `EndpointOptions.Endpoint`, jeśli domyślna ścieżka `/mcp` powinna zostać zmieniona. Żądania, których dokument odpowiada tej ścieżce, są traktowane jako wywołania MCP; pozostałe przechodzą do obsługi zasobów nadrzędnego API.
- **Skonfiguruj ustawienia MCP:** Należy użyć `MCPOptions.ServerInfo` do nazewnictwa/wersjonowania serwera, `MCPOptions.SessionTimeout` do kontrolowania czasu życia sesji oraz `MCPOptions.AuthenticationOptions` do wymagania klucza API lub niestandardowego nagłówka HTTP przed przetworzeniem jakiegokolwiek wywołania JSON-RPC.
- **Rejestrowanie odpowiedzi, narzędzi, zasobów i szablonów zasobów:** Wypełnij listy `Tools`, `Prompts`, `Resources` i `ResourceTemplates` udostępniane przez komponent (tylko do odczytu).
- **Obsługa zdarzeń:** Należy podłączyć opublikowane zdarzenia opisane poniżej, aby wpiąć logikę biznesową w cykl życia sesji, rozwiązywanie monitów/zasobów lub wykonywanie narzędzi. Wyjątki można przechwytywać w celu dostosowania kodów odpowiedzi HTTP.

## Wbudowane punkty końcowe HTTP (bez konfiguracji)

Gdy komponent jest dołączony do serwera HTTP, następujące punkty końcowe i zachowania są serwowane automatycznie obok `/mcp`, bez konieczności włączania dodatkowych właściwości. Umożliwia to natychmiastową zgodność z internetowymi łącznikami MCP, takimi jak **claude.ai**, które wymagają OAuth 2.1 z dynamiczną rejestracją klientów przez CORS.

- **Obsługa CORS z odwzorowaniem origin.** Żądanie wstępne `OPTIONS` do dowolnej obsługiwanej ścieżki zwraca `204 No Content` z następującymi nagłówkami:
  - `Access-Control-Allow-Origin`: echo nagłówka `Origin` żądania lub `*` w przypadku jego braku.
  - `Access-Control-Allow-Methods`: `POST`, `GET`, `DELETE`, `OPTIONS`
  - `Access-Control-Allow-Headers`: `content-type`, `mcp-session-id`, `mcp-protocol-version`, `authorization`, `accept`
  - `Access-Control-Expose-Headers`: `mcp-session-id`

- Access-Control-Max-Age: 86400
- Strict-Transport-Security: max-age=31536000

Wszystkie odpowiedzi inne niż OPTIONS zawierają również Access-Control-Allow-Origin, Access-Control-Expose-Headers i Strict-Transport-Security.

- **401 Unauthorized z www-Authenticate.** Gdy MCPOptions.AuthenticationOptions.ApiKey (lub CustomHeader) jest włączony, a żądanie nie zawiera prawidłowych danych uwierzytelniających, serwer odpowiada kodem 401 z nagłówkiem w postaci:  
Bearer realm="MCP", resource\_metadata="https://<host>/.well-known/oauth-protected-resource"
- **Punkty końcowe odnajdywania i przepływu OAuth 2.1.** Następujące punkty końcowe są udostępniane na tym samym serwerze HTTP co /mcp, obsługując PKCE (S256), granty authorization-code i refresh-token oraz dynamiczną rejestrację klientów:

| Ścieżka                                     | Specyfikacja        | Zachowanie                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GET /.well-known/oauth-authorization-server | RFC 8414            | Metadane serwera autoryzacji w formacie JSON. Ogłasza PKCE (S256), granty authorization_code + refresh_token, metody uwierzytelniania klienta publicznego + client_secret_basic, dynamiczną rejestrację pod adresem /oauth/register oraz zakres mcp.                                                                                  |
| GET /.well-known/oauth-protected-resource   | RFC 9728            | JSON opisujący punkt końcowy MCP jako zasób chroniony i wskazujący na ten serwer jako serwer autoryzacji.                                                                                                                                                                                                                             |
| POST /oauth/register                        | RFC 7591 (DCR)      | Przyjmuje treść rejestracji klienta JSON (client_name, redirect_uris). Wydaje wygenerowany client_id (bez client_secret — klient publiczny). Zwraca 201 z odpowiedzią zawierającą informacje o zarejestrowanym kliencie.                                                                                                              |
| GET /oauth/authorize                        | RFC 6749 + RFC 7636 | Renderuje stronę zgody HTML. Wymagane parametry zapytania: client_id, redirect_uri, response_type=code, code_challenge, code_challenge_method=S256, state, scope.                                                                                                                                                                     |
| POST /oauth/authorize                       | RFC 6749            | Przetwarza formularz zgody. Użytkownik musi wprowadzić <b>tę samą wartość co MCPOptions.AuthenticationOptions.ApiKey.Value</b> , aby zatwierdzić. Po zatwierdzeniu przekierowuje do redirect_uri z ?code=...&state=... Po odmowie lub błędnym kluczu przekierowuje z ?error=access_denied&state=... lub ponownie renderuje formularz. |
| POST /oauth/token                           | RFC 6749 + RFC 7636 | Treść zakodowana w formularzu. Obsługuje grant_type=authorization_code (waliduje PKCE: BASE64URL(SHA256(code_verifier)) == code_challenge) i grant_type=refresh_token. Wydaje 1-godzinny token dostępu Bearer oraz token odświeżania. Kody autoryzacji wygasają po 60 sekundach i są jednorazowe.                                     |

- **Rozszerzona weryfikacja tokenu Bearer.** Gdy `MCPOptions.AuthenticationOptions.ApiKey.Enabled` ma wartość `True`, nagłówek `Authorization: Bearer <X>` w żądaniach `/mcp` jest akceptowany, jeżeli `<X>` pasuje **do** skonfigurowanego statycznego klucza `ApiKey.Value` **lub** do aktualnie ważnego tokenu dostępu wydanego przez OAuth (wydanego w ramach powyższego przepływu).
- **Pamięć podręczna w pamięci operacyjnej.** Klienci OAuth, kody i tokeny są przechowywane wyłącznie w pamięci. Są tracone po restarcie serwera.
- **Nadpisanie za pomocą `OnMCPHTTPRequest`.** Procedury obsługi podłączone do `OnMCPHTTPRequest` mogą skrócić dowolne z powyższych zachowań przez ustawienie `Handled := True`.

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 12 zdarzeń do obsłużenia.

# Serwer MCP | Sesje

Po pomyślnej inicjalizacji między serwerem a klientem tworzona jest nowa sesja. Sesja jest zawsze przesyłana w żądaniu i odpowiedzi między klientem a serwerem w celu identyfikacji sesji używanej w przepływie żądanie/odpowieź. Przy korzystaniu z transportu HTTP czas życia sesji obejmuje pojedyncze żądanie i odpowiedź HTTP, co oznacza, że po wysłaniu odpowiedzi sesja zostaje zamknięta.

Obiekt sesji jest przekazywany przy każdym zdarzeniu. Aby wiedzieć, kiedy sesja została utworzona lub usunięta, należy używać następujących zdarzeń:

- **OnMCPSessionNew:** zdarzenie jest wywoływane za każdym razem, gdy tworzony jest nowy identyfikator sesji.
- **OnMCPSessionEnd:** wywoływane po zakończeniu sesji.

## OnMCPSessionNew

Zdarzenie jest wywoływane przy tworzeniu nowej sesji. Parametr Session umożliwia dostęp do identyfikatora sesji, czasu jej wygaśnięcia i innych danych.

## OnMCPSessionEnd

Zdarzenie jest wywoływane po zakończeniu sesji.

```
void __fastcall TForm1::OnMCPSessionEnd(TObject *Sender, const TsgcAI_MCP_Session *aSession)
{
    printf("#session_end: %s\n", aSession->Id.c_str());
}
void __fastcall TForm1::OnMCPSessionNew(TObject *Sender, const TsgcAI_MCP_Session *aSession)
{
    printf("#session_new: " + aSession->Id);
}
```



## Serwer MCP | Narzędzia

**TsgcWSAPIServer\_MCP** jest komponentem wysokiego poziomu udostępniającym funkcje serwera Model Context Protocol (MCP) przez `sgcWebSockets`. Gdy **klienci MCP żądają narzędzi** zgodnie ze specyfikacją narzędzi serwera MCP, komponent ten orkiestruje bramę HTTP, cykl życia sesji i serializację JSON-RPC, dzięki czemu kod Delphi może skupić się na implementacji logiki biznesowej.

Przychodzące żądania HTTP trafiające do punktu końcowego MCP zdefiniowanego w **EndpointOptions.Endpoint** są przechwytywane przez **TsgcWSServer\_API\_MCP**. Komponent tworzy (lub wznowia) sesję MCP, przekazuje żądanie do bazowego silnika `TsgcAI_MCP_Server` i na końcu zapisuje odpowiedź JSON-RPC z powrotem do klienta HTTP, zachowując nagłówek `mcp-session-id` wymagany przez specyfikację.

### Lista narzędzi

Serwer przechowuje katalog narzędzi w pamięci w **TsgcAI\_MCP\_ToolsList**. Narzędzia są gwarantowanie unikalne pod względem nazwy, udostępniają opisy i zawierają opis danych wejściowych przypominający JSON Schema, który jest emitowany w odpowiedzi na metodę `tools.list` ze specyfikacji. Gdy klient wywołuje `tools.list`, `TsgcWSAPIServer_MCP` wczytuje żądanie, serializuje bieżący katalog i odsyła go z kodem statusu HTTP 200.

Przykładowy kod publikujący proste narzędzie kalkulatora:

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Tool* oTool;
    // Add a tool named "math.add"
    oTool = MCPServer->Tools->AddTool("math.add", "Adds two numbers");
    // Define input schema properties
    oTool->InputSchema->Properties->AddProperty("a", true, aimcpjtNumber, "Left operand");
    oTool->InputSchema->Properties->AddProperty("b", true, aimcpjtNumber, "Right operand");
}
```

Zgodna odpowiedź `tools.list` wysyłana do klienta będzie odzwierciedlać ładunek schematu MCP JSON. Na przykład:

```
{
  "jsonrpc": "2.0",
  "id": "42",
  "result": {
    "tools": [
      {
        "name": "math.add",
        "description": "Adds two numbers",
        "inputSchema": {
          "type": "object",
          "required": ["a", "b"],
```

```

        "properties": {
            "a": { "type": "number", "description": "Left operand" },
            "b": { "type": "number", "description": "Right operand" }
        }
    }
}
]
}
}

```

## Żądanie narzędzia

Gdy klient wyda żądanie JSON-RPC `tools.call`, **TsgcWSAPIServer\_MCP** hydratuje silnie typowany obiekt żądania (w tym nazwę narzędzia i podane argumenty) przed wywołaniem zdarzenia **OnMCPRequestTool**. Procedura obsługi wypełnia ładunek odpowiedzi, który jest następnie serializowany z powrotem do klienta wraz z kodem statusu HTTP sukcesu.

Typowa procedura obsługi wygląda następująco:

```

void __fastcall TMainForm::MCPServerMCPRequestTool(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_ToolsCall *ARequest,
    const TsgcAI_MCP_Response_ToolsCall *AResponse)
{
    double LA, LB;
    if (ARequest->Params->Name == "math.add")
    {
        LA = ARequest->Params->Arguments->Node["a"]->AsNumber;
        LB = ARequest->Params->Arguments->Node["b"]->AsNumber;
        AResponse->Result->Content->AddText(
            Format("Sum = %.2f", ARRAYOFCONST((LA + LB)))
        );
    }
    else
    {
        AResponse->Result->IsError = true;
    }
}

```

Wygenerowana odpowiedź JSON-RPC będzie zgodna ze schematem `tools.call` ze specyfikacji, opakowując jedną lub więcej części treści oraz opcjonalnie ustrukturyzowany ładunek. Ponieważ komponent części odpowiedzi i ustawia kod HTTP automatycznie, nie jest wymagane żadne dodatkowe okablowanie.

## Publiczny interfejs API

Kilka klas pomocniczych udostępnia katalog narzędzi i pomocniki odpowiedzi na zdarzenia. Poniższe listy zawierają podsumowanie metod typowo wywoływanych z kodu produkcyjnego.

## TsgcAI\_MCP\_ToolsList

- **AddTool(const Name, Description):** tworzy lub zastępuje wpis narzędzia i zwraca bazowy obiekt **TsgcAI\_MCP\_Tool**, umożliwiając konfigurację schematów i wartości domyślnych.
- **Clear:** usuwa wszystkie zarejestrowane narzędzia. Należy wywołać tę metodę przed odbudowaniem katalogu podczas ładowania konfiguracji z dysku.
- **Count / Items[Index]:** udostępniają zawartość listy do enumeracji, diagnostyki lub niestandardowej serializacji.

## TsgcAI\_MCP\_Tool

- **InputSchema.Properties.AddProperty(Name, Required, JsonType, Description):** dokumentuje ładunek wejściowy JSON, który zostanie ogłoszony przez **tools.list**.
- **InputSchema.Properties.Clear:** przebuduj schemat w locie, gdy aplikacja odświeża możliwości narzędzi w czasie wykonania.
- **Description:** właściwość do zapisu przechowująca czytelne dla człowieka podsumowanie udostępniane klientom.

## TsgcAI\_MCP\_Response\_ToolsCall

- **Result.Content.AddText(Value):** dołącza bloki tekstu do odpowiedzi narzędzia. Wielokrotne wywołania tworzą odpowiedzi wieloczęściowe.
- **Result.Content.AddImage(Data, MimeType):** dołącza bloki obrazów do odpowiedzi narzędzia. Kolejne wywołania generują odpowiedzi wieloczęściowe.
- **Result.Content.AddAudio(Data, MimeType):** dołącza bloki audio do odpowiedzi narzędzia. Wielokrotne wywołania generują odpowiedzi wieloczęściowe.
- **Result.Content.AddEmbeddedResource(Uri, Title, Text, MimeType):** dołącza bloki osadzonych zasobów do odpowiedzi narzędzia. Wielokrotne wywołania generują odpowiedzi wieloczęściowe.
- **Result.Content.AddResourceLink(Uri, Name, MimeType):** dołącza bloki łączy zasobów do odpowiedzi narzędzia. Wielokrotne wywołania generują odpowiedzi wieloczęściowe.
- **Result.Content.Clear:** resetuje wszystkie poprzednio wygenerowane dane wyjściowe przed dołączeniem nowej zawartości.
- **Result.IsError:** ustaw tę flagę na **True**, gdy wywołanie narzędzia nie może zostać zrealizowane, aby framework serializował obiekt błędu MCP.

## Przykład zaawansowany

Poniższy fragment kodu buduje narzędzie do wyszukiwania geograficznego, które weryfikuje dane wejściowe, zwraca wiele bloków tekstowych i ustawia właściwość **IsError** na **True**, gdy parametry są nieprawidłowe.

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Tool *tool = MCPServer->Tools->AddTool("geo.locate", "Returns coordinates for a city and country");
    tool->InputSchema->Properties->Clear();
}
```

```

    tool->InputSchema->Properties->AddProperty("city", true, aimcpjtString, "City to search");
    tool->InputSchema->Properties->AddProperty("country", false, aimcpjtString, "Optional ISO country code");
}
void __fastcall TMainForm::MCPServerMCPRequestTool(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_ToolsCall *ARequest,
    const TsgcAI_MCP_Response_ToolsCall *AResponse)
{
    if (!SameText(ARequest->Params->Name, "geo.locate"))
        return;
    UnicodeString city = ARequest->Params->Arguments->Node["city"]->AsString();
    UnicodeString country = ARequest->Params->Arguments->Node["country"]->AsString();
    AResponse->Result->Content->Clear();
    if (city.IsEmpty())
    {
        AResponse->Result->Content->AddText("City cannot be empty.");
        AResponse->Result->IsError = true;
        return;
    }
    AResponse->Result->Content->AddText("City: " + city);
    if (!country.IsEmpty())
        AResponse->Result->Content->AddText("Country: " + country);
    AResponse->Result->Content->AddText("Coordinates: " + LookupCityCoordinates(city, country));
}

```

# MCP Server | Prompts

W MCP *monity* to szablony wielokrotnego użytku lub przepływy pracy udostępniane przez serwer. Są kontrolowane przez użytkownika (klienci wybierają, który monit wywołać). Dwie główne metody to:

- **prompts/list:** Żąda listy wszystkich dostępnych definicji promptów udostępnionych przez serwer. Jest obsługiwane wewnętrznie przez komponent `TsgcWSAPIServer_MCP`.
- **prompts/get:** Pobiera treść i opcjonalne metadane konkretnego monitu, zazwyczaj sekwencję wiadomości opisującą sposób interakcji klienta. Zdarzenie **OnMCPRequestPrompt** jest wywoływane, gdy serwer odbiera nowe żądanie.

Ponadto, jeśli serwer ustawia `prompts.listChanged = true` w swoich zadeklarowanych możliwościach, może wysyłać powiadomienia takie jak `notifications/prompts/list_changed` w przypadku aktualizacji katalogu monitów.

Gdy klient wysyła żądanie `prompts/get`, przekazuje argumenty (jeśli istnieją) i oczekuje sekwencji wiadomości (w roli użytkownika lub asystenta) z blokami zawartości (zazwyczaj tekstowymi, ale mogącymi zawierać osadzone zasoby lub obrazy).

Jeśli wystąpi błąd (brak promptu, nieprawidłowe argumenty), serwer zwróci kody błędów JSON-RPC, takie jak `-32602 (Invalid params)` lub `-32603 (Internal error)`.

## Lista odpowiedzi

Serwer przechowuje w pamięci katalog promptów w obiekcie **`TsgcAI_MCP_PromptsList`**. Prompty gwarantują unikalność nazwy, udostępniają opisy i dostarczają opis wejścia w stylu JSON-Schema, który jest emitowany w odpowiedzi na metodę `prompts.list` ze specyfikacji. Gdy klient wywołuje metodę `prompts.list`, `TsgcWSAPIServer_MCP` ładuje żądanie, serializuje aktualny katalog i odsyła go ze statusem HTTP 200.

Przykładowy kod publikujący monit do przeglądu kodu:

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Prompt* oPrompt;
    // Clear all existing prompts
    MCPServer->Prompts->Clear();
    // Create a new prompt named "CodeReview"
    oPrompt = MCPServer->Prompts->AddPrompt(
        "CodeReview",
        "Asks the LLM to analyze code quality and suggest improvements"
```

```

);
// Add argument 'code'
oPrompt->Arguments->AddArgument(
    "code",
    "The code to review",
    true // required
);
}

```

## Prompt Request

Gdy klient wysyła żądanie JSON-RPC `prompts.get`, komponent **TsgcWSAPIServer\_MCP** wypełnia silnie typowany obiekt żądania (zawierający nazwę monitu i dostarczone argumenty) przed wywołaniem zdarzenia **OnMCPRequestPrompt**. Procedura obsługi wypełnia ładunek odpowiedzi, który jest następnie serializowany z powrotem do klienta wraz z kodem stanu HTTP sukcesu.

Typowa procedura obsługi wygląda następująco:

```

void __fastcall TFRMMCPServer::MCPServerMCPRequestPrompt(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_PromptsGet *aRequest,
    const TsgcAI_MCP_Response_PromptsGet *aResponse)
{
    if (aRequest->Params->Name == "CodeReview")
    {
        aResponse->Result->Description = "Code review prompt";
        aResponse->Result->Messages->AddText(
            "user",
            "Please review this Delphi code: ShowMessage('Hello World');"
        );
    }
}

```

## Publiczny interfejs API

Monity są modelowane za pomocą kilku klas pomocniczych. Poniższe metody są najczęściej używane podczas tworzenia szablonów monitów i komponowania odpowiedzi **prompts/get**.

### TsgcAI\_MCP\_PromptsList

- **AddPrompt(const Name, Description):** rejestruje nowy wpis monitu i zwraca obiekt **TsgcAI\_MCP\_Prompt**, umożliwiając konfigurację argumentów i wiadomości.
- **Clear:** czyści katalog; przydatne podczas uruchamiania, gdy prompty są wczytywane z plików konfiguracyjnych.
- **Count / Items[Index]:** zapewnia dostęp do zawartości listy na potrzeby diagnostyki, wizualizacji w interfejsie użytkownika lub ręcznej serializacji.

## TsgcAI\_MCP\_Prompt

- **Arguments.AddArgument(Name, Description, Required):** deklaruje dane wejściowe strukturalne akceptowane przez podpowiedź.
- **Arguments.Clear:** odbudowuje listę argumentów, gdy monit zmienia się w czasie wykonywania.
- **Messages.AddText(Role, Text):** wstępnie wprowadza tury konwersacji, które są odsyłane do klienta jako część definicji monitu.

## TsgcAI\_MCP\_Response\_PromptsGet

- **Result.Description:** krótkie, czytelne dla człowieka podsumowanie towarzyszące ładunkowi monitu.
- **Result.Messages.AddText(Role, Text):** dynamicznie wstrzykuje dodatkowe wiadomości podczas pobierania monitu.
- **Result.Messages.AddImage(Data, MimeType):** dołącza bloki obrazów do odpowiedzi na podpowiedź. Wielokrotne wywołania tworzą odpowiedzi wieloczęściowe
- **Result.Messages.AddAudio(Data, MimeType):** dołącza bloki audio do odpowiedzi na monit. Wielokrotne wywołania generują odpowiedzi wieloczęściowe.
- **Result.Messages.AddEmbeddedResource(Uri, Title, Text, MimeType):** dołącza bloki osadzonych zasobów do odpowiedzi na monit. Wielokrotne wywołania powodują odpowiedzi wieloczęściowe.
- **Result.Messages.Clear:** resetuje odpowiedź przed ponownym użyciem obiektu.

## Przykład zaawansowany

Poniższy przykład publikuje podpowiedź, która prowadzi LLM przez przepływ pracy triage. Przykład ilustruje opcjonalne argumenty oraz sposób zwracania wieloetapowych instrukcji w procedurze obsługi.

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Prompt *prompt;
    MCPServer->Prompts->Clear();
    prompt = MCPServer->Prompts->AddPrompt("IssueTriage", "Collect details before escalating support tickets");
    prompt->Arguments->Clear();
    prompt->Arguments->AddArgument("summary", "One-line description supplied by the user", true);
    prompt->Arguments->AddArgument("customerPriority", "Optional priority label", false);
    prompt->Messages->AddText("system", "You are a support assistant that triages technical issues.");
}

void __fastcall TMainForm::MCPServerMCPRequestPrompt(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_PromptsGet *aRequest,
    const TsgcAI_MCP_Response_PromptsGet *aResponse)
{
    if (!SameText(aRequest->Params->Name, "IssueTriage"))
        return;
    aResponse->Result->Description = "Guided checklist for support analysts.";
    aResponse->Result->Messages->Clear();
    aResponse->Result->Messages->AddText("user", "Review the ticket summary and ask clarifying questions.");
    aResponse->Result->Messages->AddText("assistant", "Acknowledge the request and confirm the escalation path.");
}
```

```
if (aRequest->Params->Arguments->Node["customerPriority"]->AsString() != "")  
    aResponse->Result->Messages->AddText("assistant", "Adjust SLA checks based on the provided customer priority.");  
}
```



# Serwer MCP | Zasoby

W MCP **zasoby** (resources) reprezentują adresowalne obiekty danych udostępniane przez serwer, takie jak pliki, rekordy bazy danych, generowane dokumenty lub dynamiczne dane wyjściowe API.

Są one *kontrolowane przez klienta*, co oznacza, że klient może zdecydować, który zasób chce pobrać i jak interpretować treść.

Dwie główne metody związane z tym procesem to:

- **resources/list:** Żąda listy wszystkich dostępnych definicji zasobów udostępnianych przez serwer. Jest obsługiwane wewnętrznie przez komponent `TsgcWSAPIServer_MCP`.
- **resources/read:** Pobiera zawartość i opcjonalne metadane określonego zasobu. Zdarzenie **OnMC-PRequestResource** jest wywoływane po otrzymaniu nowego żądania przez serwer. Zasób zazwyczaj zawiera:
  - **uri:** identyfikator zasobu.
  - **contentType:** typ zawartości (text/plain, application/json, image/png itp.).
  - **data or content:** właściwy ładunek danych (tekstowy lub binarny).

Ponadto, jeśli serwer w swoich zadeklarowanych możliwościach ustawia `resources.listChanged = true`, może wysyłać powiadomienia takie jak `notifications/resources/list_changed` w przypadku aktualizacji katalogu zasobów.

## Lista zasobów

Serwer przechowuje wewnętrzny katalog zasobów w pamięci w strukturze

**TsgcAI\_MCP\_ResourcesList**. Zasoby są gwarantowanie unikalne według identyfikatora uri, udostępniają opisy i zapewniają wejściowy URI z schematem podobnym do JSON Schema, emitowany w odpowiedzi na metodę `resources.list` specyfikacji. Gdy klient wywołuje `resources.list`, komponent **TsgcWSAPIServer\_MCP** ładuje żądanie, serializuje bieżący katalog i odsyła go ze statusem HTTP 200.

Przykładowy kod publikujący zasób plikowy:

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    // Clear any previously registered resources
    MCPServer->Resources->Clear();
    // Register a Rust source file resource
    MCPServer->Resources->AddResource(
```

```

        "file:///project/src/main.rs",          // URI
        "main.rs",                             // Name
        "Rust Software Application Main File", // Title
        "Primary application entry point",     // Description
        "text/x-rust"                          // MIME type
    );
}

```

## Żądanie zasobów

Gdy klient wyda żądanie JSON-RPC `resource.read`, **TsgcWSAPIServer\_MCP** tworzy silnie typowany obiekt żądania (zawierający zasób URI, nazwę i podane argumenty) przed wywołaniem zdarzenia **OnMCPRequestResource**. Procedura obsługi wypełnia ładunek odpowiedzi, który jest następnie serializowany z powrotem do klienta wraz z kodem statusu HTTP oznaczającym powodzenie.

Typowa procedura obsługi wygląda następująco:

```

void __fastcall TMainForm::OnMCPRequestResource(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_ResourcesRead *aRequest,
    const TsgcAI_MCP_Response_ResourcesRead *aResponse)
{
    if (aRequest->Params->Uri == "file:///project/src/main.rs")
    {
        aResponse->Result->Contents->AddContentText(
            "file:///project/src/main.rs",          // URI
            "main.rs",                             // Name
            "Rust Software Application Main File", // Description
            "text/x-rust",                          // MIME type
            "fn main() {\n    println!(\"Hello world!\");\n}" // Content
        );
    }
}

```

## Publiczny interfejs API

Zasoby łączą wpis katalogu (**TsgcAI\_MCP\_Resource**) z pomocnikami strumieniowania używanymi wewnątrz procedury **OnMCPRequestResource**. Poniżej przedstawiono metody najczęściej używane.

### TsgcAI\_MCP\_ResourcesList

- **AddResource(Uri, Name, Title, Description, MimeType):** rejestruje statyczny lub dynamicznie generowany zasób.
- **Clear:** usuwa każdą definicję zasobu, przydatne przed odbudowaniem listy z plików konfiguracyjnych.
- **Count / Items[Index]:** udostępnia katalog do wyliczania lub widoków diagnostycznych.

## TsgcAI\_MCP\_Resource

- **MimeType / Title / Description:** właściwości do zapisu opisujące ładunek zasobu zwracanego do klienta.
- **Name / Uri:** właściwości te służą do aktualizacji identyfikatorów bez ponownego rejestrowania wpisu zasobu.

## TsgcAI\_MCP\_Response\_ResourcesRead

- **Result.Contents.AddContentText(Uri, Name, Title, MimeType, Text):** wysyła ładunek tekstowy z powrotem do wywołującego.
- **Result.Contents.AddContentBlob(Uri, Name, Title, MimeType, Text):** wysyła ładunek tekstowy z powrotem do wywołującego.
- **Result.Contents.Clear:** resetuje wychodzący ładunek przed ponownym użyciem obiektu odpowiedzi.
- **Result.IsError:** oznacza odpowiedź jako błąd, gdy zasób nie może zostać wyprodukowany.

## Przykład zaawansowany

Poniższy przykład publikuje zasób JSON zwracający zagregowane metryki. Ilustruje sposób walidacji przychodzącego URI i tworzenia różnych odpowiedzi w zależności od żądania.

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    MCPServer->Resources->Clear();
    MCPServer->Resources->AddResource(
        "metrics://daily",
        "DailyMetrics",
        "Daily KPI snapshot",
        "Summaries generated each night",
        "application/json");
}

void __fastcall TMainForm::MCPServerMCPRequestResource(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_ResourcesRead *aRequest,
    const TsgcAI_MCP_Response_ResourcesRead *aResponse)
{
    UnicodeString targetUri = aRequest->Params->Uri.LowerCase();
    aResponse->Result->Contents->Clear();
    if (targetUri == "metrics://daily")
    {
        aResponse->Result->Contents->AddContentText(
            "metrics://daily",
            "DailyMetrics",
            "Daily KPI snapshot",
            "application/json",
            BuildMetricsPayload(Date()));
        return;
    }
    aResponse->Result->IsError = true;
    aResponse->Result->Contents->AddContentText(
```

```
aRequest->Params->Uri,  
"UnknownResource",  
"Not found",  
"text/plain",  
"Resource is not published by this server.");  
}
```

# Serwer MCP | Katalogi główne

Katalogi główne oznaczają lokalizacje systemu plików, do których eksplorowania klient zezwala serwerowi. Każdy wpis jest identyfikatorem URI `file://` z opcjonalną nazwą wyświetlaną, co umożliwia automatyzacji MCP respektowanie granic projektu przy jednoczesnym zachowaniu kontroli użytkownika nad tym, co jest udostępniane. Serwery przeszukują katalog, wysyłając żądanie `roots/list`, a gdy klient ogłosi `listChanged`, otrzymują powiadomienie `notifications/roots/list_changed`, dzięki czemu mogą odświeżyć widoki buforowane.

Ponieważ korzenie (`roots`) należą do klienta, serwer zawsze pełni rolę wnioskodawcy. Możliwość `roots` musi być obecna w ładunku inicjalizacyjnym klienta przed podjęciem próby wywołania `discovery`; w przeciwnym razie `RequestRootsList` zgłosi wyjątek `CapabilityNotSupported`.

## Żądanie katalogu głównego

`TsgcWSAPIServer_MCP` udostępnia metodę pomocniczą serializującą `roots/list` i przekazującą odpowiedź z powrotem do kodu użytkownika:

- **Pomocnik żądania:** wywołaj `RequestRootsList(Session)`, gdy sesja MCP ogłosi możliwość `roots`. Metoda automatycznie przypisuje identyfikator (o ile nie zostanie podany) i zachowuje oczekujące żądanie, aby odpowiedź mogła być skorelowana.
- **Zdarzenie odpowiedzi:** obsłuż `OnMCPResponseRootsList(Sender, Session, Request, Response)`. Zdarzenie jest wywoływane, gdy klient odpowiada z dostępnymi katalogami. Sprawdź `Response.Roots` lub zaloguj `Response.Write`, aby zbadać ładunek JSON.
- **Wykrywanie zmian:** jeśli klient wyśle później `notifications/roots/list_changed`, należy przesłać kolejne żądanie `RequestRootsList`, aby pamięć podręczna po stronie serwera pozostała zsynchronizowana.

## Przykładowy kod

Poniższe fragmenty kodu żądają zasobów głównych natychmiast po ustanowieniu sesji i rejestrują zwrócone wpisy.

```
void __fastcall TFRMMCPServer::FormCreate(TObject *Sender)
{
    MCPServer->OnMCPSessionNew = MCPServerSessionNew;
    MCPServer->OnMCPResponseRootsList = MCPServerRootsList;
}

void __fastcall TFRMMCPServer::MCPServerSessionNew(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession)
{
    if (ASession->ClientCapabilities->Roots->Enabled)
    {
        System::UnicodeString requestId = MCPServer->RequestRootsList(ASession);
        MemoLog->Lines->Add("roots/list sent id " + requestId);
    }
}

void __fastcall TFRMMCPServer::MCPServerRootsList(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_RootsList *ARequest,
    const TsgcAI_MCP_Response_RootsList *AResponse)
{
    MemoLog->Lines->Add(
```

```
        Format("roots/list returned %d entries", ARRAYOFCNST((AResponse->Roots->Count)))
    );
    MemoLog->Lines->Add("roots/list payload " + AResponse->Write());
}
```

# Serwer MCP | Próbkowanie

Sampling umożliwia serwerowi zlecenie klientowi wykonania generowania LLM w jego imieniu. Żądania opisują historię konwersacji, opcjonalny prompt systemowy, preferencje modelu i budżety tokenów. Klient zachowuje kontrolę nad tym, który dostawca jest używany, wykonuje wszelkie niezbędne przeglądy z udziałem człowieka i ostatecznie zwraca zatwierdzone uzupełnienie.

Klienci muszą zadeklarować możliwość sampling podczas initialize; w przeciwnym razie próba próbkowania zostanie odrzucona z błędem możliwości. Specyfikacja zachęca do uzyskania zgody użytkownika i zezwala na bloki treści tekstowych, obrazów lub audio w historii konwersacji.

## Wysyłanie żądania próbkowania

**TsgcWSAPIServer\_MCP** upraszcza przepływ sampling/createMessage za pomocą procedury pomocniczej i zdarzenia odpowiedzi:

- **Pomocnik żądania:** należy zbudować obiekt `TsgcAI_MCP_Request_SamplingCreateMessage`, wypełnić pola `Params.Messages`, `Params.ModelPreferences`, `Params.SystemPrompt` i opcjonalnie `Params.MaxTokens`, a następnie przekazać go do `RequestSamplingCreateMessage(Session, Request)`; metoda opatrzy wywołanie identyfikatorem (jeśli potrzeba) i zarejestruje oczekujące wywołanie.
- **Zdarzenie odpowiedzi:** obsłuż `OnMCPResponseSamplingCreateMessage(Sender, Session, Request, Response)`. Sprawdź `Response.Role`, `Response.Content`, `Response.Model` i `Response.StopReason`, aby kontynuować przepływ pracy.
- **Obsługa błędów:** odrzucone lub anulowane żądania przychodzą jako błędy JSON-RPC, dlatego należy opakować metodę pomocniczą w blok try/except i monitorować `OnMCPEXception` w celu rejestrowania.

## Przykładowy kod

Poniższe fragmenty kodu żądają krótkiego podsumowania od klienta i drukują odpowiedź asystenta.

```
void __fastcall TFRMMCPServer::FormCreate(TObject *Sender)
{
    MCPServer->OnMCPResponseSamplingCreateMessage = MCPServerSamplingResponse;
}

void __fastcall TFRMMCPServer::RequestTicketSummary(
    const TsgcAI_MCP_Session *ASession,
    const System::UnicodeString ATicketText)
{
    if (!ASession->ClientCapabilities->Sampling->Enabled)
        return;
    std::unique_ptr<TsgcAI_MCP_Request_SamplingCreateMessage> request(
        new TsgcAI_MCP_Request_SamplingCreateMessage());
    request->Params->Messages->AddTextMessage(
        "user",
        "Summarise the following ticket: " + ATicketText
    );
    request->Params->SystemPrompt = "You are a concise support assistant.";
    request->Params->ModelPreferences->Hints->AddHint("claude-3-sonnet");
    request->Params->ModelPreferences->SpeedPriority = 0.6;
    request->Params->ModelPreferences->IntelligencePriority = 0.8;
    request->Params->HasMaxTokens = true;
    request->Params->MaxTokens = 200;
    MemoLog->Lines->Add(
        "sampling/createMessage id " +
        MCPServer->RequestSamplingCreateMessage(ASession, request.get())
    );
}
```

```

    );
}
void __fastcall TFRMMCPServer::MCPServerSamplingResponse(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_SamplingCreateMessage *ARequest,
    const TsgcAI_MCP_Response_SamplingCreateMessage *AResponse)
{
    if (dynamic_cast <TsgcAI_MCP_Response_Result_Content_Text*>(AResponse->Content) != nullptr)
    {
        auto *text = static_cast <TsgcAI_MCP_Response_Result_Content_Text*>(AResponse->Content);
        MemoLog->Lines->Add("assistant: " + text->Text);
    }
    MemoLog->Lines->Add(
        "model: " + AResponse->Model + " reason: " + AResponse->StopReason
    );
}

```



# Serwer MCP | Elicitation

Elicitation umożliwia serwerowi żądanie od klienta zebrania ustrukturyzowanych danych wejściowych od użytkownika w trakcie przepływu pracy. Żądanie zawiera czytelny dla człowieka komunikat oraz ograniczony schemat JSON opisujący pola, które muszą zostać wypełnione. Klient renderuje odpowiedni formularz, waliduje odpowiedź lokalnie i zwraca informację, czy użytkownik zaakceptował, odrzucił czy anulował monit.

Klient musi ogłosić możliwość elicitation podczas initialize. Schematy są celowo ograniczone do płaskich obiektów z prymitywnymi właściwościami (string, number, boolean lub enum), aby ułatwić renderowanie i wykluczyć dane wrażliwe.

## Żądanie danych od użytkownika

**TsgcWSAPIServer\_MCP** dostarcza metodę pomocniczą dla elicitation/create i wywołuje zdarzenie odpowiedzi po zakończeniu interakcji przez użytkownika:

- **Pomocnik żądania:** utwórz `TsgcAI_MCP_Request_ElicitationCreate`, ustaw `Params._Message` i opisz oczekiwany ładunek przez `Params.RequestedSchema`. Wywołaj `RequestElicitationCreate(Session, Request)`, aby wysłać żądanie do klienta.
- **Zdarzenie odpowiedzi:** obsłuż `OnMCPResponseElicitationCreate(Sender, Session, Request, Response)`. Sprawdź `Response.Action` (accept, decline lub cancel) i przeanalizuj `Response.Content`, gdy użytkownik zaakceptował.
- **Walidacja:** odrzucone żądania są zwracane jako błędy JSON-RPC. Przed przetworzeniem zawartości należy zawsze sprawdzić `Action` i być przygotowanym na ponowne pytanie, jeśli użytkownik anulował operację.

## Przykładowy kod

Poniższe fragmenty kodu żądają informacji kontaktowych, czekają na odpowiedź i rejestrują wynikową akcję wraz z ładunkiem.

```
void __fastcall TFRMMCPServer::FormCreate(TObject *Sender)
{
    MCPServer->OnMCPResponseElicitationCreate = MCPServerElicitationResponse;
}

void __fastcall TFRMMCPServer::RequestContactDetails(
    const TsgcAI_MCP_Session *ASession)
{
    if (!ASession->ClientCapabilities->Elicitation->Enabled)
        return;
    std::unique_ptr<TsgcAI_MCP_Request_ElicitationCreate> request(
        new TsgcAI_MCP_Request_ElicitationCreate());
    request->Params->_Message = "Please confirm your contact information.";
    request->Params->RequestedSchema->Title = "Contact information";
    request->Params->RequestedSchema->Description =
        "Used to keep you updated about this ticket.";
    auto *name = request->Params->RequestedSchema->Properties->AddProperty("name");
    name->_Type = "string";
    name->Title = "Full name";
    name->Required = true;
    auto *email = request->Params->RequestedSchema->Properties->AddProperty("email");
    email->_Type = "string";
    email->Title = "Email address";
    email->Format = "email";
    email->Required = true;
    auto *updates = request->Params->RequestedSchema->Properties->AddProperty("updates");
```

```

updates->_Type = "boolean";
updates->Title = "Receive status updates";
updates->Description = "Tick to receive notifications when the ticket changes.";
updates->DefaultBoolean = false;
MemoLog->Lines->Add(
    "elicitation/create id " +
    MCPServer->RequestElicitationCreate(ASession, request.get())
);
}
void __fastcall TFRMMCPServer::MCPServerElicitationResponse(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_ElicitationCreate *ARequest,
    const TsgcAI_MCP_Response_ElicitationCreate *AResponse)
{
    MemoLog->Lines->Add("elicitation action: " + AResponse->Action);
    if (AResponse->Action.LowerCase() == "accept" && AResponse->Content != nullptr)
        MemoLog->Lines->Add("payload: " + AResponse->Content->Text);
}

```

# TsgcWSAPIClient\_MCP

Komponent TsgcWSAPIClient\_MCP.

## Wprowadzenie

**TsgcWSAPIClient\_MCP** implementuje konsumenta Model Context Protocol (MCP) na bazie komponentów sieciowych sgcWebSockets. Obsługuje negocjację sesji, marshalling żądań/odpowiedzi JSON-RPC oraz zdarzenia pomocnicze, dzięki czemu aplikacje Delphi i C++Builder mogą wykrywać prompty, zasoby i narzędzia udostępniane przez dowolny serwer zgodny z MCP.

## Konfiguracja

- **Upuść komponent:** umieść **TsgcWSAPIClient\_MCP** na formularzu lub module danych.
- **Wskazanie punktu końcowego MCP:** należy ustawić właściwość **MCPOptions.HttpOptions.URL** na adres URL HTTP serwera udostępniającego protokół (na przykład `https://localhost:5001/mcp`). W razie potrzeby należy skonfigurować TLS i rejestrowanie zdarzeń za pomocą zagnieżdżonych obiektów **TLSoptions** i **LogOptions**.
- **Opisz klienta:** wypełnij właściwość **MCPOptions.ClientInfo**, podając *Name* produktu, czytelny dla człowieka *Title* oraz semantyczny numer *Version*. Wartości te są przekazywane podczas uzgodnienia **initialize**.
- **Zabezpiecz sesję:** skonfiguruj właściwość **MCPOptions.AuthenticationOptions**, gdy serwer oczekuje klucza API lub niestandardowego nagłówka HTTP. Poświadczenia są dołączane do każdego żądania HTTP, łącznie z aktualizacjami strumieniowymi.
- **Dostrajanie keep-alive:** należy użyć właściwości **MCPOptions.HeartBeat**, aby włączyć automatyczny cykl ping i zdefiniować preferowany interwał (w sekundach). Komponent wewnętrznie wywołuje metodę **Ping** po wyzwoleniu wewnętrznego licznika czasu keep-alive.
- **Obsługa zdarzeń:** subskrybuj zdarzenia opisane poniżej, aby rejestrować odpowiedzi, akceptować/odrzucać sesje lub wykonywać dodatkowe przetwarzanie ładunków zwróconych przez serwer.

## Dokumentacja

- Właściwości — 2 właściwości dostępne w składniku.
- Metody — 12 metod publicznych.

- Zdarzenia — 12 zdarzeń do obsługi.

# MCP Client | Narzędzia

Komponent `TsgcWSAPIClient_MCP` udostępnia wywołania JSON-RPC zdefiniowane przez Model Context Protocol (MCP) w celu wyliczenia narzędzi oferowanych przez serwer oraz ich wywoływania. Każde żądanie musi być wysłane po pomyślnym wykonaniu `Initialize`, ponieważ każda metoda weryfikuje, czy sesja jest już w stanie *initialized*.

## Żądanie ListTools

Należy wywołać metodę **ListTools**, aby pobrać rejestr narzędzi aktualnie udostępnianych przez serwer MCP. Komponent automatycznie buduje żądanie JSON-RPC `tools/list` i wywołuje zdarzenie **OnMCPListTools** z przeanalizowaną odpowiedzią.

- **Żądanie:** `ListTools` (brak parametrów).
- **Event:** `OnMCPListTools(Sender, Request, Response)`.
- **Odpowiedź:** `TsgcAI_MCP_Response_ToolsList` z kolejką `Tools`. Każdy element jest obiektem `TsgcAI_MCP_Tool` udostępniającym właściwości `Name`, `Title`, `Description` oraz `InputSchema` narzędzia (typ schematu JSON wraz z nazwanymi właściwościami).

Należy użyć pomocników `Count` i `Item[Index]` kolejki lub metody `GetTool`, aby przeglądać katalog.

## Żądanie ToolsCall

Należy wywołać **RequestTool**(`Name`, `Arguments`), aby uruchomić jedno z opublikowanych narzędzi. Komponent kopiuje argumenty JSON do ładunku żądania i wywołuje **OnMCPResponseTool** po przeanalizowaniu odpowiedzi.

- **Request:** `RequestTool` z **Name** narzędzia i opcjonalnymi **Arguments** w formacie JSON.
- **Zdarzenie:** `OnMCPResponseTool(Sender, Request, Response)`.
- **Response:** `TsgcAI_MCP_Response_ToolsCall`. Należy sprawdzić `Result.IsError`, `Result.StructuredContent` i przejść przez `Result.Content` w poszukiwaniu ustrukturyzowanych fragmentów wyjściowych. Odpowiedzi narzędzi mogą zawierać tekst, obrazy, bloki audio, osadzone zasoby lub łącza do zasobów.

## Przykładowy kod

Poniższe fragmenty kodu pokazują, jak żądać listy narzędzi, wykonywać narzędzie i odczytywać wyniki.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPListTools = MCPListTools;
    MCP->OnMCPResponseTool = MCPResponseTool;
}
void __fastcall TForm1::LoadTools()
{
    MCP->ListTools();
}
void __fastcall TForm1::MCPListTools(TObject *Sender,
    const TsgcAI_MCP_Request_ToolsList &ARequest,
```

```

    const TsgcAI_MCP_Response_ToolsList &AResponse)
{
    for (int LIndex = 0; LIndex < AResponse.Tools->Count; LIndex++)
    {
        TsgcAI_MCP_Tool *LTool =
            static_cast<TsgcAI_MCP_Tool*>(AResponse.Tools->Item[LIndex]);
        Memo1->Lines->Add(Format("%s (%s)", ARRAYOFCONST((LTool->Name, LTool->Description))));
    }
}

void __fastcall TForm1::CallTemperatureTool()
{
    TsgcJSON *LArgs = new TsgcJSON(nullptr);
    try
    {
        LArgs->AddPair("city", "Barcelona");
        MCP->RequestTool("GetTemperature", LArgs);
    }
    __finally
    {
        delete LArgs;
    }
}

void __fastcall TForm1::MCPResponseTool(TObject *Sender,
    const TsgcAI_MCP_Request_ToolsCall &ARequest,
    const TsgcAI_MCP_Response_ToolsCall &AResponse)
{
    if (AResponse.Result->IsError)
        Memo1->Lines->Add("Tool returned an error payload");
    for (int LIndex = 0; LIndex < AResponse.Result->Content->Count; LIndex++)
    {
        TsgcAI_MCP_Response_Result_Content *LContent =
            static_cast<TsgcAI_MCP_Response_Result_Content*>(
                AResponse.Result->Content->Item[LIndex]);
        Memo1->Lines->Add(LContent->Write());
    }
}

```

# MCP Client | Prompts

Komponent klienta MCP oferuje dwie metody JSON-RPC związane z monitami dostarczonymi przez serwer: listowanie dostępnych szablonów monitów oraz żądanie wykonania monitu. Przed wykonaniem tych wywołań należy upewnić się, że klient został zainicjalizowany.

## Żądanie PromptsList

**ListPrompts** żąda bieżącego katalogu monitów. Po odebraniu odpowiedzi komponent wywołuje zdarzenie **OnMCPListPrompts**.

- **Request:** ListPrompts (brak parametrów).
- **Zdarzenie:** OnMCPListPrompts(Sender, Request, Response).
- **Odpowiedź:** TsgcAI\_MCP\_Response\_PromptsList udostępniający kolejkę Prompts. Każdy obiekt TsgcAI\_MCP\_Prompt zawiera pola Name, Title, Description oraz kolekcję Arguments (nazwa argumentu, opis i flaga wymagalności).

## Żądanie PromptsGet

**RequestPrompt**(Name, Arguments) wykonuje szablon monitu z podanymi argumentami JSON. Komponent wywołuje zdarzenie **OnMCPResponsePrompt**, gdy odpowiedź serwera będzie gotowa.

- **Żądanie:** RequestPrompt z nazwą promptu (**Name**) i opcjonalnymi argumentami JSON (**Arguments**).
- **Zdarzenie:** OnMCPResponsePrompt(Sender, Request, Response).
- **Response:** TsgcAI\_MCP\_Response\_PromptsGet. Należy sprawdzić Result.Description, aby uzyskać podsumowanie, i iterować po Result.Messages, aby odczytać każdy TsgcAI\_MCP\_Response\_PromptsGet\_Result\_Message. Wiadomości są polimorficzne — należy używać konkretnych typów pochodnych (...\_Text, ...\_Image, ...\_Audio, ...\_EmbeddedResource), aby uzyskać dostęp do podstawowej zawartości i adnotacji.

## Przykładowy kod

Poniższy przykład wczytuje listę promptów, wywołuje prompt i wyświetla zwrócone wiadomości.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPListPrompts = MCPListPrompts;
    MCP->OnMCPResponsePrompt = MCPResponsePrompt;
}
void __fastcall TForm1::LoadPrompts()
{
    MCP->ListPrompts();
}
void __fastcall TForm1::MCPListPrompts(TObject *Sender,
    const TsgcAI_MCP_Request_PromptsList &ARequest,
    const TsgcAI_MCP_Response_PromptsList &AResponse)
{
    for (int LIndex = 0; LIndex < AResponse.Prompts->Count; LIndex++)
    {
```

```

        TsgcAI_MCP_Prompt *LPrompt = (TsgcAI_MCP_Prompt*)AResponse.Prompts->Item[LIndex];
        Memo1->Lines->Add(Format("%s: %s", ARRAYOFCONST((LPrompt->Name, LPrompt->Description))));
    }
}
void __fastcall TForm1::ExecutePrompt()
{
    TsgcJSON *LArgs = new TsgcJSON(nullptr);
    try
    {
        LArgs->AddPair("code", "ShowMessage('Hello World')");
        MCP->RequestPrompt("CodeReview", LArgs);
    }
    __finally
    {
        delete LArgs;
    }
}
void __fastcall TForm1::MCPResponsePrompt(TObject *Sender,
const TsgcAI_MCP_Request_PromptsGet &ARequest,
const TsgcAI_MCP_Response_PromptsGet &AResponse)
{
    Memo1->Lines->Add("Prompt description: " + AResponse.Result->Description);
    for (int LIndex = 0; LIndex < AResponse.Result->Messages->Count; LIndex++)
    {
        TsgcAI_MCP_Response_PromptsGet_Result_Message *LMessage =
            (TsgcAI_MCP_Response_PromptsGet_Result_Message*)AResponse.Result->Messages->Item[LIndex];
        if (dynamic_cast<TsgcAI_MCP_Response_PromptsGet_Result_Message_Text*>(LMessage))
        {
            auto *MsgText = (TsgcAI_MCP_Response_PromptsGet_Result_Message_Text*)LMessage;
            Memo1->Lines->Add(LMessage->Role + ": " + MsgText->Content.Text);
        }
        else
        {
            Memo1->Lines->Add(LMessage->Role + ": " + LMessage->Write());
        }
    }
}
}

```



# Klient MCP | Zasoby

API zasobów MCP umożliwiają klientowi wykrywanie artefaktów po stronie serwera i pobieranie ich wartości za pomocą komponentu **TsgcWSAPIClient\_MCP** po zainicjowaniu sesji.

## Żądanie ResourcesList

Użyj metody **ListResources**, aby pobrać katalog zasobów. Komponent wywołuje zdarzenie **OnMCPListResources** po sparsowaniu odpowiedzi JSON-RPC.

- **Request:** ListResources (brak parametrów).
- **Zdarzenie:** OnMCPListResources(Sender, Request, Response).
- **Odpowiedź:** TsgcAI\_MCP\_Response\_ResourcesList udostępniająca kolejkę Resources. Każdy wpis TsgcAI\_MCP\_Resource zawiera Uri, Name, Title, Description i MimeType zasobu.

## Żądanie ResourcesRead

Wywołaj **RequestResource(Uri)**, aby pobrać określony zasób. Komponent wywołuje **OnMCPResponseResource** z zdekodowanym ładunkiem.

- **Request:** RequestResource z docelowym **Uri**.
- **Zdarzenie:** OnMCPResponseResource(Sender, Request, Response).
- **Odpowiedź:** TsgcAI\_MCP\_Response\_ResourcesRead. Należy odczytać Result.Contents, aby uzyskać dostęp do każdego bloku TsgcAI\_MCP\_Response\_Resources\_ReadResult\_Content, który zawiera pola Uri, Name, Title, MimeType oraz dane Text lub Blob.

## Przykładowy kod

Ten fragment kodu wyświetla listę zasobów i pobiera pojedynczy plik.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPListResources = MCPListResources;
    MCP->OnMCPResponseResource = MCPResponseResource;
}
void __fastcall TForm1::LoadResources()
{
    MCP->ListResources();
}
void __fastcall TForm1::MCPListResources(TObject *Sender,
const TsgcAI_MCP_Request_ResourcesList &ARequest,
const TsgcAI_MCP_Response_ResourcesList &AResponse)
{
    for (int LIndex = 0; LIndex < AResponse.Resources->Count; LIndex++)
    {
        TsgcAI_MCP_Resource *LResource =
            static_cast<TsgcAI_MCP_Resource*>(AResponse.Resources->Item[LIndex]);
        Memo1->Lines->Add(Format("%s -> %s", ARRAYOFCONST((LResource->Name, LResource->Uri))));
    }
}
void __fastcall TForm1::FetchResource(const String AUri)
```

```

{
    MCP->RequestResource(AUri);
}
void __fastcall TForm1::MCPResponseResource(TObject *Sender,
const TsgcAI_MCP_Request_ResourcesRead &ARequest,
const TsgcAI_MCP_Response_ResourcesRead &AResponse)
{
    for (int LIndex = 0; LIndex < AResponse.Result->Contents->Count; LIndex++)
    {
        TsgcAI_MCP_Response_Resources_ReadResult_Content *LContent =
            static_cast<TsgcAI_MCP_Response_Resources_ReadResult_Content*>(
                AResponse.Result->Contents->Item[LIndex]);
        if (LContent->Text != "")
            Memo1->Lines->Add(LContent->Uri + ": " + LContent->Text);
        else
            Memo1->Lines->Add(LContent->Uri + " (binary, mime=" + LContent->MimeType + ")");
    }
}

```

# Klient MCP | Korzenie

Specyfikacja MCP wprowadza możliwość **roots**, dzięki której klient może opisać lokalizacje systemu plików, które jest gotów udostępnić serwerowi. Każdy główny katalog jest identyfikatorem URI `file://` z opcjonalną nazwą wyświetlaną. Serwery odpytują katalog za pomocą `roots/list` i, jeśli jest to obsługiwane, otrzymują wiadomość `notifications/roots/list_changed` przy każdej zmianie wyboru. Ta wymiana pozwala narzędziom respektować granice projektów, obszary robocze i układy wielu repozytoriów, zachowując kontrolę użytkownika nad tym, do czego serwer ma dostęp.

## Omówienie korzeni

Korzenie pełnią rolę barier ograniczających zakres funkcji po stronie serwera. Klient ogłasza obsługę podczas wywołania `initialize`, włączając możliwość `roots` i opcjonalnie flagę `listChanged`, aby umożliwić przesyłanie dynamicznych aktualizacji. Typowy ładunek zwracany przez `roots/list` zawiera jeden lub więcej wpisów, z których każdy posiada pola `Uri` i `Name`, reprezentujących obszary robocze, repozytoria lub inne wydzielone foldery.

## Obsługa żądań `roots/list`

`TsgcWSAPIClient_MCP` analizuje przychodzące komunikaty JSON-RPC i wywołuje `OnMCPListRoots` za każdym razem, gdy serwer żąda bieżącego katalogu. Zdarzenie umożliwia synchronizację modelu przestrzeni roboczej utrzymywanego przez aplikację z odpowiedzią protokołu.

- **Żądanie:** `TsgcAI_MCP_Request_RootsList`. Wiadomość nie zawiera parametrów, ale dostarcza identyfikator JSON-RPC, który należy odwzorować w odpowiedzi.
- **Zdarzenie:** `OnMCPListRoots(Sender, Request, Response)`. Jest wywoływane po weryfikacji sesji przez komponent.
- **Odpowiedź:** `TsgcAI_MCP_Response_RootsList`. Należy wywołać metodę `Roots.AddRoot(Uri, Name)` (lub ręcznie zbudować instancję `TsgcAI_MCP_Root`), aby opisać każdą dostępną lokalizację. Komponent serializuje wypełniony obiekt po powrocie z procedury obsługi, dlatego lista musi odzwierciedlać bieżący stan. Do diagnostyki lub rejestrowania zdarzeń należy używać metody `Response.Write`.

Gdy aplikacja umożliwia użytkownikom dodawanie lub usuwanie przestrzeni roboczych w czasie działania, należy pamiętać o wysłaniu odpowiedniego powiadomienia `notifications/roots/list_changed`, aby serwery odświeżyły swój buforowany katalog.

## Przykładowy kod

Poniższe fragmenty kodu publikują dwa korzenie, gdy serwer o nie poprosi, i rejestrują ładunek JSON, który zostanie odesłany.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPListRoots = MCPListRoots;
}

void __fastcall TForm1::MCPListRoots(TObject *Sender,
    const TsgcAI_MCP_Request_RootsList *ARequest,
    const TsgcAI_MCP_Response_RootsList *AResponse)
```

```
{
    AResponse->Roots->Clear();
    AResponse->Roots->AddRoot("file:///c:/projects/app", "Application Workspace");
    AResponse->Roots->AddRoot("file:///c:/projects/libs", "Shared Libraries");
    MemoLog->Lines->Add("Roots response: " + AResponse->Write());
}
```

# Klient MCP | Próbkowanie

Próbkowanie umożliwia serwerom MCP delegowanie generowania przez model językowy do klienta bez udostępniania danych uwierzytelniających dostawcy. Serwer wysyła żądania `sampling/createMessage` zawierające stan konwersacji, preferencje modelu i opcjonalne ograniczenia, takie jak `maxTokens`. Klient może zaangażować recenzenta-człowieka, egzekwować zabezpieczenia i ostatecznie zwrócić odpowiedź zawierającą wybrany model, wygenerowaną treść i powód zatrzymania.

## Zrozumienie próbkowania

Gdy możliwość `sampling` jest ogłaszana podczas `initialize`, serwery mogą swobodnie żądać generowania w trakcie przepływu pracy. Żądania zawierają tablice wiadomości (bloki tekstowe, obrazkowe lub audio), `systemPrompt`, opcjonalne limity tokenów oraz strukturę `ModelPreferences` balansującą koszt, szybkość i inteligencję poprzez priorytety numeryczne i ciągi wskazówek. Klient odpowiada za mapowanie tych wskazówek na dostępne modele, monitorowanie użytkownika w razie potrzeby oraz filtrowanie niebezpiecznych uzupełnień przed udzieleniem odpowiedzi.

## Obsługa żądań `sampling/createMessage`

`TsgcWSAPIClient_MCP` udostępnia generacje inicjowane przez serwer poprzez `OnMCPSamplingCreateMessage`. Procedura obsługi powinna zbadać dostarczony kontekst, przekazać wywołanie do żadanego modelu (lub osoby weryfikującej) i wypełnić ładunek odpowiedzi.

- **Żądanie:** `TsgcAI_MCP_Request_SamplingCreateMessage`. Należy przejrzeć właściwość `Params.Messages`, aby odtworzyć dialog, `Params.ModelPreferences` w celu priorytyzacji wskazówek oraz `Params.SystemPrompt` dla zabezpieczeń.
- **Zdarzenie:** `OnMCPSamplingCreateMessage(Sender, Request, Response)`. Wywoływane po zweryfikowaniu przez komponent sesji MCP i deserializacji żądania.
- **Response:** `TsgcAI_MCP_Response_SamplingCreateMessage`. Podać Role asystenta, przypisać obiekt zawartości (tekst, obraz, audio lub łącze zasobu) i opcjonalnie zapisać konkretny Model oraz `StopReason`. Wywołać `Response.Write`, aby podejrzeć ładunek JSON przed jego wysłaniem.

## Przykładowy kod

Poniższe fragmenty kodu rejestrują liczbę przychodzących wiadomości, generują stałą odpowiedź oraz opatrują ją adnotacją dotyczącą modelu i powodu zatrzymania. W środowisku produkcyjnym należy zastąpić zakodowaną na stałe treść rzeczywistym wywołaniem modelu lub pętlą zatwierdzenia przez człowieka.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPSamplingCreateMessage = MCPSamplingCreateMessage;
}
void __fastcall TForm1::MCPSamplingCreateMessage(TObject *Sender,
    const TsgcAI_MCP_Request_SamplingCreateMessage *ARequest,
    const TsgcAI_MCP_Response_SamplingCreateMessage *AResponse)
{
    MemoLog->Lines->Add(Format("Sampling request with %d messages", ARRAYOFCONST((ARequest->Params->Messages->Count))));
    AResponse->Role = "assistant";
    auto *text = new TsgcAI_MCP_Response_Result_Content_Text();
    text->Text = "The capital of France is Paris.";
    AResponse->Content = text;
```

```
AResponse->Model = "claude-3-sonnet-20240307";  
AResponse->StopReason = "endTurn";  
MemoLog->Lines->Add("Sampling response: " + AResponse->Write());  
}
```

# MCP Client | Elicitation

Elicitation umożliwia serwerowi żądanie od użytkownika danych wejściowych o określonej strukturze za pośrednictwem klienta MCP. Żądanie zawiera czytelny dla człowieka wiadomość oraz ograniczony schemat JSON definiujący oczekiwane pola. Klienci zachowują kontrolę nad interakcją, weryfikują dostarczone dane i zwracają akcję wskazującą, czy użytkownik zaakceptował, odrzucił, czy anulował monit.

## Zrozumienie procesu wywoływania

Możliwość elicitation jest negocjowana podczas initialize. Po włączeniu serwery mogą wysyłać wywołania elicitation/create, które zawierają płaski schemat obiektu ograniczony do typów prymitywnych (string, number/integer, boolean i enum z opcjonalnymi nazwami wyświetlanymi). Każda właściwość może określać metadane wyświetlania, ograniczenia długości lub zakresu oraz obsługiwane formaty (email, uri, date, date-time). Użytkownicy powinni zawsze być informowani o tym, kto prosi o informacje, mieć możliwość edycji ładunku oraz możliwość odmowy lub anulowania żądania.

## Obsługa żądań elicitation/create

**TsgcWSAPIClient\_MCP** wywołuje zdarzenie **OnMCPElicitationCreate** za każdym razem, gdy serwer żąda dodatkowych danych wejściowych. Zdarzenie daje dostęp do komunikatu z monitem, żadanego schematu oraz pustego obiektu odpowiedzi, który można wypełnić na podstawie decyzji użytkownika.

- **Request:** TsgcAI\_MCP\_Request\_ElicitationCreate. Należy sprawdzić Params.Message, aby wyświetlić prompt, oraz Params.RequestedSchema.Properties, aby budować kontrolki interfejsu użytkownika lub walidację.
- **Zdarzenie:** OnMCPElicitationCreate(Sender, Request, Response). Wywoływane po deserializacji ładunku JSON-RPC przez klienta.
- **Odpowiedź:** TsgcAI\_MCP\_Response\_ElicitationCreate. Ustaw Action na accept, decline lub cancel. Przy akceptacji zapisz zebrane wartości do Content (pomocnik TsgcJSON), tak aby odpowiadały żadanemu schematowi. Użyj Response.Write, aby sprawdzić końcowy dokument JSON.

## Przykładowy kod

Poniższy przykład prosi użytkownika o podanie nazwy użytkownika GitHub. Jeśli operator poda wartość, odpowiedź zostanie zaakceptowana i pole zostanie zwrócone; w przeciwnym razie żądanie zostanie odrzucone.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MCP->OnMCPElicitationCreate = MCPElicitationCreate;
}

void __fastcall TForm1::MCPElicitationCreate(TObject *Sender,
const TsgcAI_MCP_Request_ElicitationCreate *ARequest,
const TsgcAI_MCP_Response_ElicitationCreate *AResponse)
{
    MemoLog->Lines->Add("Elicitation prompt: " + ARequest->Params->Message);
    String gitHub = InputBox("GitHub", "Provide your GitHub username", "");
    if (!gitHub.IsEmpty())
    {
        AResponse->Action = "accept";
        AResponse->Content->Clear();
        AResponse->Content->AddPair("name", gitHub);
    }
}
```

```
}  
else  
{  
    AResponse->Action = "decline";  
}  
MemoLog->Lines->Add("Elicitation response: " + AResponse->Write());  
}
```



# OpenAI

OpenAI to prywatne laboratorium badawcze, którego celem jest rozwijanie i ukierunkowywanie sztucznej inteligencji (AI) w sposób przynoszący korzyści całej ludzkości. OpenAI opracowało następujące projekty:

- **GPT-3:** Ten zaawansowany model językowy stanowi podstawę innych produktów OpenAI. Analizuje teksty generowane przez ludzi, aby nauczyć się samodzielnie generować podobne treści.
- **DALL-E i DALL-E 2:** Te generatywne platformy AI potrafią analizować tekstowe opisy obrazów, które użytkownicy chcą uzyskać, a następnie generować te obrazy dokładnie zgodnie z opisem.
- **CLIP:** CLIP to sieć neuronowa, która łączy treści wizualne i tekstowe w celu przewidywania najlepszych możliwych podpisów, najdokładniej opisujących dane wizualizacje. Ze względu na zdolność uczenia się z więcej niż jednego typu danych (zarówno obrazów, jak i tekstu), można ją zakwalifikować do kategorii multimodalnej AI.
- **ChatGPT:** ChatGPT jest obecnie najbardziej zaawansowanym chatbotem AI zaprojektowanym do generowania tekstu podobnego do ludzkiego i udzielania odpowiedzi na pytania użytkowników. Wytrenowany na dużych zbiorach danych, potrafi generować odpowiedzi w sposób zbliżony do człowieka.
- **Codex:** Codex był trenowany na miliardach wierszy kodu w różnych językach programowania, aby pomagać programistom w upraszczaniu procesów kodowania. Opiera się na technologii GPT-3, jednak zamiast generować tekst, generuje kod.
- **Whisper:** Whisper to narzędzie do automatycznego rozpoznawania mowy (ASR). Zostało wytrenowane na dużej ilości danych audio w celu rozpoznawania, transkrypcji i tłumaczenia mowy w około 100 różnych językach, w tym języka technicznego i różnych akcentów.

## OpenAI API

Interfejs API OpenAI można stosować do praktycznie każdego zadania związanego z rozumieniem lub generowaniem języka naturalnego, kodu lub obrazów. OpenAI oferuje szereg modeli o różnych poziomach możliwości, odpowiednich do różnych zadań, a także możliwość dostrajania własnych modeli niestandardowych. Modele te można wykorzystywać do wszystkiego — od generowania treści, przez semantyczne wyszukiwanie, po klasyfikację.

## Najczęstsze zastosowania

- **Uzupełnienie**
  - [Przykłady OpenAI Completion](#)
- **Chat**
  - [Przykłady OpenAI Chat](#)

- Edytuj
  - [Przykłady OpenAI Edit](#)
- Audio
  - [Przykłady transkrypcji i tłumaczenia OpenAI](#)
- Moderacja
  - [Przykłady moderacji OpenAI](#)
- RealTime
  - [Przykłady OpenAI RealTime](#)
- Odpowiedzi
  - [Przykłady odpowiedzi OpenAI](#)
- Mowa
  - [Przykłady mowy OpenAI](#)
- Fine-Tuning
  - [Przykłady OpenAI Fine-Tuning](#)
- Batch
  - [Przykłady OpenAI Batch](#)
- Uploads
  - [Przykłady OpenAI Uploads](#)

## Konfiguracja

### OpenAI

OpenAI API używa kluczy API do uwierzytelniania. Należy odwiedzić stronę [Klucze API](#), aby pobrać klucz API używany w żądaniach.

Klucz API jest informacją poufną — nie należy udostępniać go innym ani umieszczać w kodzie po stronie klienta (przeglądarki, aplikacje). Żądania produkcyjne muszą być kierowane przez własny serwer back-endowy, gdzie klucz API może być bezpiecznie ładowany ze zmiennej środowiskowej lub usługi zarządzania kluczami.

Ten **klucz API** musi być skonfigurowany we właściwości **OpenAIOptions.ApiKey** komponentu. Opcjonalnie, dla użytkowników należących do wielu organizacji, można ustawić organizację we właściwości **OpenAIOptions.Organization**, jeśli konto należy do organizacji.

Po skonfigurowaniu klucza API poniżej znajduje się lista dostępnych funkcji do interakcji z OpenAI API.

### Azure

Klient obsługuje usługi Microsoft Azure OpenAI, więc można używać konta Azure do interakcji z API Azure OpenAI. Aby skonfigurować klienta do pracy z Azure, należy wykonać poniższe kroki:

1. Skonfiguruj właściwość **OpenAIOptions.Provider** = oapvAzure
2. Ustaw wartości **ResourceName** i **DeploymentId** (wartości te można znaleźć na koncie Azure)
  1. **OpenAIOptions.AzureOptions.ResourceName** = <nazwa zasobu>.
  2. **OpenAIOptions.AzureOptions.DeploymentId** = <identyfikator wdrożenia>.
3. Ustaw klucz API swojego konta Azure
  1. **OpenAIOptions.ApiKey** = <azure api key>.

Należy pamiętać, że nie wszystkie metody OpenAI są obsługiwane przez Azure; obecnie obsługiwane są wyłącznie następujące metody:

1. Uzupełnienie
2. Uzupełnianie czatu

## Właściwości

### OpenAIOptions

- **ApiKey:** Klucz API służący do uwierzytelniania w OpenAI API.
- **Organization:** Opcjonalny identyfikator organizacji dla użytkowników należących do wielu organizacji.
- **Provider:** Wybierz dostawcę: oapvOpenAI (domyślnie) lub oapvAzure dla Microsoft Azure OpenAI Services.
- **AzureOptions:** Konfiguracja Azure OpenAI Services.
  - **ResourceName:** Nazwa zasobu Azure.
  - **DeploymentId:** Identyfikator wdrożenia Azure.
  - **APIVersion:** Wersja Azure API.
- **HttpOptions:** Ustawienia połączenia HTTP.
  - **ReadTimeout:** Limit czasu w milisekundach na odczyt odpowiedzi HTTP. Wartość domyślna to 0 (brak limitu czasu).
- **LogOptions:** Konfiguracja rejestrowania.
  - **Enabled:** Gdy ma wartość True, żądania i odpowiedzi HTTP są zapisywane do pliku.
  - **FileName:** Ścieżka pliku, do którego zapisywane są dane dziennika.
- **RetryOptions:** Konfiguracja automatycznego ponawiania nieudanych żądań API.
  - **Enabled:** Gdy True, nieudane żądania są automatycznie ponawiane.
  - **Retries:** Maksymalna liczba prób ponowienia.
  - **Wait:** Czas oczekiwania w milisekundach między próbami ponowienia.

## Modele

Wyświetla listę i opis różnych modeli dostępnych w API.

- **GetModels:** Wyświetla listę aktualnie dostępnych modeli wraz z podstawowymi informacjami o każdym z nich, takimi jak właściciel i dostępność.
- **GetModel:** Pobiera instancję modelu, dostarczając podstawowych informacji o modelu, takich jak właściciel i uprawnienia.
  - **Model:** Identyfikator modelu używanego w tym żądaniu

## Uzupełnienia (Completions)

Na podstawie podanego monitu model zwróci jedno lub więcej przewidywanych uzupełnień, a także może zwrócić prawdopodobieństwa alternatywnych tokenów na każdej pozycji.

- **CreateCompletion:** Tworzy uzupełnienie dla podanego monitu i parametrów
  - **Model:** Identyfikator modelu do użycia. Można skorzystać z API listy modeli, aby zobaczyć wszystkie dostępne modele, lub zapoznać się z przeglądem modeli zawierającym ich opisy.
  - **Prompt:** Monit, dla którego mają być generowane uzupełnienia.

## Czat

Dla danej konwersacji czatowej model zwraca odpowiedź w formie uzupełnienia czatu.

- **Model:** Identyfikator modelu do użycia. Należy wywołać GetModels, aby uzyskać listę wszystkich modeli obsługiwanych przez Chat API.
- **Message:** Wiadomość, dla której mają zostać wygenerowane uzupełnienia czatu.
- **Role:** domyślnie user, inne opcje to: system, assistant.

## Edycje

Na podstawie monitu i instrukcji model zwróci edytowaną wersję monitu.

- **CreateEdit:** Tworzy nową edycję dla podanych danych wejściowych, instrukcji i parametrów.
  - **Model:** Identyfikator modelu do użycia. Można używać modeli text-davinci-edit-001 lub code-davinci-edit-001 z tym punktem końcowym.
  - **Instruction:** Instrukcja informująca model, jak edytować prompt.
  - **Input:** (opcjonalnie) Tekst wejściowy używany jako punkt wyjścia do edycji.

## Obrazy

Na podstawie podanego promptu i/lub obrazu wejściowego model wygeneruje nowy obraz.

- **CreateImage:** Tworzy obraz na podstawie monitu.
  - **Prompt:** Tekstowy opis żądanych obrazów. Maksymalna długość wynosi 1000 znaków.
- **CreateImageEdit:** Tworzy edytowany lub rozszerzony obraz na podstawie oryginalnego obrazu i odpowiedzi.
  - **Image:** Obraz do edycji. Musi być prawidłowym plikiem PNG, mniejszym niż 4 MB i kwadratowym. Jeśli maska nie jest podana, obraz musi mieć przezroczystość, która będzie użyta jako maska.
  - **Prompt:** Tekstowy opis pożądanego obrazu lub obrazów. Maksymalna długość wynosi 1000 znaków.
- **CreateImageVariations:** Tworzy wariant podanego obrazu.
  - **Image:** Obraz stanowiący podstawę wariantu/wariantów. Musi być prawidłowym plikiem PNG, mniejszym niż 4 MB i kwadratowym.

## Osadzenia

Uzyskanie reprezentacji wektorowej danego wejścia, którą można łatwo wykorzystać w modelach i algorytmach uczenia maszynowego.

- **CreateEmbeddings:** Tworzy wektor osadzania reprezentujący tekst wejściowy.
  - **Model:** Identyfikator modelu do użycia.
  - **Input:** Tekst wejściowy, dla którego mają zostać wygenerowane osadzenia.

## Audio

Zamień dźwięk na tekst.

- **CreateTranscriptionFromFile:** Transkrybuje audio w języku wejściowym na podstawie nazwy pliku
  - **Model:** Identyfikator modelu do użycia. Aktualnie dostępny jest tylko whisper-1.
  - **Filename:** Plik audio do transkrypcji w jednym z następujących formatów: mp3, mp4, mpeg, mpga, m4a, wav lub webm.
- **CreateTranscription:** Nagrywa dźwięk przez X sekund i transkrybuje go.
  - **Model:** Identyfikator modelu do użycia. Aktualnie dostępny jest wyłącznie whisper-1.
  - **Time:** czas w milisekundach, domyślnie 10 sekund.
- **CreateTranslationFromFile:** Tłumaczy dźwięk na język angielski.
  - **Model:** Identyfikator modelu do użycia. Obecnie dostępny jest wyłącznie whisper-1.
  - **Filename:** Plik audio do przetłumaczenia w jednym z następujących formatów: mp3, mp4, mpeg, mpga, m4a, wav lub webm.
- **CreateTranslation:** Nagrywa dźwięk przez X sekund i tłumaczy go.

- **Model:** ID modelu do użycia. Aktualnie dostępny jest tylko whisper-1.
- **Time:** czas w milisekundach, domyślnie 10 sekund.

## Pliki

Pliki służą do przysyłania dokumentów, które mogą być używane z funkcjami takimi jak dostrajanie modelu (Fine-tuning).

- **ListFiles:** Zwraca listę plików należących do organizacji użytkownika.
- **UploadFile:** Przesyła plik zawierający dokumenty do wykorzystania w różnych punktach końcowych/funkcjach. Aktualnie łączny rozmiar wszystkich plików przesłanych przez jedną organizację może wynosić do 1 GB.
  - **Filename:** Nazwa pliku JSON Lines do przesłania. Jeśli cel jest ustawiony na "fine-tune", każdy wiersz jest rekordem JSON z polami "prompt" i "completion" reprezentującymi przykłady treningowe.
  - **Purpose:** Zamierzony cel przysyłanych dokumentów. Należy użyć wartości "fine-tune" dla dostrajania modelu.
- **DeleteFile:** Usuń plik.
  - **FileId:** Identyfikator pliku używanego w tym żądaniu
- **RetrieveFile:** Zwraca informacje o konkretnym pliku.
  - **FileId:** Identyfikator pliku używanego w tym żądaniu
- **RetrieveFileContent:** Zwraca zawartość wskazanego pliku.
  - **FileId:** Identyfikator pliku używanego w tym żądaniu.

## Fine-Tunes

Zarządzaj zadaniami dostrajania, aby dostosować model do własnych danych treningowych.

- **CreateFineTune:** Tworzy zadanie dostrajające wskazany model na podstawie podanego zestawu danych. Odpowiedź zawiera szczegóły kolejgowanego zadania, w tym jego status oraz nazwę dostrajonego modelu po zakończeniu.
  - **TrainingFile:** Identyfikator przesłanego pliku zawierającego dane treningowe.
- **ListFineTunes:** Wyświetl listę zadań dostrajania Twojej organizacji
- **RetrieveFineTune:** Pobiera informacje o zadaniu fine-tune.
  - **FineTuneId:** Identyfikator zadania fine-tune
- **CancelFineTune:** Natychmiast anuluje zadanie dostrajania modelu.
  - **FineTuneId:** Identyfikator zadania dostrajania modelu
- **ListFineTuneEvents:** Pobierz szczegółowe aktualizacje statusu zadania dostrajania.
  - **FineTuneId:** Identyfikator zadania dostrajania
- **DeleteFineTuneModel:** Usuwa dostrajony model. Wymagana rola właściciela w organizacji.
  - **Model:** Model do usunięcia.

## Moderacje

Na podstawie podanego tekstu wejściowego model określa, czy treść narusza zasady dotyczące zawartości OpenAI.

- **CreateModeration:** Klasyfikuje, czy tekst narusza politykę treści OpenAI
  - **Input:** Tekst wejściowy do klasyfikacji

## RealTime

OpenAI Realtime API umożliwia interakcje niskolatencyjne z obsługą wielu modalności, w tym konwersację mowy na mowę oraz transkrypcję w czasie rzeczywistym.

- **Transkrypcja:** Realtime API można wykorzystać wyłącznie do transkrypcji, zarówno przy wejściu z mikrofonu, jak i z pliku. Na przykład można go używać do generowania napisów lub transkrypcji w czasie rzeczywistym. W trybie tylko transkrypcji model nie generuje odpowiedzi.

## Asystenci

Twórz asystentów AI zdolnych do wywoływania modeli i używania narzędzi do wykonywania zadań.

- **CreateAssistant:** Tworzy asystenta z modelem i instrukcjami.
- **ListAssistants:** Zwraca listę asystentów.
- **RetrieveAssistant:** Pobiera asystenta na podstawie ID.
- **ModifyAssistant:** Modyfikuje istniejącego asystenta.
- **DeleteAssistant:** Usuwa asystenta.

## Wątki

Wątki reprezentują sesję konwersacji. Wiadomości są dodawane do wątków, które są następnie przetwarzane przez uruchomienia.

- **CreateThread:** Tworzy nowy wątek.
- **RetrieveThread:** Pobiera wątek na podstawie identyfikatora.
- **ModifyThread:** Modyfikuje wątek.
- **DeleteThread:** Usuwa wątek.

## Wiadomości wątku

Wiadomości są dodawane do wątków i zawierają treść konwersacji.

- **CreateMessage:** Tworzy wiadomość w wątku.
  - **ThreadId:** Identyfikator wątku.
- **ListMessages:** Zwraca listę wiadomości dla danego wątku.
- **RetrieveMessage:** Pobiera wiadomość na podstawie identyfikatora wątku i wiadomości.
- **ModifyMessage:** Modyfikuje wiadomość.
- **DeleteMessage:** Usuwa wiadomość z wątku.

## Runs

Uruchomienia (Runs) reprezentują wykonanie asystenta na wątku. Asystent używa swojej konfiguracji oraz wiadomości wątku do wykonywania zadań poprzez wywoływanie modeli i narzędzi.

- **CreateRun:** Tworzy przebieg dla wątku z określonym asystentem.
- **ListRuns:** Zwraca listę przebiegów należących do wątku.
- **RetrieveRun:** Pobiera przebieg na podstawie identyfikatora wątku i przebiegu.
- **ModifyRun:** Modyfikuje przebieg.
- **SubmitToolOutputsToRun:** Przesyła dane wyjściowe narzędzia dla uruchomienia wymagającego działania.
- **CancelRun:** Anuluje trwające uruchomienie.

## Kroki przebiegu

Kroki przebiegu reprezentują poszczególne etapy wykonania danego przebiegu.

- **ListRunSteps:** zwraca listę kroków wykonania należących do danego wykonania.
- **RetrieveRunSteps:** pobiera krok uruchomienia na podstawie wątku, uruchomienia i identyfikatora kroku.

## Vector Stores

Magazyny wektorów służą do przechowywania i wyszukiwania plików przy użyciu osadzeń na potrzeby generowania wspomaganego wyszukiwaniem (RAG).

- **CreateVectorStore:** Tworzy magazyn wektorów.
- **ListVectorStores:** Zwraca listę magazynów wektorów.
- **RetrieveVectorStore:** Pobiera magazyn wektorów według identyfikatora.
- **ModifyVectorStore:** Modyfikuje magazyn wektorów.
- **DeleteVectorStore:** Usuwa magazyn wektorów.



## Pliki Vector Store

Zarządzaj plikami w magazynach wektorów.

- **CreateVectorStoreFile:** Dołącza plik do magazynu wektorów.
- **ListVectorStoreFiles:** Zwraca listę plików w magazynie wektorów.
- **RetrieveVectorStoreFile:** Pobiera plik z magazynu wektorów.
- **DeleteVectorStoreFile:** usuwa plik z magazynu wektorów.

## Partie plików magazynu wektorów

Operacje wsadowe dodawania plików do magazynów wektorów.

- **CreateVectorStoreFileBatch:** Tworzy partię w celu dodania wielu plików do magazynu wektorów.
- **RetrieveVectorStoreFileBatch:** Pobiera partię plików według identyfikatora.
- **CancelVectorStoreFileBatch:** Anuluje trwającą partię plików.
- **ListVectorStoreFilesBatch:** Wyświetla listę plików w partii.

## Mowa

Generuje mowę z tekstu przy użyciu modeli zamiany tekstu na mowę (text-to-speech).

- **CreateSpeech:** Generuje dźwięk na podstawie tekstu wejściowego. Zwraca dane audio jako ciąg znaków lub zapisuje do strumienia odpowiedzi.
  - **Model:** Model TTS do użycia (np. tts-1, tts-1-hd).
  - **Input:** Tekst, dla którego ma zostać wygenerowany dźwięk.
  - **Voice:** Głos do użycia (alloy, echo, fable, onyx, nova, shimmer).

## Zadania dostrajania modelu

Zarządzaj zadaniami dostrajania (fine-tuning) w celu tworzenia spersonalizowanych modeli. Jest to nowsze API dostrajania zastępujące starszy punkt końcowy Fine-Tunes.

- **CreateFineTuningJob:** Tworzy zadanie dostrajania na podstawie pliku treningowego.
- **ListFineTuningJobs:** Wyświetla listę zadań dostrajania należących do organizacji.
- **RetrieveFineTuningJob:** Pobiera informacje o zadaniu dostrajania modelu.
- **CancelFineTuningJob:** Anuluje zadanie dostrajania modelu.
- **ListFineTuningJobEvents:** Wyświetla aktualizacje stanu zadania dostrajania modelu.
- **ListFineTuningJobCheckpoints:** wyświetla listę punktów kontrolnych zadania dostrajania.

## Odpowiedzi

Tworzenie i zarządzanie odpowiedziami modelu. Responses API obsługuje wieloturuowe rozmowy, użycie narzędzi i ustrukturyzowane dane wyjściowe.

- **CreateResponse:** Tworzy odpowiedź modelu.
- **RetrieveResponse:** Pobiera odpowiedź według identyfikatora.
- **DeleteResponse:** Usuwa odpowiedź.
- **CancelResponse:** Anuluje trwającą odpowiedź.
- **ListInputItems:** Wyświetla elementy wejściowe odpowiedzi.

## Partie

Tworzenie żądań wsadowych API i zarządzanie nimi do asynchronicznego przetwarzania po niższych kosztach.

- **CreateBatch:** Tworzy zadanie wsadowe.
- **RetrieveBatch:** Pobiera partię według identyfikatora.
- **ListBatches:** Wyświetla listę wszystkich partii.
- **CancelBatch:** Anuluje przetwarzanie wsadu w toku.

## Przesyłanie plików

Przesyłaj duże pliki w częściach. Przydatne w przypadku plików przekraczających standardowy limit rozmiaru przesłania.

- **CreateUpload:** Tworzy sesję przesłania.
- **AddUploadPart:** Dodaje część do przesłania.
- **CompleteUpload:** Kończy przesłanie wieloczęściowe i tworzy plik.
- **CancelUpload:** Anuluje trwające przesłanie.

# TsgcHTTP\_API\_OpenAI

Klient REST API OpenAI obejmujący: Chat Completions, Images, Audio (TTS/STT), Embeddings, Files, Fine-tuning, Moderation, Assistants, Threads, Runs, Messages i Vector Stores.

## Wprowadzenie

Klient REST API OpenAI obejmujący: Chat Completions, Images, Audio (TTS/STT), Embeddings, Files, Fine-tuning, Moderation, Assistants, Threads, Runs, Messages i Vector Stores.

## Wprowadzenie

Klient REST API OpenAI obejmujący: Chat Completions, Images, Audio (TTS/STT), Embeddings, Files, Fine-tuning, Moderation, Assistants, Threads, Runs, Messages i Vector Stores.

Należy skonfigurować właściwość **OpenAIOptions**, podając klucz API oraz opcjonalne ustawienia HTTP/ponawiania/logowania, a następnie wywoływać typizowane metody, takie jak `CreateMessage/``CreateChatCompletion`, przekazując obiekt żądania i otrzymując typizowaną klasę odpowiedzi.

Informacje ogólne na temat dostawcy zawiera przewodnik [OpenAI](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 78 publicznych metod.
- Zdarzenia — 2 zdarzenia do obsługi.

# OpenAI | Uzupełnianie

Na podstawie podanego polecenia model zwróci jedno lub więcej przewidywanych uzupełnień oraz opcjonalnie prawdopodobieństwa alternatywnych tokenów na każdej pozycji.

## Prosty przykład

Użyj modelu text-davinci-003, aby uzyskać przewidywane uzupełnienie.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";
ShowMessage(OpenAI->_CreateCompletion("text-davinci-003", "Say this is a test"));
```

## Zaawansowany przykład

Użyj modelu text-davinci-003, aby uzyskać przewidywane uzupełnienie z bardziej losowym wyjściem i wygenerować 2 uzupełnienia dla każdego promptu.

```
TsgcHTTP_OpenAI_JSON *OpenAI = new TsgcHTTP_OpenAI_JSON(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

TsgcOpenAIClass_Request_Completion *oRequest = new TsgcOpenAIClass_Request_Completion();
try
{
    oRequest->Model = "text-davinci-003";
    oRequest->Prompt = "Say this is a test";
    oRequest->Temperature = 1;
    oRequest->N = 2;
    TsgcOpenAIClass_Response_Completion *oResponse = OpenAI->CreateCompletion(oRequest);

    if (Length(oResponse->Choices) > 0)
    {
        ShowMessage(oResponse->Choices[0]->Text);
    }
}
__finally
{
    delete oRequest;
    delete oResponse;
}
```

# OpenAI | Czat

Dla danej konwersacji czatowej model zwraca odpowiedź w formie uzupełnienia czatu.

## Prosty przykład

Komunikuj się z ChatGPT wysyłając wiadomość Hello.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";
ShowMessage(OpenAI->_CreateChatCompletion("gpt-3.5-turbo", "Hello!"));
```

## Zaawansowany przykład

Użyj modelu gpt-3-5, aby rozmawiać z bardziej losowymi wynikami i generować 2 uzupełnienia dla każdego podpowiedzi.

```
TsgcHTTP_OpenAI_JSON *OpenAI = new TsgcHTTP_OpenAI_JSON(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

TsgcOpenAIClass_Request_ChatCompletion *oRequest = new TsgcOpenAIClass_Request_ChatCompletion();
try
{
    oRequest->Model = "gpt-3.5-turbo";
    TsgcOpenAIClass_Request_Completion_Message *oMessage = new TsgcOpenAIClass_Request_Completion_Message();
    oMessage->Role = "user";
    oMessage->Content = "Hello!";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;
    oRequest->Temperature = 1;
    oRequest->N = 2;
    TsgcOpenAIClass_Response_ChatCompletion *oResponse = OpenAI->CreateChatCompletion(oRequest);

    if (Length(oResponse->Choices) > 0)
    {
        ShowMessage(oResponse->Choices[0]->_Message->Content);
    }
}
__finally
{
    delete oRequest;
    delete oResponse;
}
```

# OpenAI | Edycja

Na podstawie promptu i instrukcji model zwróci zredagowaną wersję promptu.

## Prosty przykład

Poproś OpenAI o poprawienie błędów ortograficznych w podanym tekście.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";
ShowMessage(OpenAI->_CreateEdit("text-davinci-edit-001", "Fix the spelling mistakes", "What day of the wek is it?"));
```

## Zaawansowany przykład

Polecenie OpenAI poprawienia błędów ortograficznych w monicie, z bardziej losowymi wynikami i generowaniem 2 uzupełnień dla każdego monitu.

```
TsgcHTTP_OpenAI_JSON *OpenAI = new TsgcHTTP_OpenAI_JSON(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

TsgcOpenAIClass_Request_Edit *oRequest = new TsgcOpenAIClass_Request_Edit();
try
{
    oRequest->Model = "text-davinci-edit-001";
    oRequest->Input = "What day of the wek is it?";
    oRequest->Instruction = "Fix the spelling mistakes";
    oRequest->Temperature = 1;
    oRequest->N = 2;
    TsgcOpenAIClass_Response_Edit *oResponse = OpenAI->CreateEdit(oRequest);

    if (Length(oResponse->Choices) > 0)
    {
        ShowMessage(oResponse->Choices[0]->Text);
    }
}
__finally
{
    delete oRequest;
    delete oResponse;
}
```

# OpenAI | Audio

## Create Transcription

Transkrybuje dźwięk w języku wejściowym.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);  
OpenAI->OpenAIOptions->ApiKey = "API_KEY";  
  
ShowMessage(OpenAI->_CreateTranscriptionFromFile("whisper-1", "c:\\media\\audio.mp3"));
```

## Tworzenie tłumaczenia

Tłumaczy audio na język angielski.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);  
OpenAI->OpenAIOptions->ApiKey = "API_KEY";  
  
ShowMessage(OpenAI->_CreateTranslationFromFile("whisper-1", "c:\\media\\audio.mp3"));
```

# OpenAI | Moderacja

Na podstawie podanego tekstu wejściowego model klasyfikuje, czy narusza on politykę treści OpenAI.

## Prosty przykład

Moderuj następujący tekst

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

ShowMessage(OpenAI->_CreateModeration("I want to kill them."));
```

## Zaawansowany przykład

Moderuj następujący tekst, wybierając model.

```
TsgcHTTP_OpenAI_JSON *OpenAI = new TsgcHTTP_OpenAI_JSON(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

TsgcOpenAIClass_Request_Moderation *oRequest = new TsgcOpenAIClass_Request_Moderation();
try
{
    oRequest->Model = "text-moderation-latest";
    oRequest->Input = "I want to kill them.";
    TsgcOpenAIClass_Response_Moderation *oResponse = OpenAI->CreateModeration(oRequest);

    if (Length(oResponse->results) > 0)
    {
        ShowMessage(oResponse->results[0]->flagged);
    }
}
__finally
{
    delete oRequest;
    delete oResponse;
}
```



# OpenAI | RealTime

Realtime API można używać wyłącznie do transkrypcji, zarówno z wejścia mikrofonowego, jak i z pliku. Na przykład można go używać do generowania napisów lub transkryptów w czasie rzeczywistym. W trybie tylko transkrypcji model nie generuje odpowiedzi.

Aby korzystać z Realtime API do transkrypcji, należy utworzyć sesję transkrypcji, łącząc się przez WebSockets. Użyj komponentu [TsgcWSAPI\\_OpenAI](#) oraz [TsgcWebSocketClient](#), aby rozpocząć nową sesję transkrypcji.

Poniżej znajduje się przykład transkrypcji w czasie rzeczywistym z użyciem OpenAI API.

```
TsgcWebSocketClient *WSClient = new TsgcWebSocketClient(NULL);
TsgcAudioRecorderWave *oAudio = new TsgcAudioRecorderWave(NULL);
TsgcWSAPI_OpenAI *OpenAI = new TsgcWSAPI_OpenAI(NULL);
OpenAI->Client = WSClient;
OpenAI->AudioRecorder = oAudio;
OpenAI->OpenAIOptions->APIKey = "your-api-key-here";
OpenAI->OpenAIOptions->Method = sgcoaimTranscription;
OpenAI->OpenAIOptions->Provider = sgcoaipOpenAI;
OpenAI->InputAudio->Language = "en";
OpenAI->InputAudio->Model = "whisper-1";
void __fastcall TForm1::OnOpenAIAudioTranscriptionCompleted(TObject *Sender, TsgcWSOpenAIConversation_Item_Completed *aItem)
{
    Log("#transcription_completed: " + aItem->Transcript);
}
```

## Ręczne wysyłanie dźwięku

Komponent umożliwia ręczne wysyłanie plików audio. Można użyć metody `AppendInputAudioBuffer` i przekazać audio jako `TStream`. Format audio musi być 24 kHz mono PCM (obsługiwana jest wyłącznie częstotliwość próbkowania 24000).

# OpenAI | Responses

Responses API to podstawowe API do interakcji z modelami OpenAI. Obsługuje wejścia tekstowe i graficzne, korzystanie z narzędzi, strumieniowanie i strukturyzowane dane wyjściowe.

## Prosty przykład

Utwórz odpowiedź za pomocą Responses API.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";
ShowMessage(OpenAI->_CreateResponse("gpt-4o", "What is the capital of France?"));
```

## Metody

- **CreateResponse:** Tworzy odpowiedź modelu na podstawie modelu i danych wejściowych.
  - **Model:** Identyfikator modelu do użycia (np. gpt-4o, gpt-4o-mini).
  - **Input:** Tekst lub strukturalne dane wejściowe dla modelu.
- **RetrieveResponse:** Pobiera wcześniej utworzoną odpowiedź na podstawie jej identyfikatora.
  - **ResponseId:** Identyfikator odpowiedzi do pobrania.
- **DeleteResponse:** Usuwa odpowiedź według jej identyfikatora.
  - **ResponseId:** Identyfikator odpowiedzi do usunięcia.
- **CancelResponse:** Anuluje odpowiedź w trakcie przetwarzania.
  - **ResponseId:** Identyfikator odpowiedzi do anulowania.
- **ListInputItems:** Zwraca listę elementów wejściowych dla danej odpowiedzi.
  - **ResponseId:** Identyfikator odpowiedzi.

# OpenAI | Mowa

Generuje mowę z tekstu przy użyciu interfejsu API Text-to-Speech (TTS). Obsługuje wiele głosów i formatów wyjściowych.

## Prosty przykład

Generuj mowę z tekstu i zapisz do strumienia pliku.

```
OpenAI = new TsgcHTTP_API_OpenAI();
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

TFileStream *oStream = new TFileStream("output.mp3", fmCreate);
try
{
    OpenAI->_CreateSpeech("tts-1", "Hello, how are you?", "alloy", oStream);
}
__finally
{
    delete oStream;
}
```

## Metody

- **CreateSpeech:** Generuje audio z tekstu wejściowego.
  - **Model:** Identyfikator modelu do użycia (tts-1 lub tts-1-hd).
  - **Input:** Tekst, dla którego generowane jest audio. Maksymalnie 4096 znaków.
  - **Voice:** Głos do użycia (alloy, echo, fable, onyx, nova, shimmer).
  - **ResponseStream:** Strumień, do którego będą zapisywane dane audio.
  - **ResponseFormat:** (opcjonalnie) Format audio: mp3 (domyślnie), opus, aac lub flac.
  - **Speed:** (opcjonalnie) Szybkość generowanego audio (0,25 do 4,0, domyślnie 1,0).

# OpenAI | Fine-Tuning

Zarządzaj zadaniami dostrajania (fine-tuning), aby dostosować model do konkretnych danych treningowych. Dostrajanie pozwala uzyskać lepsze wyniki z modeli dzięki wyższej jakości odpowiedzi, możliwości trenowania na większej liczbie przykładów i oszczędnościom kosztów.

## Prosty przykład

Utwórz zadanie dostrajania i wylistuj istniejące zadania.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

// Create a fine-tuning job
ShowMessage(OpenAI->_CreateFineTuningJob("gpt-4o-mini-2024-07-18", "file-abc123"));

// List fine-tuning jobs
ShowMessage(OpenAI->_ListFineTuningJobs());
```

## Metody

- **CreateFineTuningJob:** Tworzy zadanie dostrajania, które rozpoczyna proces tworzenia nowego modelu na podstawie podanego zbioru danych.
  - **Model:** Nazwa modelu do dostrojenia.
  - **TrainingFile:** Identyfikator przesłanego pliku zawierającego dane treningowe.
- **ListFineTuningJobs:** Lista zadań fine-tuning organizacji.
- **RetrieveFineTuningJob:** Pobiera informacje o zadaniu dostrajania.
  - **JobId:** Identyfikator zadania dostrajania.
- **CancelFineTuningJob:** Natychmiastowe anulowanie zadania dostrajania.
  - **JobId:** Identyfikator zadania dostrajania do anulowania.
- **ListFineTuningJobEvents:** Pobierz aktualizacje stanu zadania dostrajania modelu.
  - **JobId:** Identyfikator zadania dostrajania modelu.
- **ListFineTuningJobCheckpoints:** Lista punktów kontrolnych zadania dostrajania.
  - **JobId:** Identyfikator zadania dostrajania.

# OpenAI | Batch

Tworzenie dużych partii żądań API do uruchomienia asynchronicznego. Batch API zwraca wyniki w ciągu 24 godzin z 50% rabatem.

## Prosty przykład

Utwórz zadanie wsadowe i pobierz jego status.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

// Create a batch
ShowMessage(OpenAI->_CreateBatch("file-abc123", "/v1/chat/completions"));

// List batches
ShowMessage(OpenAI->_ListBatches());
```

## Metody

- **CreateBatch:** Tworzy i wykonuje wsad z przesłanego pliku żądań.
  - **InputFileId:** Identyfikator przesłanego pliku zawierającego żądania dla wsadu.
  - **Endpoint:** Punkt końcowy używany dla wszystkich żądań w wsadzie (np. /v1/chat/completions).
  - **CompletionWindow:** (opcjonalne) Przedział czasu, w którym wsad powinien zostać przetworzony. Domyślnie 24h.
- **RetrieveBatch:** Pobiera partię według jej identyfikatora.
  - **BatchId:** Identyfikator partii do pobrania.
- **ListBatches:** Wyświetl listę partii swojej organizacji.
- **CancelBatch:** Anuluje przetwarzaną partię.
  - **BatchId:** Identyfikator partii do anulowania.

# OpenAI | Uploads

Przesyłanie dużych plików w częściach. Uploads API umożliwia przesyłanie plików większych niż 512 MB poprzez podzielenie ich na części i przesyłanie każdej części osobno.

## Prosty przykład

Utwórz przesłanie, dodaj część i zakończ je.

```
TsgcHTTP_API_OpenAI *OpenAI = new TsgcHTTP_API_OpenAI(NULL);
OpenAI->OpenAIOptions->ApiKey = "API_KEY";

// Create an upload
AnsiString vUploadResponse = OpenAI->_CreateUpload("training_data.jsonl",
    "fine-tune", 2147483648, "application/jsonl");

// Add a part
AnsiString vPartResponse = OpenAI->_AddUploadPart("upload_abc123", "part1.jsonl");

// Complete the upload
ShowMessage(OpenAI->_CompleteUpload("upload_abc123", "[\"part_abc123\"]"));
```

## Metody

- **CreateUpload:** Tworzy pośredni obiekt Upload w celu rozpoczęcia przesyłania wieloczęściowego.
  - **Filename:** Nazwa pliku do przesłania.
  - **Purpose:** Zamierzony cel przesłanego pliku (np. fine-tune).
  - **Bytes:** Liczba bajtów w pliku.
  - **MimeType:** Typ MIME pliku (np. application/jsonl).
- **AddUploadPart:** Dodaje część do obiektu Upload.
  - **UploadId:** Identyfikator przesłania.
  - **Filename:** Fragment bajtów dla tej części.
- **CompleteUpload:** Kończy przesłanie. Przesłany plik jest dostępny dopiero po jego ukończeniu.
  - **UploadId:** Identyfikator przesłania.
  - **PartIds:** Uporządkowana lista identyfikatorów części jako ciąg znaków JSON.
- **CancelUpload:** anuluje przesłanie. Po anulowaniu nie można dodawać kolejnych części.
  - **UploadId:** identyfikator przesłania do anulowania.

# Aplikacje OpenAI

## Przegląd

Korzystając z OpenAI API, można tworzyć szeroki zakres aplikacji. Oto kilka przykładów:

- **Chatboty i wirtualni asystenci:** Aplikacje zdolne do prowadzenia rozmów z ludźmi w naturalny, zbliżony do ludzkiego sposób. Mogą być wykorzystywane do obsługi klienta, obsługi zapytań oraz udostępniania informacji na stronie internetowej lub w aplikacji mobilnej.
- **Generowanie treści:** Aplikacje generujące tekst przypominający ludzki, takie jak artykuły, wpisy blogowe czy raporty. Na przykład GPT-3 może być wykorzystywany do automatyzacji tworzenia treści dla mediów społecznościowych, generowania kodu lub tworzenia treści przyjaznych dla SEO.
- **Usługi tłumaczeniowe:** Aplikacje umożliwiające tłumaczenie tekstu z jednego języka na inny.
- **Korepetycje i edukacja:** AI może być wykorzystywane do tworzenia spersonalizowanych doświadczeń edukacyjnych, pomocy w odrabianiu zadań domowych lub wyjaśniania złożonych pojęć prostym językiem.
- **Gry:** OpenAI może być używany do tworzenia wciągających i interaktywnych gier, szczególnie tych zawierających konwersacyjne postacie lub złożone narracje.
- **Analiza sentymentu:** Analizowanie i kategoryzowanie nastrojów wyrażonych w danych tekstowych może być przydatne w badaniach rynku, monitorowaniu marki i rozumieniu opinii klientów.
- **Spersonalizowane rekomendacje:** Na podstawie dotychczasowych zachowań i preferencji użytkowników AI może generować spersonalizowane rekomendacje produktów, usług lub treści.
- **Text Completion:** Uzupełnianie zdań użytkownika lub wspomaganie pisania w klientach poczty e-mail lub oprogramowaniu do edycji tekstu.
- **Rozpoznawanie mowy:** Transkrypcja mówionego języka na tekst pisany, przydatna w usługach transkrypcji, asystentach głosowych i innych zastosowaniach.
- **Doradztwo medyczne i prawne:** Choć modele AI nie są w stanie zastąpić profesjonalnego doradztwa, mogą udzielać wstępnych wskazówek lub sugestii na podstawie podanych danych wejściowych.

## Komponenty

Poniżej znajduje się lista dostępnych komponentów wraz z krótkim opisem.

- **TsgcAIOpenAIAssistant:** Asystent posiada instrukcje i może korzystać z modeli, narzędzi i plików, aby odpowiadać na zapytania użytkowników.
- **TsgcAIOpenAIChatBot:** ChatBot, który nasłuchuje mowy i konwertuje ją na tekst przy użyciu API OpenAI Whisper. Tekst ten jest wysyłany do API ChatCompletion, które dostarcza odpowiedź od OpenAI, a ta odpowiedź jest konwertowana z tekstu na mowę.
- **TsgcAIOpenAITranslator:** aplikacja tłumacząca, która umożliwia tłumaczenie mowy w dowolnym języku na język angielski i odsłuchiwanie tłumaczenia przy użyciu dowolnego z dostępnych składników TextToSpeech.

- **TsgcAIOpenAIEmbeddings:** komponent służący do tworzenia aplikacji AI z niestandardowymi danymi, na przykład chatbot oparty na danych produktowych. Obsługiwane są następujące bazy danych:
  - **TsgcAIDatabaseVectorPinecone:** obsługuje bazę danych wektorowych Pinecone.

Poniższe komponenty służą do przechwytywania dźwięku z mikrofonu, odtwarzania plików audio oraz konwersji tekstu na mowę.

- **TsgcAudioRecorderMCI:** (tylko dla systemu Windows) komponent umożliwiający dostęp do mikrofonu i konwersję mowy do pliku wave.
- **TsgcAudioPlayerMCI:** (tylko dla systemu Windows) komponent umożliwiający odtwarzanie pliku mp3.
- **TsgcTextToSpeechSystem:** (tylko dla systemu Windows) konwertuje tekst na mowę bez konieczności używania zewnętrznego pliku mp3.
- **TsgcTextToSpeechGoogle:** konwertuje tekst na mowę przy użyciu dowolnego z dostępnych głosów Google Cloud.
- **TsgcTextToSpeechAmazon:** konwertuje tekst na mowę przy użyciu dowolnego głosu Amazon AWS.



# TsgcAIOpenAIAssistant

Assistants API umożliwia budowanie asystentów AI w ramach własnych aplikacji.

## Wprowadzenie

API asystentów umożliwia tworzenie asystentów AI w ramach własnych aplikacji. Asystent posiada instrukcje i może korzystać z modeli, narzędzi i plików, aby odpowiadać na zapytania użytkowników. API asystentów obsługuje aktualnie trzy typy narzędzi: interpreter kodu, wyszukiwanie plików oraz wywołanie funkcji.

## Przegląd

Typowy przepływ integracji z Assistants API wygląda następująco:

- Utwórz asystenta, definiując niestandardowe instrukcje i wybierając model. W razie potrzeby można dodać pliki i włączyć narzędzia takie jak Code Interpreter, File Search i Function calling.
- Utwórz wątek, gdy użytkownik rozpoczyna rozmowę.
- Dodawaj wiadomości do wątku w miarę jak użytkownik zadaje pytania.
- Uruchom asystenta na wątku, aby wygenerować odpowiedź przez wywołanie modelu i narzędzi.

### Krok 1: Utwórz asystenta

Asystent reprezentuje podmiot, który można skonfigurować do odpowiadania na wiadomości użytkownika przy użyciu kilku parametrów, takich jak model, instrukcje i narzędzia.

```
// Create a new assistant
TsgcAIOpenAIAssistant *oAssistant = new TsgcAIOpenAIAssistant(NULL);
// Set your API key
oAssistant->OpenAIOptions->ApiKey = txtAPIKey->Text;
// Assistant options
oAssistant->AssistantOptions->Name = "Math Tutor";
oAssistant->AssistantOptions->Instructions->Text = "You are a personal math tutor. Write and run code to answer math questions";
oAssistant->AssistantOptions->Model = "gpt-4o";
// Create the assistant
oAssistant->CreateAssistant();
```

### Krok 2: Utwórz wątek

Wątek (Thread) reprezentuje rozmowę między użytkownikiem a jednym lub wieloma asystentami. Wątek można utworzyć w momencie, gdy użytkownik (lub aplikacja AI) rozpoczyna rozmowę z asystentem.

```
oThread = oAssistant->CreateThread();
```

### Krok 3: Dodaj wiadomość do wątku i uruchom

Treść wiadomości tworzonych przez użytkowników lub aplikacje jest dodawana jako obiekty Message do wątku. Wiadomości mogą zawierać zarówno tekst, jak i pliki. Nie ma ograniczeń co do liczby wiadomości, które można dodać do wątków — kontekst, który nie mieści się w oknie kontekstowym modelu, jest inteligentnie przycinany.

Po dodaniu wszystkich wiadomości użytkownika do wątku można uruchomić wątek z dowolnym asystentem. Utworzenie przebiegu powoduje użycie modelu i narzędzi powiązanych z asystentem w celu wygenerowania odpowiedzi. Odpowiedzi te są dodawane do wątku jako wiadomości asystenta.

```
void SendMessage(TsgcAIOpenAIAssistant *oAssistant, TsgcAIClass_Thread *oThread, const UnicodeString aMessage)
{
    int i;
    TsgcOpenAIClass_Message *oMessage;
    TsgcOpenAIClass_Response_List_Messages *oMessages;
    TsgcOpenAIClass_Run *oRun;
    DoLog(L"[user]: " + aMessage);
    oMessage = oAssistant->CreateMessageText(oThread->Id, aMessage);
    if (oMessage != NULL)
    {
        oRun = oAssistant->CreateRunAndWait(oThread->Id);
        if (oRun != NULL)
        {
            oMessages = oAssistant->GetMessages(oThread->Id, oRun->Id);
            if (oMessages != NULL && oMessages->Messages.Length > 0)
            {
                for (i = 0; i < oMessages->Messages.Length; i++)
                {
                    DoLog(L"[assistant]: " + DoFormatResponse(oMessages->Messages[i]->ContentText + L"\r\n"));
                }
            }
        }
    }
}
```

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 8 zdarzeń do obsługi.

# TsgcAIOpenAIAssistant | File Search

Wyszukiwanie plików (File Search) rozszerza możliwości asystenta, integrując wiedzę z zewnętrznych źródeł, takich jak zastrzeżone informacje o produkcie lub dokumenty dostarczone przez użytkowników. Oto jak to działa:

- **Przetwarzanie dokumentów:** OpenAI automatycznie przetwarza i segmentuje (lub „dzieli na fragmenty”) dokumenty na łatwe do zarządzania części informacji.
- **Tworzenie osadzeń:** Te fragmenty są przekształcane w osadzenia, czyli matematyczne reprezentacje treści, które uchwycają ich znaczenie i kontekst.
- **Przechowywanie i wyszukiwanie:** System przechowuje te osadzenia, umożliwiając zarówno wyszukiwanie wektorowe (w celu znalezienia semantycznie podobnych treści), jak i wyszukiwanie słów kluczowych (w celu lokalizowania dokładnych dopasowań).

Ta kombinacja zapewnia, że asystent może pobierać i wykorzystywać odpowiednie treści, dostarczając dokładnych i rzetelnych odpowiedzi na zapytania użytkowników.

## Krok 1: Utwórz nowego asystenta z włączoną funkcją wyszukiwania plików

Utwórz nowego asystenta z włączoną opcją `file_search` w parametrze `tools` asystenta. Po włączeniu narzędzia `file_search` model decyduje, kiedy pobierać zawartość na podstawie wiadomości użytkownika.

```
TsgcAIOpenAIAssistant* Assistant = new TsgcAIOpenAIAssistant(nullptr);
Assistant->OpenAIOptions->ApiKey = "sk-askdjfalskdjfl23kjkjasdefasdjf";
Assistant->AssistantOptions->Name = "sgcWebSockets HelpDesk";
Assistant->AssistantOptions->Instructions->Text =
    "You are a sgcWebSockets HelpDesk Agent. "
    "Answer questions briefly, in a sentence or less. When asked a question, use the manual to answer the question.";
Assistant->AssistantOptions->Model = "gpt-4o-mini";
Assistant->AssistantOptions->Tools->FileSearch->Enabled = true;
Assistant->AssistantOptions->Tools->CodeInterpreter->Enabled = false;
```

## Krok 2: Prześlij pliki i dodaj je do zasobu Vector Store

Aby uzyskać dostęp do plików, narzędzie `file_search` korzysta z obiektu Vector Store. Należy przesłać pliki i utworzyć Vector Store, aby je przechowywać.

```
void UploadFile()
{
```

```

TOpenDialog* oDialog = new TOpenDialog(nullptr);
try
{
    if (oDialog->Execute())
    {
        Screen->Cursor = crHourGlass; // Change cursor to hourglass
        try
        {
            Assistant->UploadVectorStoreFile("sgcVectorStore", oDialog->FileName.c_str());
        }
        __finally
        {
            Screen->Cursor = crDefault; // Reset cursor to default
        }
    }
}
__finally
{
    delete oDialog; // Clean up dialog
}
}

```

## Krok 3: Utwórz uruchomienie i sprawdź wynik

Teraz utwórz Run i zaobserwuj, że model korzysta z narzędzia File Search, aby udzielić odpowiedzi na pytanie użytkownika.

```

void SendMessage()
{
    int i;
    TsgcOpenAIClass_Message* oMessage = nullptr;
    TsgcOpenAIClass_Response_List_Messages* oMessages = nullptr;
    TsgcOpenAIClass_Run* oRun = nullptr;
    DoLog("[user]: " + memoMessage->Lines->Text);
    Screen->Cursor = crHourGlass; // Change cursor to hourglass
    try
    {
        oMessage = Assistant->CreateMessageText("thread_id", "Create a WebSocket Client that connects to eSeGeCe WebSocket");
        if (oMessage != nullptr)
        {
            oRun = Assistant->CreateRunAndWait("thread_id");
            if (oRun != nullptr)
            {
                oMessages = Assistant->GetMessages("thread_id", oRun->Id);
                if (oMessages != nullptr && oMessages->Messages.Length > 0)
                {
                    memoMessage->Lines->Clear();
                    for (i = 0; i < oMessages->Messages.Length; i++)
                    {
                        DoLog("[assistant]: " + DoFormatResponse(oMessages->Messages[i].ContentText + "\r\n"));
                    }
                }
            }
        }
    }
}
__finally
{

```

```
        Screen->Cursor = crDefault; // Reset cursor to default
    }
}
```

# TsgcAIOpenAIAssistant | Strumieniowanie

Zamiast czekać na pełną odpowiedź asystenta, można strumieniować odpowiedź przy użyciu Server-Sent Events. Wystarczy przekazać parametr **Stream = True** podczas korzystania z funkcji **CreateRun**, a odpowiedź będzie używać strumieniowania.

Do obsługi odpowiedzi strumieniowych używane są następujące zdarzenia:

- **OnStreamRun:** zdarzenie jest wywoływane, gdy nastąpi aktualizacja obiektu run.
- **OnStreamMessage:** zdarzenie wywoływane przy aktualizacji obiektu wiadomości: created, in-progress, completed...
- **OnStreamMessageDelta:** Wywoływane, gdy fragmenty wiadomości są strumieniowane.
- **OnStreamDone:** Wywoływane po zakończeniu strumienia.
- **OnStreamError:** Wywoływane, gdy wystąpi błąd. Może to nastąpić z powodu wewnętrznego błędu serwera lub przekroczenia limitu czasu.

## Krok 1: Utwórz asystenta

Asystent reprezentuje podmiot, który można skonfigurować do odpowiadania na wiadomości użytkownika przy użyciu kilku parametrów, takich jak model, instrukcje i narzędzia.

```
// Create a new assistant
TsgcAIOpenAIAssistant *oAssistant = new TsgcAIOpenAIAssistant(NULL);
// Set your API key
oAssistant->OpenAIOptions->ApiKey = txtAPIKey->Text;
// Assistant options
oAssistant->AssistantOptions->Name = "Math Tutor";
oAssistant->AssistantOptions->Instructions->Text = "You are a personal math tutor. Write and run code to answer math questions";
oAssistant->AssistantOptions->Model = "gpt-4o";
// Create the assistant
oAssistant->CreateAssistant();
```

## Krok 2: Utwórz wątek

Wątek (Thread) reprezentuje rozmowę między użytkownikiem a jednym lub wieloma asystentami. Wątek można utworzyć w momencie, gdy użytkownik (lub aplikacja AI) rozpoczyna rozmowę z asystentem.

```
oThread = oAssistant->CreateThread();
```

### Krok 3: Dodaj wiadomość do wątku i uruchom z użyciem strumieniowania

Treść wiadomości tworzonych przez użytkowników lub aplikacje jest dodawana jako obiekty Message do wątku. Wiadomości mogą zawierać zarówno tekst, jak i pliki. Nie ma ograniczeń co do liczby wiadomości, które można dodać do wątków — kontekst, który nie mieści się w oknie kontekstowym modelu, jest inteligentnie przycinany.

Po dodaniu wszystkich wiadomości użytkownika do wątku można uruchomić wątek z dowolnym asystentem. Tworzenie uruchomienia powoduje użycie modelu i narzędzi powiązanych z asystentem do wygenerowania odpowiedzi. Odpowiedzi są wysyłane strumieniowo za pośrednictwem Server-Sent Events.

```
void SendMessage(TsgcAIOpenAIAssistant *oAssistant, TsgcAIClass_Thread *oThread, const UnicodeString aMessage)
{
    int i;
    TsgcOpenAIClass_Message *oMessage;
    TsgcOpenAIClass_Response_List_Messages *oMessages;
    TsgcOpenAIClass_Run *oRun;
    DoLog(L"[user]: " + aMessage);
    oMessage = oAssistant->CreateMessageText(oThread->Id, aMessage);
    if (oMessage != NULL)
    {
        oRun = oAssistant->CreateRun(oThread->Id, true);
    }
}
```

### Krok 4: Obsługa odpowiedzi

Użyj zdarzenia OnStreamMessageDelta, aby odczytać strumieniową wiadomość wysyłaną przez serwer.

```
void __fastcall OnStreamMessageDelta(TObject *Sender, const TsgcOpenAIClass_MessageDelta &aMessageDelta, const String &aRaw)
{
    for (int i = aMessageDelta.Content.Low(); i <= aMessageDelta.Content.High(); i++)
    {
        String vType = aMessageDelta.Content[i]->_Type;
        if (vType == "text")
        {
            String vResponse = static_cast<TsgcOpenAIClass_MessageDeltaContent_Text*>(aMessageDelta.Content[i])->Value;
        }
    }
}
```

# TsgcAIOpenAIAssistant Function Calling

Podobnie jak Chat Completions API, Assistants API obsługuje wywoływanie funkcji. Wywoływanie funkcji umożliwia opisanie funkcji w Assistants API, a następnie inteligentne zwracanie funkcji, które należy wywołać, wraz z ich argumentami.

W tym przykładzie zostanie stworzony asystent pogody i zdefiniowane dwie funkcje, **get\_current\_temperature** i **get\_rain\_probability**, jako narzędzia, które Asystent może wywoływać. W przykładzie z równoległym wywoływaniem funkcji Asystent zostanie zapytany o aktualną pogodę w San Francisco i szansę na deszcz. Pokazano również, jak wyświetlić odpowiedź Asystenta ze strumieniowaniem.

## Krok 1: Definiowanie funkcji

Podczas tworzenia asystenta należy najpierw zdefiniować funkcje w parametrze tools asystenta.

```
TsgcAIOpenAIAssistant *Assistant = new TsgcAIOpenAIAssistant(nullptr);
Assistant->OpenAIOptions->ApiKey = "sk-askdjfalskdjfl23kjkjasdefasdfj";
Assistant->AssistantOptions->Name = "Delphi Weather Bot";
Assistant->AssistantOptions->Instructions->Text = "You are a weather bot. Use the provided functions to answer questions.";
Assistant->AssistantOptions->Model = "gpt-4o";
Assistant->AssistantOptions->Tools->Functions->Enabled = false;
Assistant->AssistantOptions->Tools->Functions->Functions->Text =
    "[{\"type\":\"function\",\"function\":{\"name\":\"get_current_temperature\",\"description\":\"Get the current temperature in San Francisco\"}}]";
Assistant->AssistantOptions->Tools->FileSearch->Enabled = false;
Assistant->AssistantOptions->Tools->CodeInterpreter->Enabled = false;
```

## Krok 2: Utwórz wątek i dodaj wiadomości

Należy utworzyć wątek, gdy użytkownik rozpoczyna rozmowę, i dodawać do niego wiadomości w miarę zadawania pytań przez użytkownika.

```
void SendMessage()
{
    int i;
    TsgcOpenAIClass_Message* oMessage = nullptr;
    TsgcOpenAIClass_Response_List_Messages* oMessages = nullptr;
    TsgcOpenAIClass_Run* oRun = nullptr;
    DoLog("[user]: " + memoMessage->Lines->Text);
    Screen->Cursor = crHourGlass; // Change cursor to hourglass
    try
    {
        oMessage = Assistant->CreateMessageText("thread_id", "What is the weather in San Francisco today and the likelihood of rain?");
        if (oMessage != nullptr)
        {
            oRun = Assistant->CreateRunAndWait("thread_id");
            if (oRun != nullptr)
            {
                // ...
            }
        }
    }
    catch (...)
    {
        // ...
    }
}
```



```

    {
        oMessages = Assistant->GetMessages("thread_id", oRun->Id);
        if (oMessages != nullptr && oMessages->Messages.Length > 0)
        {
            memoMessage->Lines->Clear();
            for (i = 0; i < oMessages->Messages.Length; i++)
            {
                DoLog("[assistant]: " + DoFormatResponse(oMessages->Messages[i].ContentText + "\r\n"));
            }
        }
    }
}
__finally
{
    Screen->Cursor = crDefault; // Reset cursor to default
}
}

```

### Krok 3: obsługa zdarzenia OnFunctionCall

Gdy komponent wykryje, że wymagana jest wartość parametru funkcji, wywoływane jest zdarzenie **OnFunctionCall**. Należy użyć parametru **Request.\_Function**, aby poznać szczegóły żądania, a **Response.Output** do wysłania odpowiedzi.

```

void __fastcall TFRMOpenAIAssistant::AssistantFunctionCall(TObject *Sender,
    const TsgcOpenAIClass_ToolCall &aRequest,
    const TsgcHTTPOpenAI_ToolCall_Response &aResponse)
{
    if (aRequest.Function.Name == "get_current_temperature")
        aResponse.Output = 30;
    else if (aRequest.Function.Name == "get_rain_probability")
        aResponse.Output = 10;
}

```

# OpenAI Audio

Aby korzystać z API OpenAI z komendami głosowymi, wymagane są następujące kroki:

1. Dźwięk z mikrofonu musi zostać przechwycony, dlatego potrzebny jest system zamiany mowy na tekst w celu uzyskania tekstu, który zostanie wysłany do OpenAI.
  1. Przechwytywanie dźwięku z mikrofonu odbywa się przy użyciu komponentu [TsgcAudioRecorderMCI](#).
  2. Po przechwyceniu audio jest ono wysyłane do Whisper API OpenAI w celu konwersji pliku audio na tekst.
2. Po uzyskaniu wyniku rozpoznawania mowy, tekst jest wysyłany do OpenAI za pomocą API Chat-Completion.
3. Odpowiedź z OpenAI musi następnie zostać przekształcona w mowę za pomocą jednego z następujących komponentów:
  1. [TsgcTextToSpeechSystem](#): (obecnie tylko dla Windows) używa systemowej funkcji zamiany tekstu na mowę z systemu operacyjnego.
  2. [TsgcTextToSpeechGoogle](#): wysyła odpowiedź z OpenAI do serwerów Google Cloud i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).
  3. [TsgcTextToSpeechAmazon](#): wysyła odpowiedź z OpenAI do serwerów Amazon AWS i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).

## Komponenty

Poniższe komponenty służą do przechwytywania dźwięku z mikrofonu, odtwarzania plików audio oraz konwersji tekstu na mowę.

- [TsgcAudioRecorderMCI](#): (tylko dla systemu Windows) komponent umożliwiający dostęp do mikrofonu i konwersję mowy do pliku wave.
- [TsgcAudioPlayerMCI](#): (tylko dla Windows) komponent umożliwiający odtwarzanie pliku mp3.
- [TsgcTextToSpeechSystem](#): (wyłącznie dla systemu Windows) konwertuje tekst na mowę bez potrzeby używania zewnętrznego pliku mp3.
- [TsgcTextToSpeechGoogle](#): konwertuje tekst na mowę przy użyciu dostępnych głosów Google Cloud.
- [TsgcTextToSpeechAmazon](#): konwertuje tekst na mowę przy użyciu dowolnego z dostępnych głosów Amazon AWS.

# TsgcAudioRecorderMCI

Ten składnik służy do przechwytywania dźwięku z mikrofonu i zapisywania go w pliku wave.

## Wprowadzenie

Ten komponent służy do przechwytywania dźwięku z mikrofonu i zapisywania go w pliku wave. Obecnie obsługiwany jest tylko system Windows.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 0 zdarzeń do obsłużenia.

# TsgcAudioPlayerMCI

Ten komponent służy do odtwarzania plików mp3 odbieranych od dostawców zamiany tekstu na mowę.

## Wprowadzenie

Ten komponent służy do odtwarzania plików mp3 otrzymanych od dostawców zamiany tekstu na mowę. Obecnie obsługiwany jest wyłącznie system Windows.

## Dokumentacja

- Właściwości — 1 właściwość dostępna na składniku.
- Metody — 1 metoda publiczna.
- Zdarzenia — 0 zdarzeń do obsłużenia.

# TsgcTextToSpeechSystem

Jest to domyślna funkcja zamiany tekstu na mowę dostarczona przez system operacyjny; obecnie obsługiwany jest wyłącznie system Windows.

## Wprowadzenie

Jest to domyślna funkcja zamiany tekstu na mowę dostarczona przez system operacyjny; obecnie obsługiwany jest wyłącznie system Windows.

## Dokumentacja

- Właściwości — 1 właściwość dostępna w komponencie.
- Metody — 1 metoda publiczna.
- Zdarzenia — 0 zdarzeń do obsługi.

# TsgcTextToSpeechGoogle

Text-To-Speech to API dostarczane przez Google Cloud umożliwiające konwersję tekstu na pliki mp3.

## Wprowadzenie

Text-To-Speech to API udostępniane przez Google Cloud, które umożliwia konwersję tekstu na pliki mp3. Wymaga konta Google Cloud oraz skonfigurowania usługi Text-To-Speech.

Po skonfigurowaniu konta Text-To-Speech należy pobrać plik ustawień JSON i przypisać go do właściwości `GoogleOptions.Settings`.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 1 metoda publiczna.
- Zdarzenia — 0 zdarzeń do obsługi.

# TsgcTextToSpeechAmazon

Text-To-Speech to API udostępniane przez Amazon AWS, które umożliwia konwersję tekstu na pliki mp3.

## Wprowadzenie

Text-To-Speech to API dostarczane przez Amazon AWS, które umożliwia konwersję tekstu na pliki mp3. Wymaga konta Amazon AWS oraz konfiguracji API Polly.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 1 metoda publiczna.
- Zdarzenia — 0 zdarzeń do obsługi.

# TsgcAIChat

TsgcAIChat to pojedynczy komponent współpracujący z dowolnym dostawcą AI.

## Wprowadzenie

TsgcAIChat jest pojedynczym komponentem współpracującym z dowolnym dostawcą AI. Zmiana dostawcy następuje przez zmianę jednej właściwości. Komponent zawiera wbudowane zarządzanie historią konwersacji, co ułatwia tworzenie aplikacji czatu bez ręcznego śledzenia wiadomości.

## Obsługiwani dostawcy

- OpenAI
- Anthropic
- Gemini
- DeepSeek
- Ollama
- Grok
- Mistral

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 4 metody publiczne.
- Zdarzenia — 3 zdarzenia do obsłużenia.



# TsgcAIOpenAIChatBot

Należy przechwycić dźwięk z mikrofonu, dlatego wymagany jest system rozpoznawania mowy (speech-to-text), który umożliwi uzyskanie tekstu wysyłanego do OpenAI.

## Wprowadzenie

Aby zbudować ChatBot z poleceniami głosowymi, wymagane są następujące kroki:

1. Dźwięk z mikrofonu musi zostać przechwycony, dlatego potrzebny jest system rozpoznawania mowy, aby uzyskać tekst do wysłania do OpenAI.
  1. Przechwytywanie dźwięku z mikrofonu odbywa się przy użyciu komponentu [TsgcAudioRecorderMCI](#).
  2. Po przechwyceniu dźwięku jest on wysyłany do OpenAI Whisper API w celu konwersji pliku audio na tekst.
2. Po uzyskaniu wyniku rozpoznawania mowy, tekst jest wysyłany do OpenAI za pomocą API Chat-Completion.
3. Odpowiedź z OpenAI musi następnie zostać przekonwertowana na mowę przy użyciu jednego z następujących komponentów:
  1. [TsgcTextToSpeechSystem](#): (obecnie tylko dla systemu Windows) używa systemowej funkcji zamiany tekstu na mowę systemu operacyjnego.
  2. [TsgcTextToSpeechGoogle](#): wysyła odpowiedź z OpenAI do serwerów Google Cloud i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).
  3. [TsgcTextToSpeechAmazon](#): wysyła odpowiedź z OpenAI do serwerów Amazon AWS i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).

## Dokumentacja

- Właściwości — 6 właściwości dostępnych w komponencie.
- Metody — 4 publiczne metody.
- Zdarzenia — 4 zdarzenia do obsługi.

# TsgcAIOpenAITranslator

Należy przechwycić dźwięk z mikrofonu, dlatego wymagany jest system rozpoznawania mowy (speech-to-text), który umożliwi uzyskanie tekstu wysyłanego do OpenAI.

## Wprowadzenie

Aby zbudować translator z obsługą poleceń głosowych, wymagane są następujące kroki:

1. Dźwięk z mikrofonu musi zostać przechwycony, dlatego potrzebny jest system rozpoznawania mowy, aby uzyskać tekst do wysłania do OpenAI.
  1. Przechwytywanie dźwięku z mikrofonu odbywa się przy użyciu komponentu [TsgcAudioRecorderMCI](#).
  2. Po przechwyceniu dźwięku jest on wysyłany do OpenAI Whisper API w celu konwersji pliku audio na tekst.
2. Po uzyskaniu wyniku rozpoznawania mowy, tekst jest wysyłany do OpenAI za pomocą API Chat-Completion.
3. Odpowiedź z OpenAI musi następnie zostać przekonwertowana na mowę przy użyciu jednego z następujących komponentów:
  1. [TsgcTextToSpeechSystem](#): (obecnie tylko dla systemu Windows) używa systemowej funkcji zamiany tekstu na mowę systemu operacyjnego.
  2. [TsgcTextToSpeechGoogle](#): wysyła odpowiedź z OpenAI do serwerów Google Cloud i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).
  3. [TsgcTextToSpeechAmazon](#): wysyła odpowiedź z OpenAI do serwerów Amazon AWS i zwraca plik mp3, który jest odtwarzany przez [TsgcAudioPlayerMCI](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 3 zdarzenia do obsłużenia.

# TsgcAIOpenAIEmbeddings

Osadzenia (embeddings) to sposób reprezentowania słów, fraz lub innych typów danych, takich jak obrazy czy dźwięk, w postaci numerycznej.

## Wprowadzenie

Osadzenia to sposób reprezentowania słów, fraz lub innych typów danych, takich jak obrazy lub dźwięk, w postaci numerycznej. Jest to jak zamiana słów lub danych w liczby, dzięki czemu komputery mogą je lepiej rozumieć i przetwarzać.

Wyobraź sobie zbiór słów, takich jak „pies”, „kot” i „ptak”. Słowa te mają znaczenie, prawda? Otóż osadzenia przypisują każdemu słowu unikalny zestaw liczb (wektorów), który oddaje ich znaczenie i relacje względem innych słów.

Na przykład słowo „dog” może być reprezentowane jako [0,5, 0,2, -0,7], „cat” jako [0,8, -0,3, 0,1], a „bird” jako [0,3, 0,9, 0,4]. Liczby w wektorach niosą informacje o cechach każdego słowa, takich jak przynależność do kategorii zwierząt lub stopień podobieństwa między nimi.

Niesamowitą właściwością osadzeń jest to, że można je wyuczyć na dużych zbiorach danych, dzięki czemu automatycznie rozpoznają podobieństwa i różnice między słowami. Te reprezentacje numeryczne pomagają algorytmom AI rozumieć język i identyfikować złożone wzorce, co jest kluczowe w takich zastosowaniach jak tłumaczenie języka, analiza sentymentu i systemy rekomendacji. Ułatwiają one również modelom AI szybsze przetwarzanie informacji i dostarczanie dokładniejszych wyników.

## Dokumentacja

- Właściwości — 4 właściwości dostępne w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 5 zdarzeń do obsługi.

# TsgcAIDatabaseVectorFile

Komponent przechowuje wektory bazy danych i odpowiedzi w dwóch plikach tekstowych.

## Wprowadzenie

Komponent przechowuje wektory bazy danych i monity w dwóch plikach tekstowych. Komponent ten powinien być używany wyłącznie do celów testowych, a nie produkcyjnych, ponieważ nie jest zoptymalizowany przy dużej liczbie wektorów.

## Konfiguracja

- **VectorFileOptions:**
  - **InputFilename:** nazwa pliku używanego do przechowywania danych wejściowych.
  - **VectorFilename:** nazwa pliku używanego do przechowywania wektorów.

## Dokumentacja

- Właściwości — 2 właściwości dostępne w komponencie.
- Metody — 4 metody publiczne.
- Zdarzenia — 0 zdarzeń do obsługi.

# TsgcAIDatabaseVectorPinecone

Komponent TsgcAIDatabaseVectorPinecone.

## Wprowadzenie

Komponent oparty jest na kliencie [Pinecone API REST](#), który umożliwia tworzenie, aktualizowanie i usuwanie indeksów oraz wektorów.

## Konfiguracja

- **PineconeOptions:**
  - **ApiKey:** należy tu skonfigurować klucz API dostarczony przez Pinecone, który można uzyskać z konta Pinecone.
  - **Environment:** domyślnie jest to środowisko bezpłatnego konta "us-west4-gcp-free".
  - **LogOptions:** należy tu skonfigurować, czy żądania HTTP mają być zapisywane w pliku tekstowym.
- **PineconeIndexOptions:**
  - **IndexName:** nazwa indeksu używanego do przechowywania lub odpytywania danych.
  - **ProjectId:** identyfikator projektu.

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 4 metody publiczne.
- Zdarzenia — 0 zdarzeń do obsługi.

# Embeddings | Tworzenie wektorów

Aby korzystać z osadzeń, najpierw należy przekonwertować dane do postaci wektorów.

## Przykład

Jeśli dostępny jest plik PDF, należy najpierw przekonwertować go na tekst, a następnie użyć metody **CreateEmbeddingsFromFile** w celu uzyskania danych wektorowych. Ze względu na ograniczenie rozmiaru osadzeń OpenAI, jeśli plik jest zbyt duży, dane zostaną automatycznie podzielone na fragmenty, dzięki czemu z jednego pliku można uzyskać jeden lub więcej wektorów.

Poniżej znajduje się przykładowy kod.

```
void ConvertFileToVector()
{
    TOpenDialog* oDialog = new TOpenDialog(NULL);
    try
    {
        oDialog->Filter = "TXT Files|*.txt";
        if (oDialog->Execute())
        {
            TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
            try
            {
                TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
                try
                {
                    {
                        oEmbeddings->Database = oFile;
                        oEmbeddings->OpenAIOptions->ApiKey = "<your api key>";
                        oEmbeddings->CreateEmbeddingsFromFile(oDialog->FileName);
                    }
                    __finally
                    {
                        delete oFile;
                    }
                }
                __finally
                {
                    delete oEmbeddings;
                }
            }
        }
    }
    __finally
    {
        delete oDialog;
    }
}
```

# Embeddings | ChatBot

Po przekształceniu wszystkich danych w wektory można przystąpić do budowy własnego modelu. Idea jest bardzo prosta: za każdym razem, gdy bot otrzymuje pytanie, najpierw zamienia je na wektor, następnie wyszukuje w bazie danych wektor najbardziej podobny do pytania, a na końcu używa najbardziej podobnych danych jako kontekstu.

```
void AskToChatGPT(const std::string& aQuestion)
{
    TsgcAIOpenAIChatBot* oChatBot = new TsgcAIOpenAIChatBot(NULL);
    try
    {
        oChatBot->OpenAIOptions->ApiKey = "<your api key>";
        TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
        try
        {
            oChatBot->Embeddings = oEmbeddings;
            TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
            try
            {
                oEmbeddings->Database = oFile;
                std::string vContext = oChatBot->GetEmbedding(aQuestion);
                std::string message = "Answer the question based on the context below.\n\nContext:\n" +
                    vContext + "\nQuestion:" + aQuestion + "\nAnswer:";
                oChatBot->ChatAsUser(message.c_str());
            }
            __finally
            {
                delete oFile;
            }
        }
        __finally
        {
            delete oEmbeddings;
        }
    }
    __finally
    {
        delete oChatBot;
    }
}
```

# Pinecone

[Pinecone.io](https://pinecone.io)

Pinecone to wektorowa baza danych umożliwiająca łatwe i wydajne przesyłanie, odpytywanie oraz usuwanie danych wektorowych.

Pinecone posiada publiczny interfejs API umożliwiający stronom trzecim integrację Pinecone z własnymi aplikacjami. Komponent `TsgcHTTP_API_Pinecone` jest opakowaniem nad API Pinecone.

## Konfiguracja

Przed rozpoczęciem pracy należy zarejestrować się w serwisie Pinecone i uzyskać klucz API. Klucz ten służy do wysyłania żądań API i musi być ustawiony we właściwości **`PineconeOptions.ApiKey`** komponentu `TsgcHTTP_API_Pinecone`.

## Operacje na indeksach

Obsługiwane są następujące metody:

|                             |                                          |                                                                                                                                                                        |
|-----------------------------|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>IndexesList</code>    |                                          | Ta operacja zwraca listę indeksów Pinecone.                                                                                                                            |
| <code>IndexCreate</code>    | <code>TsgcHTTTPineconeIndexCreate</code> | Ta operacja tworzy indeks Pinecone. Można jej użyć do określenia miary podobieństwa, wymiaru wektorów przechowywanych w indeksie, liczby replik do użycia i nie tylko. |
| <code>IndexDescribe</code>  | Nazwa indeksu                            | Pobierz opis indeksu.                                                                                                                                                  |
| <code>IndexDelete</code>    | Nazwa indeksu                            | Ta operacja usuwa istniejący indeks.                                                                                                                                   |
| <code>IndexConfigure</code> | Index Name, Replicas, PodType            | Ta operacja określa typ poda i liczbę replik dla indeksu.                                                                                                              |

## Operacje na kolekcji

Obsługiwane są następujące metody:

|                              |  |                                             |
|------------------------------|--|---------------------------------------------|
| <code>CollectionsList</code> |  | Ta operacja zwraca listę kolekcji Pinecone. |
|------------------------------|--|---------------------------------------------|



|                    |                        |                                        |
|--------------------|------------------------|----------------------------------------|
| CollectionCreate   | Nazwa kolekcji, źródło | Ta operacja tworzy kolekcję Pinecone.  |
| CollectionDescribe | Nazwa kolekcji         | Pobiera opis kolekcji.                 |
| CollectionDelete   | Nazwa kolekcji         | Ta operacja usuwa istniejącą kolekcję. |

## Operacje na wektorach

Obsługiwane są następujące metody:

|                           |                                                       |                                                                                                                                                                                                                                             |
|---------------------------|-------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VectorsDescribeIndexStats | Nazwa indeksu, identyfikator projektu, filtr          | Operacja DescribeIndexStats zwraca statystyki dotyczące zawartości indeksu, w tym liczbę wektorów na przestrzeń nazw oraz liczbę wymiarów.                                                                                                  |
| VectorsQuery              | Index Name, Project Id, Params                        | Operacja Query przeszukuje przestrzeń nazw przy użyciu wektora zapytania. Pobiera identyfikatory najbardziej podobnych elementów w przestrzeni nazw wraz z ich wynikami podobieństwa.                                                       |
| VectorsDelete             | Index Name, Project Id, Params                        | Operacja Delete usuwa wektory według identyfikatora z pojedynczej przestrzeni nazw. Elementy można usuwać według ich identyfikatora z pojedynczej przestrzeni nazw.                                                                         |
| VectorsFetch              | Nazwa indeksu, identyfikator projektu, identyfikatory | Operacja Fetch wyszukuje i zwraca wektory według identyfikatora z pojedynczej przestrzeni nazw.<br>Zwrócone wektory zawierają dane wektorowe i/lub metadane.                                                                                |
| VectorsUpdate             | Index Name, Project Id, Params                        | Operacja Update aktualizuje wektor w przestrzeni nazw.<br>Jeśli zostanie podana wartość, nadpisze poprzednią wartość.<br>Jeśli zostanie podane set_metadata, wartości pól w nim określonych zostaną dodane lub nadpiszą poprzednią wartość. |
| VectorsUpsert             | Index Name, Project Id, Params                        | Operacja Upsert zapisuje wektory w przestrzeni nazw.<br>Jeśli dla istniejącego identyfikatora wektora zostanie wstawiona nowa wartość, poprzednia wartość zostanie nadpisana.                                                               |

## Przykład UPSERT

Poniżej znajduje się przykład UPSERT pojedynczego wektora o identyfikatorze Id = "id1".

```
void UpsertPinecone(const String aIndexName, const String aProjectId, const std::vector<double> aVector)
{
    TsgcHTTP_API_Pinecone* oPinecone = new TsgcHTTP_API_Pinecone(NULL);
    try
    {
        oPinecone->PineconeOptions.API = "your-api-key";
        TsgcHTTTPineconeVectorUpserts* oParams = new TsgcHTTTPineconeVectorUpserts();
        try
```

```

{
    TsgcArrayOfVectorUpsert oVectors;
    oVectors.push_back(new TsgcHTTPPineconeVectorUpsert());
    oVectors[0]->Id = "id1";
    oVectors[0]->Values = aVector;
    oParams->Vectors = oVectors;
    oPinecone->VectorsUpsert(aIndexName, aProjectId, oParams);
}
__finally
{
    oParams->Free();
}
}
__finally
{
    oPinecone->Free();
}
}

```

## Przykład QUERY

Poniżej znajduje się przykład zapytania QUERY o pojedynczy wektor.

```

void QueryPinecone(const string aIndexName, const string aProjectId, const std::vector<double>& aVector)
{
    TsgcHTTPPineconeVectorQuery* oParams = new TsgcHTTPPineconeVectorQuery();
    try
    {
        oParams->Vector = aVector;
        Pinecone.VectorsQuery(aIndexName, aProjectId, oParams);
    }
    __finally
    {
        oParams->Free();
    }
}

```

# Anthropic Claude

Anthropic to firma zajmująca się bezpieczeństwem AI, która tworzy niezawodne, interpretowalne i sterowalne systemy AI. Jej flagową rodziną modeli jest Claude, który wyróżnia się w przemyślanym dialogu, tworzeniu treści, złożonym rozumowaniu, programowaniu i nie tylko.

Biblioteka `sgcWebSockets` udostępnia komponent Delphi **TsgcHTTP\_API\_Anthropic** do interakcji z Anthropic Claude API.

## Anthropic API

Anthropic API zapewnia dostęp do modeli Claude do tworzenia aplikacji opartych na AI. API obsługuje generowanie tekstu, wizję (rozumienie obrazów), używanie narzędzi (wywoływanie funkcji), rozszerzone myślenie, przetwarzanie dokumentów/PDF, buforowanie monitów, cytaty, wyszukiwanie w sieci, przesyłanie strumieniowe, liczenie tokenów i partie wiadomości.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Anthropic](#)
- **Vision**
  - [Przykłady Anthropic Vision](#)
- **Użycie narzędzi**
  - [Przykłady użycia narzędzi Anthropic](#)
- **Modele**
  - [Przykłady modeli Anthropic](#)
- **Partie**
  - [Przykłady partii Anthropic](#)
- **Extended Thinking**
  - [Przykłady Anthropic Extended Thinking](#)
- **Dokumenty**
  - [Przykłady dokumentów Anthropic](#)
- **Buforowanie promptów**
  - [Przykłady buforowania promptów Anthropic](#)
- **Cytowania**
  - [Przykłady cytowań Anthropic](#)
- **Wyszukiwanie w sieci**
  - [Przykłady wyszukiwania w sieci Anthropic](#)
- **Ustrukturyzowane dane wyjściowe**
  - [Przykłady ustrukturyzowanych danych wyjściowych Anthropic](#)
- **Pliki**
  - [Przykłady Anthropic Files API](#)
- **MCP Connector**
  - [Przykłady Anthropic MCP Connector](#)

## Konfiguracja

Anthropic API używa kluczy API do uwierzytelniania. Odwiedź stronę [Klucze API](#) w konsoli Anthropic, aby pobrać klucz API używany w żądaniach.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi być skonfigurowany we właściwości **AnthropicOptions.ApiKey** komponentu. Właściwość **AnthropicOptions.AnthropicVersion** określa wersję API (domyślnie: 2023-06-01).

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "YOUR_API_KEY";
```

## Właściwości

### AnthropicOptions

- **ApiKey:** Klucz API do uwierzytelniania z Anthropic API.
- **AnthropicVersion:** Ciąg wersji API (domyślnie: 2023-06-01).
- **BetaHeaders:** Opcjonalne nagłówki funkcji beta umożliwiające korzystanie z przedpremierowych funkcji API (np. files-api-2025-04-14, mcp-client-2025-11-20).
- **HttpOptions:** Ustawienia połączenia HTTP.
  - **ReadTimeout:** Limit czasu w milisekundach na odczyt odpowiedzi HTTP. Wartość domyślna to 0 (brak limitu czasu).
- **LogOptions:** Konfiguracja rejestrowania.
  - **Enabled:** Gdy ma wartość True, żądania i odpowiedzi HTTP są zapisywane do pliku.
  - **FileName:** Ścieżka pliku, do którego zapisywane są dane dziennika.
- **RetryOptions:** Konfiguracja automatycznego ponawiania nieudanych żądań API.
  - **Enabled:** Gdy True, nieudane żądania są automatycznie ponawiane.
  - **Retries:** Maksymalna liczba prób ponowienia.
  - **Wait:** Czas oczekiwania w milisekundach między próbami ponowienia.

## Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z zawartością tekstową i/lub graficzną, a model wygeneruje kolejną wiadomość w konwersacji.

- **CreateMessage:** Tworzy wiadomość z określonym modelem i parametrami.
  - **Model:** Model do użycia (np. claude-sonnet-4-20250514).
  - **Message:** Treść wiadomości użytkownika.
  - **MaxTokens:** Maksymalna liczba tokenów do wygenerowania (wymagane, domyślnie 4096).
- **CreateMessageWithSystem:** Tworzy wiadomość z promptem systemowym.
  - **System:** Prompt systemowy określający zachowanie asystenta.
- **CreateMessageStream:** Tworzy wiadomość z włączonym strumieniowaniem (SSE). Zdarzenia są dostarczane za pośrednictwem procedury obsługi zdarzenia OnHTTPAPISSE.
- **CreateMessageJSON:** Tworzy wiadomość zwracającą ustrukturyzowany JSON zgodny z podanym schematem JSON.
  - **Schema:** Ciąg znaków JSON Schema definiujący format wyjściowy.
- **CreateMessageWithThinking:** Tworzy wiadomość z włączonym rozszerzonym myśleniem.
  - **BudgetTokens:** Maksymalny budżet tokenów na myślenie (minimum 1024).
- **CreateDocumentMessage:** Tworzy wiadomość z dokumentem zakodowanym w base64 (PDF lub tekst).
  - **DocumentBase64:** Dane dokumentu zakodowane w base64.

- **MediaType:** Typ MIME (np. application/pdf).

## Vision

Claude rozumie obrazy przekazywane jako bloki treści zakodowane w formacie base64 wewnątrz wiadomości.

- **CreateVisionMessage:** Wysyła obraz wraz z monitem tekstowym.
  - **ImageBase64:** Dane obrazu zakodowane w base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.

## Użycie narzędzi

Claude może używać narzędzi (function calling) do interakcji z zewnętrznymi systemami. Narzędzia definiuje się za pomocą nazw, opisów i schematów wejściowych, a Claude generuje bloki treści tool\_use, gdy chce wywołać narzędzie.

## Modele

Wyświetla i opisuje dostępne modele Claude.

- **GetModels:** Wyświetla wszystkie dostępne modele.
- **GetModel:** Pobiera informacje o określonym modelu.
  - **ModelId:** Identyfikator modelu do pobrania.

## Extended Thinking

Rozszerzone myślenie umożliwia Claude'owi rozumowanie krok po kroku przed udzieleniem odpowiedzi, co poprawia jakość wyników w przypadku złożonych zadań, takich jak matematyka, programowanie i analiza.

- **ThinkingType:** Ustaw na 'enabled', aby aktywować tryb myślenia, lub na 'disabled', aby go wyłączyć.
- **ThinkingBudgetTokens:** Budżet tokenów dla myślenia (minimum 1024, musi być mniejszy niż MaxTokens).
- **CreateMessageWithAdaptiveThinking:** Tworzy wiadomość z adaptacyjnym myśleniem, pozwalając Claude samodzielnie decydować o głębokości rozumowania.

## Dokumenty

Claude może przetwarzać dokumenty PDF i pliki tekstowe wysyłane jako bloki treści. Obsługuje typy źródłowe base64, text i URL.

- **CreateDocumentMessage:** Wysyła dokument z monitem tekstowym.
  - **DocumentBase64:** Dane dokumentu zakodowane w base64.
  - **MediaType:** Typ MIME (application/pdf, text/plain).

## Buforowanie promptów

Buforuj często używany kontekst (monity systemowe, bloki treści, definicje narzędzi) między wywołaniami API, aby zmniejszyć koszty nawet o 90% przy odczytach z pamięci podręcznej.

- **SystemCacheControl:** ustaw na True w żądaniu, aby buforować prompt systemowy.

- **CacheControl:** Ustaw wartość 'ephemeral' dla bloków treści lub definicji narzędzi.
- **BetaHeaders:** Opcjonalne nagłówki funkcji beta w AnthropicOptions.

## Cytowania

Gdy dokumenty są wysyłane z włączonymi cytatami, Claude dołącza odwołania do źródeł w swojej odpowiedzi. Typy cytatów obejmują: `char_location` (tekst), `page_location` (PDF), `content_block_location` (treść niestandardowa) i `web_search_result_location` (wyniki wyszukiwania internetowego).

## Wyszukiwanie w internecie

Claude może przeszukiwać Internet w celu uzyskania informacji w czasie rzeczywistym za pomocą wbudowanego narzędzia `web_search`. Inne wbudowane narzędzia to `code_execution` i `computer use`.

- **CreateMessageWithWebSearch:** Tworzy wiadomość z włączonym wyszukiwaniem w sieci.
  - **Model:** Model do użycia.
  - **Message:** Zapytanie użytkownika.

## Zliczanie tokenów

Policz liczbę tokenów w wiadomości przed jej wysłaniem.

- **CountTokens:** Zlicza liczbę tokenów wejściowych dla wiadomości.
  - **Model:** Model używany do zliczania tokenów.
  - **Message:** Zawartość wiadomości, dla której zliczane są tokeny.

## Paczki komunikatów

API wsadowego przetwarzania wiadomości umożliwia asynchroniczne przetwarzanie dużych wolumenów wiadomości.

- **ListBatches:** Wyświetla listę wszystkich partii wiadomości.
- **GetBatch:** Pobiera określoną partię według identyfikatora.
- **CancelBatch:** Anuluje wsad, który jest nadal przetwarzany.
- **GetBatchResults:** Pobiera wyniki ukończonej partii.

## Wyjścia strukturyzowane

Wymusza na Claude zwrócenie prawidłowego JSON zgodnego z dostarczonym schematem JSON. Można łączyć z parametrem `Effort`, aby kontrolować jakość danych wyjściowych i koszty.

- **CreateMessageJSON:** tworzy wiadomość z wyjściem w schemacie JSON.
  - **Schema:** ciąg JSON Schema definiujący format wyjściowy.
- **OutputFormatSchema:** (właściwość żądania) schemat JSON dla ustrukturyzowanego wyjścia.
- **Effort:** (właściwość żądania) Steruje stosunkiem jakości do kosztu: 'low', 'medium', 'high', 'max'.
- **Strict:** (właściwość narzędzia) Gdy wartość wynosi True, dane wejściowe narzędzia są gwarantowanie zgodne z `input_schema`.

## Files API

Przesyłanie, wyświetlanie listy, pobieranie, pobieranie i usuwanie plików. Przesłane pliki mogą być przywoływane w wiadomościach za pomocą bloków treści dokumentów ze źródłowym typem pliku. Wymaga nagłówka beta `files-api-2025-04-14`.

- **UploadFile:** Przesyła plik lokalny. Zwraca metadane pliku.
- **ListFiles:** Wyświetla przesłane pliki z podziałem na strony.
- **GetFile:** Pobiera metadane dla określonego pliku.
- **DownloadFile:** Pobiera zawartość pliku.
- **DeleteFile:** Trwale usuwa plik.

## Parametry żądania

Dodatkowe parametry żądania dostępne w Messages API.

- **ServiceTier:** Kontroluje użycie warstwy priorytetu: 'auto' (domyślnie) lub 'standard\_only'.
- **InferenceGeo:** Kontroluje lokalizację geograficzną inferencji: 'global' (domyślnie) lub 'us'.
- **Container:** Identyfikator kontenera do ponownego użycia środowisk wykonywania kodu w kolejnych żądaniach.
- **IsError:** (właściwość bloku treści) Należy ustawić True w blokach tool\_result, aby wskazać błąd wykonania narzędzia.
- **CacheTTL:** Rozszerzony czas życia pamięci podręcznej: '5m' (domyślnie) lub '1h'.
- **ThinkingType:** Ustaw na 'adaptive', aby Claude automatycznie decydował o głębokości myślenia.

## MCP Connector

Połącz Claude z zewnętrznymi serwerami MCP (Model Context Protocol), aby uzyskać dostęp do narzędzi zewnętrznych. Wymaga nagłówka beta `mcp-client-2025-11-20`.

- **CreateMessageWithMCP:** Tworzy wiadomość z połączeniem serwera MCP.
  - **MCPServerUrl:** Adres URL HTTPS serwera MCP.
  - **MCPServerName:** Unikalna nazwa serwera.
- **MCPServers:** (właściwość Request) Tablica definicji serwerów MCP z polami ServerType, Url, Name, AuthorizationToken.
- **MCPServerName:** (właściwość narzędzia) Odwołuje się do serwera MCP, gdy ToolType ma wartość 'mcp\_toolset'.

# TsgcHTTP\_API\_Anthropic

Klient Anthropic REST API dla wiadomości Claude, plików, wsadów i liczenia tokenów.

## Wprowadzenie

Klient Anthropic REST API dla wiadomości Claude, plików, wsadów i liczenia tokenów.

## Wprowadzenie

Klient Anthropic REST API dla wiadomości Claude, plików, wsadów i liczenia tokenów.

Skonfiguruj właściwość **AnthropicOptions** podając klucz API i opcjonalne ustawienia HTTP/retry/log, a następnie wywołuj typowane metody, takie jak `CreateMessage/CreateChatCompletion`, przekazując obiekt żądania i odbierając typowaną klasę odpowiedzi.

Informacje na temat dostawcy zawiera przewodnik [Anthropic](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 2 zdarzenia do obsługi.



# Anthropic | Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

## Prosty przykład

Wyślij wiadomość Hello do Claude.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_CreateMessage("claude-sonnet-4-20250514", "Hello!"));
```

## Przykład monitu systemowego

Wyślij wiadomość z systemowym promptem, aby kontrolować zachowanie Claude.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_CreateMessageWithSystem("claude-sonnet-4-20250514",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Zaawansowany przykład

Użyj typizowanych klas żądania/odpowiedzi, aby mieć pełną kontrolę nad parametrami wiadomości.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 1024;
    oRequest->System = "You are a helpful assistant.";
    oRequest->Temperature = 0.7;

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oMessage->Content = "Hello!";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        if (oResponse->Content.Length > 0)
            ShowMessage(oResponse->Content[0]->Text);
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oMessage;
    delete oRequest;
}
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
Anthropic->OnHTTPAPISSE = OnSSEEvent;
Anthropic->_CreateMessageStream("claude-sonnet-4-20250514", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    // aEvent contains the event type (e.g. content_block_delta)
    // aData contains the JSON data for this event
    Memo1->Lines->Add(aData);
}
```

# Anthropic | Vision

Claude potrafi rozumieć i analizować obrazy. Obrazy można wysyłać jako dane zakodowane w base64 w blokach zawartości.

## Obsługiwane formaty obrazów

- **image/jpeg** - obrazy JPEG
- **image/png** - Obrazy PNG
- **image/gif** — obrazy GIF
- **image/webp** — obrazy WebP

## Prosty przykład

Wyślij obraz z monitem z prośbą do Claude o jego opisanie.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

// Load image and encode to base64
TFileStream *oStream = new TFileStream("photo.png", fmOpenRead);
try {
    TBytesStream *oBytes = new TBytesStream();
    try {
        oBytes->CopyFrom(oStream, 0);
        vBase64 = EncodeBase64(oBytes->Memory, oBytes->Size);
    } __finally {
        oBytes->Free();
    }
} __finally {
    oStream->Free();
}

ShowMessage(Anthropic->_CreateVisionMessage("claude-sonnet-4-20250514",
    "What is in this image?", vBase64, "image/png"));
```

## Zaawansowany przykład

Należy użyć bloków treści, aby uzyskać większą kontrolę nad wiadomością z obrazem.

```
TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages(NULL);
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message(NULL);
    oMessage->Role = "user";

    // Image content block
    TsgcAnthropicClass_Request_Content_Block *oImageBlock = new TsgcAnthropicClass_Request_Content_Block(NULL);
    oImageBlock->ContentType = "image";
    oImageBlock->MediaType = "image/jpeg";
    oImageBlock->Data = vBase64;

    // Text content block
    TsgcAnthropicClass_Request_Content_Block *oTextBlock = new TsgcAnthropicClass_Request_Content_Block(NULL);
    oTextBlock->ContentType = "text";
    oTextBlock->Text = "Describe this image in detail.";
```

```
SetLength(oBlocks, 2);
oBlocks[0] = oImageBlock;
oBlocks[1] = oTextBlock;
oMessage->ContentBlocks = oBlocks;

SetLength(oMessages, 1);
oMessages[0] = oMessage;
oRequest->Messages = oMessages;

TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
try {
    if (Length(oResponse->Content) > 0)
        ShowMessage(oResponse->Content[0]->Text);
} __finally {
    oResponse->Free();
}
} __finally {
    oImageBlock->Free();
    oTextBlock->Free();
    oMessage->Free();
    oRequest->Free();
}
```

# Anthropic | Użycie narzędzi

Claude obsługuje użycie narzędzi (wywoływanie funkcji), umożliwiając definiowanie narzędzi, które Claude może wywoływać podczas rozmowy. Gdy Claude zdecyduje się użyć narzędzia, zwraca blok treści **tool\_use**. Następnie należy wykonać narzędzie i odesłać wynik jako blok treści **tool\_result**.

## Przebieg użycia narzędzia

1. Definiowanie narzędzi z nazwą, opisem i input\_schema (JSON Schema).
2. Wyślij wiadomość z określonymi narzędziami.
3. Claude odpowiada blokiem zawartości **tool\_use** (stop\_reason = 'tool\_use').
4. Wykonanie narzędzia z podanymi danymi wejściowymi.
5. Wyślij nową wiadomość z blokiem treści **tool\_result** zawierającym wynik.
6. Claude odpowiada ostateczną odpowiedzią.

## Przykład

Zdefiniuj narzędzie pogodowe i obsłuż pętlę użycia narzędzia.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

// Step 1: Create request with tools
TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages(NULL);
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    // Define user message
    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message(NULL);
    oMessage->Role = "user";
    oMessage->Content = "What is the weather in San Francisco?";
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    // Define tool
    TsgcAnthropicClass_Request_Tool *oTool = new TsgcAnthropicClass_Request_Tool(NULL);
    oTool->Name = "get_weather";
    oTool->Description = "Get the current weather in a given location";
    oTool->InputSchema =
        "{\"type\":\"object\",\"properties\":{\"location\":{\"type\":\"string\", \"description\":\"The city and state\"}},\"required\":[\"location\"]}";
    SetLength(oTools, 1);
    oTools[0] = oTool;
    oRequest->Tools = oTools;

    // Step 2: Send request
    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        // Step 3: Check if Claude wants to use a tool
        if (oResponse->StopReason == "tool_use") {
            // Find the tool_use content block
            for (int i = 0; i < Length(oResponse->Content); i++) {
                if (oResponse->Content[i]->ContentType == "tool_use") {
                    vToolUseId = oResponse->Content[i]->Id;
                    vToolName = oResponse->Content[i]->Name;
                    vToolInput = oResponse->Content[i]->Input;
                    // Step 4: Execute your tool (get_weather) and get result
                    vToolResult = "72 degrees and sunny";
                }
            }
        }
    }
}
```

```
        break;
    }
}

// Step 5: Send tool result back
// Build new message array with assistant response + tool result
// ... (continue the conversation with tool_result content block)
}
} __finally {
    oResponse->Free();
}
} __finally {
    oMessage->Free();
    oTool->Free();
    oRequest->Free();
}
```

# Anthropic | Modele

Wyświetla dostępne modele Claude i pobiera o nich informacje.

## Lista modeli

Wyświetla listę wszystkich dostępnych modeli Claude.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_GetModels());
```

## Pobierz model

Pobiera informacje o określonym modelu.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_GetModel("claude-sonnet-4-20250514"));
```

## Odpowiedź z typowaniem

Użyj typowanej klasy odpowiedzi w celu strukturyzowanego dostępu do danych modelu.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Response_Models *oModels = Anthropic->GetModels();
try {
    for (int i = 0; i < oModels->Data.Length; i++)
        ShowMessage(oModels->Data[i]->Id + " - " + oModels->Data[i]->DisplayName);
} __finally {
    delete oModels;
}
```

# Anthropic | Partie wiadomości

Message Batches API umożliwia asynchroniczne przetwarzanie dużych ilości wiadomości. Jest to idealne rozwiązanie dla zadań, które nie wymagają natychmiastowych odpowiedzi, takich jak masowe generowanie treści, analiza danych lub potoki przetwarzania wsadowego.

## Wyświetl partii

Wyświetla listę wszystkich wsadów wiadomości w organizacji.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_ListBatches());
```

## Pobierz wsad

Pobiera informacje o określonym wsadzie.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_GetBatch("batch_id_here"));
```

## Anuluj partię

Anuluje wsad, który jest nadal przetwarzany.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_CancelBatch("batch_id_here"));
```

## Pobierz wyniki wsadowe

Pobiera wyniki zakończonej partii.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_GetBatchResults("batch_id_here"));
```

## Odpowiedź z typowaniem

Należy użyć typizowanej klasy odpowiedzi w celu ustrukturyzowanego dostępu do danych wsadowych.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Response_Batches *oBatches = Anthropic->ListBatches();
try {
    for (int i = 0; i < Length(oBatches->Batches); i++)
        ShowMessage(oBatches->Batches[i]->Id + " - " + oBatches->Batches[i]->ProcessingStatus);
} __finally {
    oBatches->Free();
}
```



# Anthropic | Rozszerzone myślenie

Rozszerzone myślenie umożliwia modelowi Claude analizowanie złożonych problemów krok po kroku przed udzieleniem odpowiedzi. Po włączeniu Claude tworzy wewnętrzne rozumowanie (bloki myślenia), które poprawia jakość odpowiedzi dotyczących matematyki, programowania, analizy i innych złożonych zadań.

## Prosty przykład

Wyślij wiadomość z włączonym rozszerzonym myśleniem przy użyciu metody wygodnej. Temperatura jest automatycznie ustawiana na 1,0 (wymagane przez API, gdy włączone jest myślenie).

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_CreateMessageWithThinking("claude-sonnet-4-20250514",
    "How many r's are in the word strawberry?", 10000));
```

## Zaawansowany przykład

Użyj typowanych klas żądań i odpowiedzi, aby uzyskać pełną kontrolę. Ustaw ThinkingType na „enabled”, a ThinkingBudgetTokens na żądany budżet tokenów (minimum 1024). Odpowiedź będzie zawierać bloki treści myślenia i tekstu.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 16384;
    oRequest->ThinkingType = "enabled";
    oRequest->ThinkingBudgetTokens = 10000;

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oMessage->Content = "Explain the proof that there are infinitely many primes.";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        for (int i = 0; i < oResponse->Content.Length; i++) {
            if (oResponse->Content[i]->ContentType == "thinking")
                ShowMessage("Thinking: " + oResponse->Content[i]->Thinking);
            else if (oResponse->Content[i]->ContentType == "text")
                ShowMessage("Response: " + oResponse->Content[i]->Text);
        }
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oMessage;
    delete oRequest;
}
```

## Wieloetapowy dialog z myśleniem

Podczas używania rozszerzonego myślenia w konwersacjach wieloturuowych bloki thinking i redacted\_thinking z poprzednich odpowiedzi muszą być przekazywane z powrotem w konwersacji. Należy użyć tablicy ContentBlocks, aby uwzględnić te bloki.

```
// Pass back thinking blocks from a previous response
TsgcAnthropicClass_Request_Content_Block *oBlock = new TsgcAnthropicClass_Request_Content_Block();
oBlock->ContentType = "thinking";
oBlock->Text = oPrevThinkingBlock->Thinking; // thinking text
oBlock->Signature = oPrevThinkingBlock->Signature; // signature string

// Pass back redacted thinking blocks
TsgcAnthropicClass_Request_Content_Block *oBlock2 = new TsgcAnthropicClass_Request_Content_Block();
oBlock2->ContentType = "redacted_thinking";
oBlock2->Data = oPrevRedactedBlock->Data;
```

## Właściwości

- **ThinkingType:** Ustaw na 'enabled', aby aktywować rozszerzone myślenie, lub na 'disabled', aby je wyłączyć.
- **ThinkingBudgetTokens:** Maksymalna liczba tokenów przeznaczonych na myślenie (minimum 1024). Musi być mniejsza niż MaxTokens.

## Uwagi

- Temperatura musi wynosić 1,0, gdy tryb myślenia jest włączony (metoda pomocnicza ustawia to automatycznie).
- Parametru TopK nie można używać, gdy włączone jest myślenie (thinking).
- BudgetTokens musi wynosić co najmniej 1024 i być mniejsze niż MaxTokens.
- Bloki treści odpowiedzi mogą zawierać następujące typy: thinking, redacted\_thinking i text.

# Anthropic | Dokumenty

Claude może przetwarzać dokumenty PDF i pliki tekstowe wysyłane jako bloki treści w wiadomościach. Dokumenty są przesyłane jako dane zakodowane w base64 lub przez URL, a Claude może analizować, streszczać i odpowiadać na pytania dotyczące ich zawartości.

## Prosty przykład

Wyślij dokument PDF z pytaniem, korzystając z metody pomocniczej.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

// Load PDF file and encode to base64
AnsiString vBase64 = sgcBase64Encode(LoadFileToBytes("document.pdf"));
ShowMessage(Anthropic->_CreateDocumentMessage("claude-sonnet-4-20250514",
    "Summarize this document.", vBase64, "application/pdf"));
```

## Zaawansowany przykład z cytatami

Używaj typowanych klas do wysyłania dokumentu z włączonymi cytatami. Gdy cytaty są włączone, odpowiedź Claude będzie zawierać odwołania do określonych fragmentów dokumentu źródłowego.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    // Create document content block
    TsgcAnthropicClass_Request_Content_Block *oDocBlock = new TsgcAnthropicClass_Request_Content_Block();
    oDocBlock->ContentType = "document";
    oDocBlock->SourceType = "base64";
    oDocBlock->MediaType = "application/pdf";
    oDocBlock->Data = sgcBase64Encode(LoadFileToBytes("report.pdf"));
    oDocBlock->Title = "Annual Report";
    oDocBlock->CitationsEnabled = true;

    // Create text prompt block
    TsgcAnthropicClass_Request_Content_Block *oTextBlock = new TsgcAnthropicClass_Request_Content_Block();
    oTextBlock->ContentType = "text";
    oTextBlock->Text = "What are the key findings?";

    // Build message with content blocks
    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oBlocks = oMessage->ContentBlocks;
    SetLength(oBlocks, 2);
    oBlocks[0] = oDocBlock;
    oBlocks[1] = oTextBlock;
    oMessage->ContentBlocks = oBlocks;

    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
```

```
try {  
    if (oResponse->Content.Length > 0)  
        ShowMessage(oResponse->Content[0]->Text);  
} __finally {  
    delete oResponse;  
}  
} __finally {  
    delete oRequest;  
}
```

## Właściwości

- **ContentType:** Ustawić na 'document' dla bloków zawartości dokumentu.
- **SourceType:** Typ źródła: 'base64' (domyślnie), 'text' lub 'url'.
- **MediaType:** Typ MIME (np. application/pdf, text/plain).
- **Data:** Dane dokumentu zakodowane w base64 lub URL, gdy SourceType ma wartość 'url'.
- **Title:** Opcjonalny tytuł dokumentu służący jako odniesienie.
- **Kontekst:** Opcjonalny kontekst lub metadane dotyczące dokumentu.
- **CitationsEnabled:** Ustaw na True, aby włączyć cytowania źródeł w odpowiedzi.

## Obsługiwane formaty

- **PDF:** application/pdf (maks. 32 MB, maks. 100 stron na żądanie)
- **Plain Text:** text/plain

# Anthropic | Buforowanie wypowiedzi

Buforowanie promptów umożliwia zapisywanie często używanego kontekstu między wywołaniami API, redukując koszty nawet o 90% przy odczycie z pamięci podręcznej i poprawiając opóźnienia. Buforowana zawartość jest oznaczana parametrem `cache_control` i jest ponownie wykorzystywana w kolejnych żądaniach w oknie TTL pamięci podręcznej.

## Buforowanie systemowego monitu

Buforuj długi systemowy monit, aby uniknąć jego ponownego przetwarzania przy każdym żądaniu. Ustawić `SystemCacheControl` na `True`, aby automatycznie owijać systemowy monit kontrolą pamięci podręcznej.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages(NULL);
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;
    oRequest->System = "You are an expert legal assistant... (long system prompt)";
    oRequest->SystemCacheControl = true; // Enable caching for system prompt

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message(NULL);
    oMessage->Role = "user";
    oMessage->Content = "Analyze this contract clause.";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        // Check cache usage in response
        ShowMessage("Cache created: " +
            IntToStr(oResponse->Usage->CacheCreationInputTokens));
        ShowMessage("Cache read: " +
            IntToStr(oResponse->Usage->CacheReadInputTokens));
        ShowMessage(oResponse->Content[0]->Text);
    } __finally {
        oResponse->Free();
    }
} __finally {
    oMessage->Free();
    oRequest->Free();
}
```

## Buforowanie bloków zawartości

Buforuje określone bloki treści (np. duże dokumenty lub kontekst) przez ustawienie `CacheControl` na `'ephemeral'` dla poszczególnych bloków treści.

```
TsgcAnthropicClass_Request_Content_Block *oBlock = new TsgcAnthropicClass_Request_Content_Block(NULL);
oBlock->ContentType = "text";
oBlock->Text = "(large reference text to cache)";
oBlock->CacheControl = "ephemeral"; // Mark for caching
```

## Buforowanie definicji narzędzi

Definicje narzędzi należy buforować, aby uniknąć ich ponownego przetwarzania przy każdym żądaniu. Jest to przydatne, gdy istnieje wiele definicji narzędzi, które pozostają stałe w kolejnych żądaniach.

```
TsgcAnthropicClass_Request_Tool *oTool = new TsgcAnthropicClass_Request_Tool(NULL);
oTool->Name = "search_database";
oTool->Description = "Search the database for records.";
oTool->InputSchema = "{\"type\":\"object\", \"properties\": {\"query\": {\"type\":\"string\"}}}";
oTool->CacheControl = "ephemeral"; // Cache this tool definition
```

## Właściwości

- **SystemCacheControl:** Ustaw wartość True w żądaniu, aby buforować monit systemowy z efemeryczną kontrolą pamięci podręcznej (cache\_control).
- **CacheControl:** Ustaw na 'ephemeral' w blokach zawartości lub definicjach narzędzi, aby oznaczyć je do buforowania.
- **CacheCreationInputTokens:** (Odpowiedź) Liczba tokenów użytych do utworzenia wpisów pamięci podręcznej.
- **CacheReadInputTokens:** (Odpowiedź) Liczba tokenów odczytanych z pamięci podręcznej (oszczędności kosztów).

## Cennik

- **Zapisy w pamięci podręcznej:** 1,25-krotność ceny bazowej tokenu wejściowego (TTL 5 minut).
- **Odczyty z pamięci podręcznej:** 0,1x ceny bazowego tokenu wejściowego (oszczędność 90%).
- **Czas życia pamięci podręcznej:** domyślnie 5 minut. Kolejne żądania odświeżają czas życia.

# Anthropic | Cytaty

Gdy dokumenty są wysyłane z włączonymi cytowaniami, odpowiedź Claude zawiera odwołania do konkretnych części dokumentów źródłowych. Umożliwia to weryfikację twierdzeń i śledzenie informacji do ich źródła.

## Włączanie cytowań

Należy ustawić `CitationsEnabled` na `True` w blokach zawartości dokumentów. Cytowania muszą być włączone dla WSZYSTKICH lub ŻADNEGO dokumentu w żądaniu.

```
TsgcAnthropicClass_Request_Content_Block *oDocBlock = new TsgcAnthropicClass_Request_Content_Block(NULL);
oDocBlock->ContentType = "document";
oDocBlock->SourceType = "base64";
oDocBlock->MediaType = "application/pdf";
oDocBlock->Data = sgcBase64Encode(LoadFileToBytes("report.pdf"));
oDocBlock->CitationsEnabled = true; // Enable citations
```

## Odczytywanie cytowań z odpowiedzi

Bloki treści tekstowej w odpowiedzi mogą zawierać tablicę `Citations` z odniesieniami do źródeł.

```
TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
try {
    for (int i = 0; i < Length(oResponse->Content); i++) {
        if (oResponse->Content[i]->ContentType == "text") {
            ShowMessage(oResponse->Content[i]->Text);

            // Process citations
            for (int j = 0; j < Length(oResponse->Content[i]->Citations); j++) {
                TsgcAnthropicClass_Response_Citation *oCitation = oResponse->Content[i]->Citations[j];
                ShowMessage(Format(" Citation [%s]: \"%s\"",
                    ARRAYOFCONST((oCitation->CitationType, oCitation->CitedText))));

                if (oCitation->CitationType == "page_location")
                    ShowMessage(Format(" Pages %d-%d of \"%s\"",
                        ARRAYOFCONST((oCitation->StartPageNumber, oCitation->EndPageNumber,
                            oCitation->DocumentTitle))));
            }
        }
    }
} __finally {
    oResponse->Free();
}
```

## Typy cytatów

- **char\_location:** Odwołanie na poziomie znaku w dokumentach tekstowych. Pola: `DocumentIndex`, `DocumentTitle`, `StartCharIndex`, `EndCharIndex`.
- **page\_location:** odwołanie do poziomu strony w dokumentach PDF. Pola: `DocumentIndex`, `DocumentTitle`, `StartPageNumber`, `EndPageNumber`.
- **content\_block\_location:** Odwołanie na poziomie bloku w niestandardowych dokumentach zawartości. Pola: `DocumentIndex`, `DocumentTitle`, `StartBlockIndex`, `EndBlockIndex`.
- **web\_search\_result\_location:** Odwołanie do wyników wyszukiwania w sieci. Pola: `Url`, `Title`, `EncryptedIndex`.

# Anthropic | Wyszukiwanie w sieci

Narzędzie Web Search umożliwia Claude'owi wyszukiwanie informacji w sieci w czasie rzeczywistym podczas rozmowy. Jest to wbudowane narzędzie po stronie serwera, hostowane i wykonywane automatycznie przez Anthropic.

## Prosty przykład

Użyj metody pomocniczej, aby utworzyć wiadomość z włączonym przeszukiwaniem sieci.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
ShowMessage(Anthropic->_CreateMessageWithWebSearch("claude-sonnet-4-20250514",
    "What are the latest news about Delphi programming?"));
```

## Zaawansowany przykład

Użyj typowanych klas, aby uzyskać pełną kontrolę nad parametrami narzędzia wyszukiwania w Internecie, takimi jak MaxUses.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    // Add web search tool
    TsgcAnthropicClass_Request_Tool *oTool = new TsgcAnthropicClass_Request_Tool();
    oTool->ToolType = "web_search_20250305";
    oTool->Name = "web_search";
    oTool->MaxUses = 5; // Max 5 searches per request
    oTools = oRequest->Tools;
    SetLength(oTools, 1);
    oTools[0] = oTool;
    oRequest->Tools = oTools;

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oMessage->Content = "Find the current price of Bitcoin.";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        for (int i = 0; i < oResponse->Content.Length; i++) {
            if (oResponse->Content[i]->ContentType == "text")
                ShowMessage(oResponse->Content[i]->Text);
        }
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oTool;
    delete oMessage;
    delete oRequest;
```



}

## Wbudowane typy narzędzi

- **web\_search\_20250305:** Wyszukiwanie w sieci po stronie serwera. Właściwości: Name ('web\_search'), MaxUses.
- **code\_execution\_20250825:** Izolowane wykonywanie kodu. Właściwości: Name ('code\_execution'), MaxUses.
- **computer\_20250124:** Korzystanie z komputera (mysz, klawiatura, zrzuty ekranu). Właściwości: Name ('computer'), DisplayWidthPx, DisplayHeightPx.

## Typy zawartości odpowiedzi

Gdy używane są wbudowane narzędzia, odpowiedź może zawierać dodatkowe typy bloków treści:

- **server\_tool\_use:** Wskazuje, że Claude wywołał wbudowane narzędzie. Pola: Id, Name, Input.
- **web\_search\_tool\_result:** Zawiera wyniki wyszukiwania w sieci. Pola: ToolUseId, Data (nieprzetworzona zawartość JSON).

# Anthropic | Strukturyzowane wyjścia

Structured Outputs wymuszają na Claude zwracanie odpowiedzi zgodnych ze schematem JSON. Gwarantuje to prawidłowe, parsowalne dane wyjściowe JSON zgodne z definicją schematu.

## Prosty przykład

Użyj metody pomocniczej do tworzenia wiadomości z wyjściem w schemacie JSON.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

String vSchema = "{\n\"type\": \"object\", \"properties\": {\n\"name\": {\n\"type\": \"string\"}, \"\n\"age\": {\n\"type\": \"integer\"}}, \"required\": [\n\"name\", \"age\"], \"\n\"additionalProperties\": false}";

ShowMessage(Anthropic->_CreateMessageJSON("claude-sonnet-4-20250514",
    "Extract the name and age: John is 30 years old.", vSchema));
```

## Zaawansowany przykład

Należy użyć typizowanych klas, aby połączyć ustrukturyzowane dane wyjściowe z parametrem effort.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    // JSON schema for structured output
    oRequest->OutputFormatSchema =
        "{\n\"type\": \"object\", \"properties\": {\n\"sentiment\": {\n\"type\": \"string\", \"\n\"enum\": [\n\"positive\", \"negative\", \"neutral\"], \"confidence\": \"\n\"type\": \"number\"}}, \"required\": [\n\"sentiment\", \"confidence\"], \"\n\"additionalProperties\": false}";

    // Set effort level (low, medium, high, max)
    oRequest->Effort = "medium";

    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oMessage->Content = "Analyze the sentiment: I love this product!";
    oMessages = oRequest->Messages;
    SetLength(oMessages, 1);
    oMessages[0] = oMessage;
    oRequest->Messages = oMessages;

    TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
    try {
        for (int i = 0; i < oResponse->Content.Length; i++) {
            if (oResponse->Content[i]->ContentType == "text")
                ShowMessage(oResponse->Content[i]->Text);
        }
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oMessage;
```

```
    delete oRequest;
}
```

## Ścisłe użycie narzędzi

Włącz tryb ścisły dla definicji narzędzi, aby zagwarantować, że dane wejściowe narzędzia dokładnie odpowiadają schematowi input\_schema.

```
TsgcAnthropicClass_Request_Tool *oTool = new TsgcAnthropicClass_Request_Tool();
oTool->Name = "get_weather";
oTool->Description = "Get the current weather for a location";
oTool->Strict = true; // Guarantee schema conformance
oTool->InputSchema = "{\"type\":\"object\",\"properties\":{\"location\":\""
    "\"type\":\"string\"}},\"required\":[\"location\"],\"additionalProperties\":false}";
```

## Właściwości

- **OutputFormatSchema:** Ciąg JSON Schema. Tekst odpowiedzi Claude'a będzie prawidłowym kodem JSON zgodnym z tym schematem.
- **Effort:** Steruje jakością danych wyjściowych w stosunku do kosztu. Wartości: 'low', 'medium', 'high' (domyślnie), 'max' (tylko Opus 4.6).
- **Strict:** (dla narzędzi) Gdy True, dane wejściowe narzędzia gwarantują zgodność z input\_schema.

# Anthropic | Files API

API plików umożliwia przesyłanie, wyświetlanie listy, pobieranie, pobieranie do lokalnego systemu i usuwanie plików. Przesłane pliki mogą być odwoływane w wiadomościach przy użyciu bloków zawartości dokumentów ze źródłowym typem pliku.

**Uwaga:** Files API wymaga nagłówka beta `files-api-2025-04-14`. Ustaw go we właściwości `AnthropicOptions.BetaHeaders`.

## Prześlij plik

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
Anthropic->AnthropicOptions->BetaHeaders = "files-api-2025-04-14";

TsgcAnthropicClass_Response_File *oFile = Anthropic->UploadFile("C:\\documents\\report.pdf");
try {
    ShowMessage("File ID: " + oFile->Id);
} __finally {
    delete oFile;
}
```

## Lista plików

```
TsgcAnthropicClass_Response_FileList *oList = Anthropic->ListFiles(50);
try {
    for (int i = 0; i < oList->Data.Length; i++)
        ShowMessage(oList->Data[i]->Id + " - " + oList->Data[i]->Filename +
            " (" + IntToStr(oList->Data[i]->SizeBytes) + " bytes");
    ShowMessage("Has more: " + BoolToStr(oList->HasMore, true));
} __finally {
    delete oList;
}
```

## Użyj pliku w wiadomościach

Odwoływać się do przesłanych plików za pomocą bloku zawartości dokumentu z `SourceType` ustawionym na 'file'.

```
TsgcAnthropicClass_Request_Content_Block *oDocBlock = new TsgcAnthropicClass_Request_Content_Block();
oDocBlock->ContentType = "document";
oDocBlock->SourceType = "file";
oDocBlock->FileId = "file_abc123"; // ID from UploadFile

TsgcAnthropicClass_Request_Content_Block *oTextBlock = new TsgcAnthropicClass_Request_Content_Block();
oTextBlock->ContentType = "text";
oTextBlock->Text = "Summarize this document.";
```

## Usuń plik

```
TsgcAnthropicClass_Response_File *oDeleted = Anthropic->DeleteFile("file_abc123");
try {
    ShowMessage("Deleted: " + oDeleted->Id);
} __finally {
    delete oDeleted;
}
```

}

## Metody API

- **UploadFile:** Przesyła plik z lokalnej ścieżki. Zwraca metadane pliku (Id, Filename, MimeType, SizeBytes).
- **ListFiles:** Wyświetla listę wszystkich przesłanych plików z obsługą stronicowania (Limit, AfterId, BeforeId).
- **GetFile:** Pobiera metadane dla określonego pliku na podstawie jego identyfikatora.
- **DownloadFile:** Pobiera zawartość pliku (możliwe jest pobieranie tylko plików utworzonych przez narzędzie do wykonywania kodu).
- **DeleteFile:** Trwale usuwa plik na podstawie identyfikatora.

# Anthropic | MCP Connector

Konektor MCP (Model Context Protocol) umożliwia Claude dostęp do narzędzi z zewnętrznych serwerów MCP. Pozwala to na integrację z usługami zewnętrznymi i niestandardowymi dostawcami narzędzi.

**Uwaga:** Łącznik MCP wymaga nagłówka beta `mcp-client-2025-11-20`. Należy ustawić go we właściwości **AnthropicOptions.BetaHeaders**.

## Prosty przykład

Użyj metody pomocniczej, aby utworzyć wiadomość z serwerem MCP.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
Anthropic->AnthropicOptions->BetaHeaders = "mcp-client-2025-11-20";

ShowMessage(Anthropic->_CreateMessageWithMCP("claude-sonnet-4-20250514",
    "What tools are available?",
    "https://my-mcp-server.example.com/sse",
    "my-mcp-server"));
```

## Zaawansowany przykład

Należy używać typizowanych klas do pełnej kontroli nad konfiguracją serwera MCP, w tym uwierzytelnianiem.

```
TsgcHTTP_API_Anthropic *Anthropic = new TsgcHTTP_API_Anthropic(NULL);
Anthropic->AnthropicOptions->ApiKey = "API_KEY";
Anthropic->AnthropicOptions->BetaHeaders = "mcp-client-2025-11-20";

TsgcAnthropicClass_Request_Messages *oRequest = new TsgcAnthropicClass_Request_Messages();
try {
    oRequest->Model = "claude-sonnet-4-20250514";
    oRequest->MaxTokens = 4096;

    // Configure MCP server
    TsgcAnthropicClass_Request_MCPServer *oServer = new TsgcAnthropicClass_Request_MCPServer();
    oServer->ServerType = "url";
    oServer->Url = "https://my-mcp-server.example.com/sse";
    oServer->Name = "my-server";
    oServer->AuthorizationToken = "OAUTH_TOKEN"; // Optional auth
    oServers = oRequest->MCPServers;
    SetLength(oServers, 1);
    oServers[0] = oServer;
    oRequest->MCPServers = oServers;

    // Add MCP toolset
    TsgcAnthropicClass_Request_Tool *oTool = new TsgcAnthropicClass_Request_Tool();
    oTool->ToolType = "mcp_toolset";
    oTool->MCPServerName = "my-server";
    oTools = oRequest->Tools;
    SetLength(oTools, 1);
    oTools[0] = oTool;
    oRequest->Tools = oTools;

    // Add user message
    TsgcAnthropicClass_Request_Message *oMessage = new TsgcAnthropicClass_Request_Message();
    oMessage->Role = "user";
    oMessage->Content = "Search for recent news about AI.";
```

```

oMessages = oRequest->Messages;
SetLength(oMessages, 1);
oMessages[0] = oMessage;
oRequest->Messages = oMessages;

TsgcAnthropicClass_Response_Messages *oResponse = Anthropic->CreateMessage(oRequest);
try {
    for (int i = 0; i < oResponse->Content.Length; i++) {
        if (oResponse->Content[i]->ContentType == "text")
            ShowMessage(oResponse->Content[i]->Text);
        else if (oResponse->Content[i]->ContentType == "mcp_tool_use")
            ShowMessage("MCP tool call: " + oResponse->Content[i]->Name +
                " on " + oResponse->Content[i]->ServerName);
    }
} __finally {
    delete oResponse;
}
} __finally {
    delete oServer;
    delete oTool;
    delete oMessage;
    delete oRequest;
}

```

## Właściwości serwera MCP

- **ServerType:** Obecnie obsługiwany jest tylko typ 'url'.
- **Url:** Adres HTTPS serwera MCP.
- **Name:** unikalny identyfikator tego serwera. Musi odpowiadać wartości MCPServerName w zestawie narzędzi.
- **AuthorizationToken:** Opcjonalny token Bearer OAuth dla serwerów wymagających uwierzytelnienia.

## Typy zawartości odpowiedzi

- **mcp\_tool\_use:** Claude wywołał narzędzie MCP. Pola: Id, Name, ServerName, Input.
- **mcp\_tool\_result:** Wynik wykonania narzędzia MCP. Pola: ToolUseId, IsError, Data (zawartość).

# Gemini

Google Gemini to rodzina multimodalnych modeli AI opracowanych przez Google DeepMind. Modele Gemini obsługują generowanie tekstu, przetwarzanie obrazów, strukturyzowane dane wyjściowe, osadzenia i korzystanie z narzędzi, oferując zaawansowane możliwości do tworzenia aplikacji opartych na AI.

Biblioteka `sgcWebSockets` udostępnia komponent Delphi **TsgcHTTP\_API\_Gemini** do integracji z Gemini API.

## Gemini API

Gemini API zapewnia dostęp do modeli Google Gemini do tworzenia aplikacji opartych na AI. API obsługuje generowanie treści, rozumienie obrazów (wizję), ustrukturyzowane dane wyjściowe JSON, strumieniowanie, liczenie tokenów, osadzenia, użycie narzędzi (wywoływanie funkcji) i wyświetlanie listy modeli.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Gemini](#)
- **Vision**
  - [Przykłady Gemini Vision](#)
- **Modele**
  - [Przykłady modeli Gemini](#)
- **Ustrukturyzowane dane wyjściowe**
  - [Przykłady ustrukturyzowanych danych wyjściowych Gemini](#)
- **Token Counting**
  - [Gemini Token Counting Examples](#)
- **Embeddingi**
  - [Przykłady Gemini Embeddings](#)
- **Tool Use**
  - [Przykłady użycia narzędzi Gemini](#)

## Konfiguracja

Gemini API używa kluczy API do uwierzytelniania. Odwiedź stronę [Klucze API](#) w Google AI Studio, aby pobrać klucz API używany w żądaniach.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi zostać skonfigurowany we właściwości **GeminiOptions.ApiKey** komponentu.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "YOUR_API_KEY";
```



## Wiadomości

Wyślij treść do modelu Gemini i odbierz wygenerowane odpowiedzi. Model generuje następną wiadomość na podstawie podanych danych wejściowych.

- **\_CreateContent:** Tworzy treść przy użyciu określonego modelu i monitu użytkownika.
  - **Model:** Model do użycia (np. gemini-2.0-flash).
  - **Message:** Treść wiadomości użytkownika.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.
- **\_CreateContentWithSystem:** Tworzy treść z instrukcją systemową.
  - **Model:** Model do użycia.
  - **System:** Instrukcja systemowa określająca zachowanie modelu.
  - **Message:** Treść wiadomości użytkownika.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.
- **\_CreateContentStream:** Tworzy treść z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia OnHTTPAPISSE.

## Vision

Modele Gemini mogą rozumieć obrazy przekazywane jako zawartość zakodowana w base64 wraz z monitami tekstowymi.

- **\_CreateVisionContent:** Wysyła obraz wraz z monitem tekstowym.
  - **Model:** Model do użycia.
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.
  - **Base64:** Dane obrazu zakodowane w Base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.

## Wyjścia strukturyzowane

Generuj ustrukturyzowane dane wyjściowe JSON z modelu Gemini, dostarczając schemat JSON definiujący oczekiwany format odpowiedzi.

- **\_CreateContentJSON:** Tworzy treść ze strukturyzowanym wyjściem JSON.
  - **Model:** Model do użycia.
  - **Message:** Treść wiadomości użytkownika.
  - **Schema:** Schemat JSON definiujący oczekiwaną strukturę wyjściową.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.

## Modele

Wyświetlenie listy i pobranie szczegółów dostępnych modeli Gemini.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.
- **\_GetModel:** Pobiera szczegóły dla określonego modelu.
  - **ModelId:** Identyfikator modelu do pobrania.

## Zliczanie tokenów

Zlicza liczbę tokenów w wiadomości przed wysłaniem jej do modelu.

- **\_CountTokens:** Zlicza tokeny dla wiadomości.
  - **Model:** Model używany do tokenizacji.

- **Message:** Treść tekstowa, dla której zliczane są tokeny.

## Osadzenia

Generowanie wektorowych osadzeń dla treści tekstowych przy użyciu modeli Gemini.

- **\_EmbedContent:** Generuje osadzenia dla tekstu.
  - **Model:** Model używany do generowania osadzeń.
  - **Text:** Treść tekstowa, dla której generowane są osadzenia.

# TsgcHTTP\_API\_Gemini

Klient REST Google Gemini do generowania treści, liczenia tokenów i tworzenia osadzeń.

## Wprowadzenie

Klient REST Google Gemini do generowania treści, liczenia tokenów i tworzenia osadzeń.

## Wprowadzenie

Klient REST Google Gemini do generowania treści, liczenia tokenów i tworzenia osadzeń.

Skonfiguruj właściwość **GeminiOptions**, podając klucz API oraz opcjonalne ustawienia HTTP/ponowień/logowania, a następnie wywołaj typowane metody, takie jak `CreateMessage/CreateChatCompletion`, przekazując obiekt żądania i otrzymując typowaną klasę odpowiedzi.

Informacje ogólne na temat dostawcy zawiera przewodnik [Gemini](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 3 publiczne metody.
- Zdarzenia — 2 zdarzenia do obsługi.

# DeepSeek

DeepSeek to chińska firma z obszaru AI, skupiająca się na budowaniu wydajnych modeli językowych o otwartym kodzie źródłowym. Modele te wyróżniają się w kodowaniu, rozumowaniu i zadaniach ogólnych, oferując wysoką wydajność w konkurencyjnych cenach.

Biblioteka `sgcWebSockets` udostępnia komponent Delphi **TsgcHTTP\_API\_DeepSeek** do interakcji z DeepSeek API.

## DeepSeek API

DeepSeek API zapewnia dostęp do modeli DeepSeek umożliwiające tworzenie aplikacji opartych na AI. API obsługuje generowanie tekstu, wizję (rozumienie obrazów), strumieniowanie oraz wyświetlanie listy modeli. API stosuje format zgodny z OpenAI, co ułatwia integrację.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości DeepSeek](#)
- **Vision**
  - [Przykłady DeepSeek Vision](#)
- **Modele**
  - [Przykłady modeli DeepSeek](#)

## Konfiguracja

DeepSeek API używa kluczy API do uwierzytelniania. Aby pobrać klucz API, należy odwiedzić stronę [kluczy API](#) na platformie DeepSeek.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi być skonfigurowany we właściwości **DeepSeekOptions.ApiKey** komponentu.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);  
DeepSeek->DeepSeekOptions->ApiKey = "YOUR_API_KEY";
```

## Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

- **\_CreateMessage:** Tworzy wiadomość z określonym modelem i monitem użytkownika.
  - **Model:** Model do użycia (np. `deepseek-chat`).
  - **Message:** Treść wiadomości użytkownika.
- **\_CreateMessageWithSystem:** Tworzy wiadomość z monitem systemowym.
  - **System:** Monit systemowy określający zachowanie asystenta.
- **\_CreateMessageStream:** Tworzy wiadomość z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia `OnHTTPAPISSE`.

## Vision

DeepSeek potrafi analizować obrazy przekazane jako zawartość zakodowana w base64 wewnątrz wiadomości.

- **\_CreateVisionMessage:** Wysyła obraz z monitem tekstowym.
  - **ImageBase64:** Dane obrazu zakodowane w base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.

## Modele

Wyświetla dostępne modele DeepSeek.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.

# TsgcHTTP\_API\_DeepSeek

Klient REST API DeepSeek do uzupełniania czatu.

## Wprowadzenie

Klient REST API DeepSeek do uzupełniania czatu.

## Wprowadzenie

Klient REST API DeepSeek do uzupełniania czatu.

Należy skonfigurować właściwość **DeepSeekOptions** za pomocą klucza API i opcjonalnych ustawień HTTP/ponawiania/dziennika, a następnie wywołać typowane metody, takie jak `CreateMessage/CreateChatCompletion`, z obiektem żądania i otrzymać typowaną klasę odpowiedzi.

Ogólne informacje o dostawcy znajdują się w przewodniku [DeepSeek](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 1 metoda publiczna.
- Zdarzenia — 2 zdarzenia dostępne do obsługi.

# DeepSeek | Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

## Prosty przykład

Wysyłanie wiadomości Hello do DeepSeek.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);
DeepSeek->DeepSeekOptions->ApiKey = "API_KEY";
ShowMessage(DeepSeek->_CreateMessage("deepseek-chat", "Hello!"));
```

## Przykład monitu systemowego

Wyślij wiadomość z monitem systemowym, aby kontrolować zachowanie DeepSeek.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);
DeepSeek->DeepSeekOptions->ApiKey = "API_KEY";
ShowMessage(DeepSeek->_CreateMessageWithSystem("deepseek-chat",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);
DeepSeek->DeepSeekOptions->ApiKey = "API_KEY";
DeepSeek->OnHTTPAPISSE = OnSSEEvent;
DeepSeek->_CreateMessageStream("deepseek-chat", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    // aEvent contains the event type
    // aData contains the JSON data for this event
    Memo1->Lines->Add(aData);
}
```

# DeepSeek | Vision

DeepSeek potrafi rozumieć i analizować obrazy. Obrazy można przesyłać jako dane zakodowane w formacie base64 wewnątrz bloków treści.

## Przykład wizji

Wyślij obraz z pytaniem do DeepSeek z prośbą o jego opisanie.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);
DeepSeek->DeepSeekOptions->ApiKey = "API_KEY";

// Load image and encode to base64
TFileStream *oStream = new TFileStream("photo.jpg", fmOpenRead);
try {
    TBytesStream *oBytes = new TBytesStream();
    try {
        oBytes->CopyFrom(oStream, 0);
        vImageBase64 = EncodeBase64(oBytes->Memory, oBytes->Size);
    } __finally {
        oBytes->Free();
    }
} __finally {
    oStream->Free();
}

ShowMessage(DeepSeek->_CreateVisionMessage("deepseek-chat",
    "Describe this image", vImageBase64, "image/jpeg"));
```



# DeepSeek | Modele

Wyświetla dostępne modele DeepSeek.

## Lista modeli

Wyświetla listę wszystkich dostępnych modeli DeepSeek.

```
TsgcHTTP_API_DeepSeek *DeepSeek = new TsgcHTTP_API_DeepSeek(NULL);  
DeepSeek->DeepSeekOptions->ApiKey = "API_KEY";  
ShowMessage(DeepSeek->_GetModels());
```

# Gemini

Google Gemini to rodzina multimodalnych modeli AI opracowanych przez Google DeepMind. Modele Gemini obsługują generowanie tekstu, przetwarzanie obrazów, strukturyzowane dane wyjściowe, osadzenia i korzystanie z narzędzi, oferując zaawansowane możliwości do tworzenia aplikacji opartych na AI.

Biblioteka sgcWebSockets udostępnia komponent Delphi **TsgcHTTP\_API\_Gemini** do integracji z Gemini API.

## Gemini API

Gemini API zapewnia dostęp do modeli Google Gemini do tworzenia aplikacji opartych na AI. API obsługuje generowanie treści, rozumienie obrazów (wizję), ustrukturyzowane dane wyjściowe JSON, strumieniowanie, liczenie tokenów, osadzenia, użycie narzędzi (wywoływanie funkcji) i wyświetlanie listy modeli.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Gemini](#)
- **Vision**
  - [Przykłady Gemini Vision](#)
- **Modele**
  - [Przykłady modeli Gemini](#)
- **Ustrukturyzowane dane wyjściowe**
  - [Przykłady ustrukturyzowanych danych wyjściowych Gemini](#)
- **Token Counting**
  - [Gemini Token Counting Examples](#)
- **Embeddingi**
  - [Przykłady Gemini Embeddings](#)
- **Tool Use**
  - [Przykłady użycia narzędzi Gemini](#)

## Konfiguracja

Gemini API używa kluczy API do uwierzytelniania. Odwiedź stronę [Klucze API](#) w Google AI Studio, aby pobrać klucz API używany w żądaniach.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi zostać skonfigurowany we właściwości **GeminiOptions.ApiKey** komponentu.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "YOUR_API_KEY";
```

## Wiadomości

Wyślij treść do modelu Gemini i odbierz wygenerowane odpowiedzi. Model generuje następną wiadomość na podstawie podanych danych wejściowych.

- **\_CreateContent:** Tworzy treść przy użyciu określonego modelu i monitu użytkownika.
  - **Model:** Model do użycia (np. gemini-2.0-flash).
  - **Message:** Treść wiadomości użytkownika.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.
- **\_CreateContentWithSystem:** Tworzy treść z instrukcją systemową.
  - **Model:** Model do użycia.
  - **System:** Instrukcja systemowa określająca zachowanie modelu.
  - **Message:** Treść wiadomości użytkownika.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.
- **\_CreateContentStream:** Tworzy treść z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia OnHTTPAPISSE.

## Vision

Modele Gemini mogą rozumieć obrazy przekazywane jako zawartość zakodowana w base64 wraz z monitami tekstowymi.

- **\_CreateVisionContent:** Wysyła obraz wraz z monitem tekstowym.
  - **Model:** Model do użycia.
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.
  - **Base64:** Dane obrazu zakodowane w Base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.

## Wyjścia strukturyzowane

Generuj ustrukturyzowane dane wyjściowe JSON z modelu Gemini, dostarczając schemat JSON definiujący oczekiwany format odpowiedzi.

- **\_CreateContentJSON:** Tworzy treść ze strukturyzowanym wyjściem JSON.
  - **Model:** Model do użycia.
  - **Message:** Treść wiadomości użytkownika.
  - **Schema:** Schemat JSON definiujący oczekiwaną strukturę wyjściową.
  - **MaxOutputTokens:** Maksymalna liczba tokenów do wygenerowania.

## Modele

Wyświetlenie listy i pobranie szczegółów dostępnych modeli Gemini.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.
- **\_GetModel:** Pobiera szczegóły dla określonego modelu.
  - **ModelId:** Identyfikator modelu do pobrania.

## Zliczanie tokenów

Zlicza liczbę tokenów w wiadomości przed wysłaniem jej do modelu.

- **\_CountTokens:** Zlicza tokeny dla wiadomości.
  - **Model:** Model używany do tokenizacji.

- **Message:** Treść tekstowa, dla której zliczane są tokeny.

## Osadzenia

Generowanie wektorowych osadzeń dla treści tekstowych przy użyciu modeli Gemini.

- **\_EmbedContent:** Generuje osadzenia dla tekstu.
  - **Model:** Model używany do generowania osadzeń.
  - **Text:** Treść tekstowa, dla której generowane są osadzenia.

# Gemini | Wiadomości

Wyślij odpowiedź do modelu Gemini i odbierz wygenerowaną treść. API Gemini używa punktu końcowego generateContent do przetwarzania danych wejściowych tekstowych i zwracania odpowiedzi modelu.

## Prosty przykład

Wysyłanie wiadomości Hello do Gemini.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_CreateContent("gemini-2.0-flash", "Hello!"));
```

## Przykład instrukcji systemowej

Wyślij wiadomość z instrukcją systemową sterującą zachowaniem Gemini.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_CreateContentWithSystem("gemini-2.0-flash",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
Gemini->OnHTTPAPISSE = OnSSEEvent;
Gemini->_CreateContentStream("gemini-2.0-flash", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    Memo1->Lines->Add(aData);
}
```

## Zaawansowany przykład

Należy używać typowanych klas żądań i odpowiedzi w celu pełnej kontroli nad parametrami generowania zawartości.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

TsgcGeminiClass_Request_GenerateContent *oRequest = new TsgcGeminiClass_Request_GenerateContent();
try {
    oRequest->Model = "gemini-2.0-flash";
    oRequest->MaxOutputTokens = 4096;
    oRequest->Temperature = 0.7;
    oRequest->SystemInstruction = "You are a creative writer.";
```

```
SetLength(oContents, 1);
oContents[0] = new TsgcGeminiClass_Request_Content();
oContents[0]->Role = "user";
SetLength(oParts, 1);
oParts[0] = new TsgcGeminiClass_Request_Part();
oParts[0]->Text = "Write a haiku about programming.";
oContents[0]->Parts = oParts;
oRequest->Contents = oContents;

TsgcGeminiClass_Response_GenerateContent *oResponse = Gemini->CreateContent(oRequest);
try {
    if (Length(oResponse->Candidates) > 0)
        if (Length(oResponse->Candidates[0]->Parts) > 0)
            ShowMessage(oResponse->Candidates[0]->Parts[0]->Text);
} __finally {
    delete oResponse;
}
} __finally {
    delete oParts[0];
    delete oContents[0];
    delete oRequest;
}
```

# Gemini | Vision

Gemini potrafi rozumieć i analizować obrazy. Obrazy można przesyłać jako dane zakodowane w base64 w ramach części zawartości.

## Przykład wizji

Wyślij obraz zakodowany w base64 do Gemini w celu analizy.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

// Load image and encode to base64
TFileStream *oStream = new TFileStream("photo.jpg", fmOpenRead);
try {
    TBytesStream *oBytes = new TBytesStream();
    try {
        oBytes->CopyFrom(oStream, 0);
        vImageBase64 = EncodeBase64(oBytes->Memory, oBytes->Size);
    } __finally {
        oBytes->Free();
    }
} __finally {
    oStream->Free();
}

ShowMessage(Gemini->_CreateVisionContent("gemini-2.0-flash",
    "Describe this image", vImageBase64, "image/jpeg"));
```

# Gemini | Modele

Wyświetlanie listy i pobieranie informacji o dostępnych modelach Gemini.

## Lista modeli

Wyświetla listę wszystkich dostępnych modeli.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_GetModels());
```

## Pobierz model

Pobierz szczegóły określonego modelu.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_GetModel("gemini-2.0-flash"));
```

## Zaawansowany przykład

Należy używać typowanych klas odpowiedzi, aby uzyskać pełną kontrolę nad danymi modelu.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

TsgcGeminiClass_Response_Models *oModels = Gemini->GetModels();
try {
    for (int i = 0; i < oModels->Models.Length; i++)
        ShowMessage(oModels->Models[i]->DisplayName + " - " + oModels->Models[i]->Description);
} __finally {
    delete oModels;
}
```



# Gemini | Ustrukturyzowane dane wyjściowe

Gemini może zwracać odpowiedzi w ustrukturyzowanym formacie JSON zgodnym ze zdefiniowanym schematem. Gwarantuje to prawidłowe, możliwe do przetworzenia wyjście JSON pasujące do definicji schematu.

## Prosty przykład

Użyj metody pomocniczej, aby tworzyć treści z wyjściem w schemacie JSON.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

String vSchema = "{\"type\":\"object\",\"properties\":{\"name\":{\"type\":\"string\"}, \"age\":{\"type\":\"integer\"}}, \"required\": [\"name\", \"age\"]}";

ShowMessage(Gemini->_CreateContentJSON("gemini-2.0-flash",
    "Extract the name and age: John is 30 years old.", vSchema));
```

## Zaawansowany przykład

Używać typizowanych klas do konfigurowania strukturyzowanych danych wyjściowych z ResponseMimeType i ResponseSchema.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

TsgcGeminiClass_Request_GenerateContent *oRequest = new TsgcGeminiClass_Request_GenerateContent();
try {
    oRequest->Model = "gemini-2.0-flash";
    oRequest->MaxOutputTokens = 4096;
    oRequest->ResponseMimeType = "application/json";
    oRequest->ResponseSchema =
        "{\"type\":\"object\",\"properties\":{\"sentiment\":{\"type\":\"string\"}, \"confidence\":{\"type\":\"number\"}}, \"required\": [\"sentiment\", \"confidence\"]}";

    SetLength(oContents, 1);
    oContents[0] = new TsgcGeminiClass_Request_Content();
    oContents[0]->Role = "user";
    SetLength(oParts, 1);
    oParts[0] = new TsgcGeminiClass_Request_Part();
    oParts[0]->Text = "Analyze the sentiment: I love this product!";
    oContents[0]->Parts = oParts;
    oRequest->Contents = oContents;

    TsgcGeminiClass_Response_GenerateContent *oResponse = Gemini->CreateContent(oRequest);
    try {
        if (Length(oResponse->Candidates) > 0)
            if (Length(oResponse->Candidates[0]->Parts) > 0)
                ShowMessage(oResponse->Candidates[0]->Parts[0]->Text);
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oParts[0];
    delete oContents[0];
    delete oRequest;
}
```

## Właściwości

- **ResponseMimeType:** Ustaw na 'application/json', aby włączyć ustrukturyzowane dane wyjściowe JSON.
- **ResponseSchema:** Ciąg znaków JSON Schema określający oczekiwaną strukturę wyjściową.

# Gemini | Liczenie tokenów

Zlicza tokeny w wiadomości przed jej wysłaniem do API. Pozwala to oszacować koszty i upewnić się, że wiadomości mieszczą się w limitach modelu.

## Prosty przykład

Zlicz tokeny dla wiadomości.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_CountTokens("gemini-2.0-flash", "Hello, how are you?"));
```

## Zaawansowany przykład

Użyj typowanych klas żądań/odpowiedzi, aby uzyskać pełną kontrolę nad parametrami zliczania tokenów.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

TsgcGeminiClass_Request_CountTokens *oRequest = new TsgcGeminiClass_Request_CountTokens();
try {
    oRequest->Model = "gemini-2.0-flash";
    SetLength(oContents, 1);
    oContents[0] = new TsgcGeminiClass_Request_Content();
    oContents[0]->Role = "user";
    SetLength(oParts, 1);
    oParts[0] = new TsgcGeminiClass_Request_Part();
    oParts[0]->Text = "Explain quantum computing in simple terms.";
    oContents[0]->Parts = oParts;
    oRequest->Contents = oContents;

    TsgcGeminiClass_Response_CountTokens *oResponse = Gemini->CountTokens(oRequest);
    try {
        ShowMessage("Total tokens: " + IntToStr(oResponse->TotalTokens));
    } __finally {
        delete oResponse;
    }
} __finally {
    delete oParts[0];
    delete oContents[0];
    delete oRequest;
}
```

# Gemini | Embeddings

Generuje osadzenia wektorowe dla treści tekstowych. Osadzenia przechwytyją semantyczne znaczenie tekstu i mogą być używane do wyszukiwania podobieństw, grupowania i innych zadań NLP.

## Prosty przykład

Generowanie osadzeń dla tekstu.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";
ShowMessage(Gemini->_EmbedContent("text-embedding-004", "Hello world"));
```

## Zaawansowany przykład

Należy używać typowanych klas odpowiedzi w celu pełnej kontroli nad danymi osadzania.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

TsgcGeminiClass_Response_Embedding *oEmbedding = Gemini->EmbedContent("text-embedding-004", "Hello world");
try {
    ShowMessage("Dimensions: " + IntToStr(oEmbedding->Values.Length));
    for (int i = 0; i < oEmbedding->Values.Length; i++)
        ShowMessage(FloatToStr(oEmbedding->Values[i]));
} __finally {
    delete oEmbedding;
}
```

# Gemini | Użycie narzędzi

Gemini obsługuje wywoływanie funkcji (użycie narzędzi), co pozwala definiować funkcje, które model może wywoływać podczas konwersacji. Gdy Gemini zdecyduje się wywołać funkcję, zwraca część `functionCall` w swojej odpowiedzi. Następnie należy wykonać funkcję i odesłać wynik.

## Przebieg wywołania funkcji

1. Definiuje deklaracje funkcji z nazwą, opisem i parametrami (JSON Schema).
2. Wysyła żądanie generowania zawartości z deklaracjami funkcji.
3. Gemini odpowiada częścią **functionCall** zawierającą nazwę funkcji i argumenty.
4. Wykonaj funkcję z podanymi argumentami.
5. Wyślij nowe żądanie z częścią **functionResponse** zawierającą wynik.
6. Gemini odpowiada z ostateczną odpowiedzią.

## Przykład

Zdefiniuj funkcję pogodową i obsłuż wywołanie funkcji.

```
TsgcHTTP_API_Gemini *Gemini = new TsgcHTTP_API_Gemini(NULL);
Gemini->GeminiOptions->ApiKey = "API_KEY";

// Step 1: Create request with function declarations
TsgcGeminiClass_Request_GenerateContent *oRequest = new TsgcGeminiClass_Request_GenerateContent();
try {
    oRequest->Model = "gemini-2.0-flash";
    oRequest->MaxOutputTokens = 4096;

    // Define user message
    SetLength(oContents, 1);
    oContents[0] = new TsgcGeminiClass_Request_Content();
    oContents[0]->Role = "user";
    SetLength(oParts, 1);
    oParts[0] = new TsgcGeminiClass_Request_Part();
    oParts[0]->Text = "What is the weather in San Francisco?";
    oContents[0]->Parts = oParts;
    oRequest->Contents = oContents;

    // Define function
    SetLength(oFunctions, 1);
    oFunctions[0] = new TsgcGeminiClass_Request_FunctionDeclaration();
    oFunctions[0]->Name = "get_weather";
    oFunctions[0]->Description = "Get the current weather in a given location";
    oFunctions[0]->Parameters =
        "{\"type\":\"object\",\"properties\":{\"location\":{\"type\":\"string\", \"description\":\"The city and state\"}},\"required\":[\"location\"]}";
    oRequest->FunctionDeclarations = oFunctions;

    // Step 2: Send request
    TsgcGeminiClass_Response_GenerateContent *oResponse = Gemini->CreateContent(oRequest);
    try {
        // Step 3: Check for function call
        if (Length(oResponse->Candidates) > 0) {
            if (Length(oResponse->Candidates[0]->Parts) > 0) {
                if (oResponse->Candidates[0]->Parts[0]->FunctionCallName != "") {
                    vFunctionName = oResponse->Candidates[0]->Parts[0]->FunctionCallName;
                    vFunctionArgs = oResponse->Candidates[0]->Parts[0]->FunctionCallArgs;
                    // Step 4: Execute function and get result
                    vResult = "72 degrees and sunny";
                }
            }
        }
    }
```

```
    }  
  }  
  } __finally {  
    delete oResponse;  
  }  
} __finally {  
  delete oParts[0];  
  delete oContents[0];  
  delete oFunctions[0];  
  delete oRequest;  
}
```

## Właściwości

- **FunctionDeclarations:** Tablica obiektów TsgcGeminiClass\_Request\_FunctionDeclaration definiująca dostępne funkcje.
  - **Name:** Nazwa funkcji.
  - **Description:** Opis działania funkcji.
  - **Parameters:** Ciąg JSON Schema definiujący parametry funkcji.
- **ToolChoice:** Steruje zachowaniem wywoływania funkcji. Należy ustawić tryb (np. „AUTO”, „ANY”, „NONE”).

# Ollama

Ollama to narzędzie open source służące do lokalnego uruchamiania dużych modeli językowych. Obsługuje szeroką gamę modeli, w tym Llama, Mistral, Gemma, Phi i wiele innych, umożliwiając lokalną inferencję AI bez konieczności korzystania z chmurowego dostępu do API.

Biblioteka `sgcWebSockets` dostarcza komponent Delphi **TsgcHTTP\_API\_Ollama** do interakcji z API Ollama.

## Ollama API

Ollama API zapewnia dostęp do lokalnie uruchomionych modeli dla aplikacji opartych na AI. API obsługuje generowanie tekstu, strumieniowanie, zarządzanie modelami oraz osadzanie. Domyślnie nie jest wymagany klucz API, ponieważ modele działają lokalnie. Host i port są konfigurowalne.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Ollama](#)
- **Modele**
  - [Przykłady modeli Ollama](#)
- **Osadzenia**
  - [Przykłady osadzeń Ollama](#)

## Konfiguracja

Ollama działa lokalnie i domyślnie nasłuchuje pod adresem **http://localhost:11434**. Host należy skonfigurować we właściwości **OllamaOptions.Host**. Klucz API jest opcjonalny i wymagany tylko w przypadku skonfigurowania uwierzytelniania w instancji Ollama.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";
```

## Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

- **\_CreateMessage:** Tworzy wiadomość z użyciem określonego modelu i monitu użytkownika.
  - **Model:** Model do użycia (np. llama3).
  - **Message:** Treść wiadomości użytkownika.
- **\_CreateMessageWithSystem:** Tworzy wiadomość z monitem systemowym.
  - **System:** Monit systemowy określający zachowanie asystenta.
- **\_CreateMessageStream:** Tworzy wiadomość z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia `OnHTTPAPISSE`.

## Modele

Zarządzanie lokalnie dostępnymi modelami i ich odpytywanie.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.

- **\_GetTags:** Wyświetla wszystkie lokalnie dostępne tagi modeli.
- **\_ShowModel:** Pobiera szczegółowe informacje o konkretnym modelu.
  - **Model:** Nazwa modelu do zapytania.
- **\_PullModel:** Pobiera model z biblioteki modeli Ollama.
  - **Model:** Nazwa modelu do pobrania.
- **\_DeleteModel:** Usuwa lokalnie dostępny model.
  - **Model:** Nazwa modelu do usunięcia.

## Osadzenia

Generuj wektorowe osadzenia z tekstu wejściowego przy użyciu lokalnie uruchomionych modeli.

- **\_CreateEmbeddings:** Generuje osadzenia dla podanego tekstu.
  - **Model:** Model używany do generowania osadzeń (np. llama3).
  - **Input:** Tekst, dla którego mają być wygenerowane osadzenia.



# TsgcHTTP\_API\_Ollama

Klient REST API Ollama do lokalnych uzupełnień czatu i osadzeń.

## Wprowadzenie

Klient REST API Ollama do lokalnych uzupełnień czatu i osadzeń.

## Wprowadzenie

Klient REST API Ollama do lokalnych uzupełnień czatu i osadzeń.

Skonfiguruj właściwość **OllamaOptions** przy użyciu klucza API oraz opcjonalnych ustawień HTTP/po-  
nawiania/dziennika, a następnie wywołaj typowane metody takie jak `CreateMessage/CreateChatCom-  
pletion` z obiektem żądania i odbierz typowaną klasę odpowiedzi.

Informacje ogólne o dostawcy zawiera przewodnik [Ollama](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 2 zdarzenia do obsługi.

# Ollama | Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

## Prosty przykład

Wyślij wiadomość powitalną do lokalnego modelu Ollama.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);
Ollama->OllamaOptions->Host = "http://localhost:11434";
ShowMessage(Ollama->_CreateMessage("llama3", "Hello!"));
```

## Przykład monitu systemowego

Wysyłanie wiadomości z komunikatem systemowym w celu kontroli zachowania modelu.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);
Ollama->OllamaOptions->Host = "http://localhost:11434";
ShowMessage(Ollama->_CreateMessageWithSystem("llama3",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);
Ollama->OllamaOptions->Host = "http://localhost:11434";
Ollama->OnHTTPAPISSE = OnSSEEvent;
Ollama->_CreateMessageStream("llama3", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    // aEvent contains the event type
    // aData contains the JSON data for this event
    Memo1->Lines->Add(aData);
}
```

# Ollama | Modele

Zarządzaj lokalnie dostępnymi modelami Ollama i wykonuj na nich zapytania.

## Lista modeli

Wyświetla listę wszystkich dostępnych modeli.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";  
ShowMessage(Ollama->_GetModels());
```

## Get Tags

Wyświetla wszystkie lokalnie dostępne tagi modeli.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";  
ShowMessage(Ollama->_GetTags());
```

## Pokaż model

Pobiera szczegółowe informacje o konkretnym modelu.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";  
ShowMessage(Ollama->_ShowModel("llama3"));
```

## Model Pull

Pobiera model z biblioteki modeli Ollama.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";  
ShowMessage(Ollama->_PullModel("llama3"));
```

# Ollama | Embeddings

Generuje osadzenia wektorowe z danych tekstowych przy użyciu lokalnie działających modeli Ollama. Osadzenia mogą być używane do wyszukiwania semantycznego, klastrowania i innych zadań przetwarzania języka naturalnego.

## Tworzenie osadzeń

Generuje osadzenia dla podanego tekstu wejściowego.

```
TsgcHTTP_API_Ollama *Ollama = new TsgcHTTP_API_Ollama(NULL);  
Ollama->OllamaOptions->Host = "http://localhost:11434";  
ShowMessage(Ollama->_CreateEmbeddings("llama3", "Hello world"));
```

# Grok

Grok to konwersacyjny asystent AI opracowany przez xAI, zaprojektowany w celu dostarczania pomocnych i informacyjnych odpowiedzi z dostępem do wiedzy w czasie rzeczywistym. Modele Grok są znane z silnych możliwości wnioskowania i dostępu do aktualnych informacji.

Biblioteka `sgcWebSockets` udostępnia komponent Delphi **TsgcHTTP\_API\_Grok** do interakcji z Grok API.

## Grok API

Grok API zapewnia dostęp do modeli Grok służących do tworzenia aplikacji opartych na AI. API obsługuje generowanie tekstu, wizję (rozumienie obrazów), strumieniowanie oraz wylistowanie modeli.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Grok](#)
- **Vision**
  - [Przykłady Grok Vision](#)
- **Modele**
  - [Przykłady modeli Grok](#)

## Konfiguracja

Grok API używa kluczy API do uwierzytelniania. Odwiedź swoją [konsolę xAI](#), aby pobrać klucz API używany w żądaniach.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi być skonfigurowany we właściwości **GrokOptions.ApiKey** komponentu.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);
Grok->GrokOptions->ApiKey = "YOUR_API_KEY";
```

## Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

- **\_CreateMessage:** Tworzy wiadomość z określonym modelem i promptem użytkownika.
  - **Model:** Model do użycia (np. grok-3).
  - **Message:** Treść wiadomości użytkownika.
- **\_CreateMessageWithSystem:** Tworzy wiadomość z monitem systemowym.
  - **System:** Monit systemowy określający zachowanie asystenta.
- **\_CreateMessageStream:** Tworzy wiadomość z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia `OnHTTPAPISSE`.

## Vision

Grok rozumie obrazy przekazywane jako treść zakodowana w formacie base64 w wiadomościach.

- **\_CreateVisionMessage:** Wysyła obraz z monitem tekstowym.
  - **ImageBase64:** Dane obrazu zakodowane w base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.

## Modele

Wyświetl dostępne modele Grok.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.

# TsgcHTTP\_API\_Grok

Komponent TsgcHTTP\_API\_Grok.

## Wprowadzenie

Klient REST API xAI Grok do uzupełniania czatu i listowania modeli.

## Wprowadzenie

Klient REST API xAI Grok do uzupełniania czatu i listowania modeli.

Należy skonfigurować właściwość **GrokOptions** z kluczem API i opcjonalnymi ustawieniami HTTP/powtórzeń/logowania, a następnie wywoływać typowane metody takie jak `CreateMessage/CreateChatCompletion` z obiektem żądania i otrzymywać typowaną klasę odpowiedzi.

Informacje o dostawcy zawiera przewodnik [Grok](#).

## Dokumentacja

- Właściwości — 7 właściwości dostępnych w komponencie.
- Metody — 1 metoda publiczna.
- Zdarzenia — 0 zdarzeń do obsługi.

# Grok | Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

## Prosty przykład

Wyślij wiadomość Hello do Grok.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);
Grok->GrokOptions->ApiKey = "API_KEY";
ShowMessage(Grok->_CreateMessage("grok-3", "Hello!"));
```

## Przykład monitu systemowego

Wyślij wiadomość z systemowym monitem, aby kontrolować zachowanie Grok.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);
Grok->GrokOptions->ApiKey = "API_KEY";
ShowMessage(Grok->_CreateMessageWithSystem("grok-3",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);
Grok->GrokOptions->ApiKey = "API_KEY";
Grok->OnHTTPAPISSE = OnSSEEvent;
Grok->_CreateMessageStream("grok-3", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    Memo1->Lines->Add(aData);
}
```



# Grok | Wizja

Grok rozumie obrazy przekazywane jako zawartość zakodowana w formacie base64 w wiadomościach. Użyj punktu końcowego wizji, aby wysłać obraz wraz z odpowiedzią tekstową i otrzymać opis lub analizę.

## Przykład wizji

Wysyłanie obrazu zakodowanego w base64 do Grok w celu analizy.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);
Grok->GrokOptions->ApiKey = "API_KEY";
ShowMessage(Grok->_CreateVisionMessage("grok-2-vision",
    "Describe this image.", vBase64, "image/jpeg"));
```

# Grok | Modele

Wyświetlenie listy dostępnych modeli Grok wraz z podstawowymi informacjami.

## Przykład wyświetlania listy modeli

Pobierz listę wszystkich dostępnych modeli Grok.

```
TsgcHTTP_API_Grok *Grok = new TsgcHTTP_API_Grok(NULL);  
Grok->GrokOptions->ApiKey = "API_KEY";  
ShowMessage(Grok->_GetModels);
```

# Mistral AI

Mistral AI to francuska firma zajmująca się sztuczną inteligencją, która opracowuje wydajne i potężne modele językowe. Jej modele są znane z wysokiej wydajności w zakresie rozumowania, kodowania i zadań wielojęzycznych, przy jednoczesnym zachowaniu konkurencyjnej efektywności.

Biblioteka `sgcWebSockets` udostępnia komponent Delphi **TsgcHTTP\_API\_Mistral** do interakcji z Mistral AI API.

## Mistral API

Mistral API zapewnia dostęp do modeli Mistral służących do budowania aplikacji opartych na AI. API obsługuje generowanie tekstu, wizję (rozumienie obrazów), przesyłanie strumieniowe, osadzenia, tryb JSON i listowanie modeli.

## Funkcje

- **Wiadomości**
  - [Przykłady wiadomości Mistral](#)
- **Wizja**
  - [Przykłady Mistral Vision](#)
- **Modele**
  - [Przykłady modeli Mistral](#)
- **Osadzenia**
  - [Przykłady osadzeń Mistral](#)

## Konfiguracja

API Mistral używa kluczy API do uwierzytelniania. Należy odwiedzić stronę [Klucze API](#) w konsoli Mistral, aby pobrać klucz API używany w żądaniach.

Należy pamiętać, że klucz API jest tajny. Nie należy udostępniać go innym osobom ani umieszczać w kodzie po stronie klienta.

Ten **klucz API** musi być skonfigurowany we właściwości **MistralOptions.ApiKey** komponentu.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);  
Mistral->MistralOptions->ApiKey = "YOUR_API_KEY";
```

## Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

- **\_CreateMessage:** Tworzy wiadomość z określonym modelem i promptem użytkownika.
  - **Model:** Model do użycia (np. `mistral-large-latest`).
  - **Message:** Treść wiadomości użytkownika.
- **\_CreateMessageWithSystem:** Tworzy wiadomość z monitem systemowym.
  - **System:** Monit systemowy określający zachowanie asystenta.
- **\_CreateMessageStream:** Tworzy wiadomość z włączonym przesyłaniem strumieniowym (SSE). Zdarzenia są dostarczane przez procedurę obsługi zdarzenia `OnHTTPAPISSE`.

- **\_CreateMessageJSON:** Tworzy wiadomość z włączonym trybem JSON. Model zwróci prawidłową odpowiedź JSON.
  - **Model:** Model do użycia.
  - **Message:** Treść wiadomości użytkownika.

## Vision

Mistral potrafi rozumieć obrazy przekazywane jako zawartość zakodowana w Base64 w wiadomościach.

- **\_CreateVisionMessage:** Wysyła obraz z monitem tekstowym.
  - **ImageBase64:** Dane obrazu zakodowane w base64.
  - **MediaType:** Typ MIME (image/jpeg, image/png, image/gif, image/webp).
  - **Prompt:** Monit tekstowy towarzyszący obrazowi.

## Modele

Wyświetl listę dostępnych modeli Mistral.

- **\_GetModels:** Wyświetla listę wszystkich dostępnych modeli.

## Osadzenia

Uzyskaj wektorową reprezentację podanego wejścia, która może być używana do wyszukiwania semantycznego, grupowania i innych zadań uczenia maszynowego.

- **\_CreateEmbeddings:** Tworzy wektor osadzenia reprezentujący tekst wejściowy.
  - **Model:** Model do użycia (np. mistral-embed).
  - **Input:** Tekst wejściowy, dla którego mają zostać wygenerowane osadzenia.

# TsgcHTTP\_API\_Mistral

Klient Mistral REST API dla uzupełnień czatu i osadzeń.

## Wprowadzenie

Klient Mistral REST API dla uzupełnień czatu i osadzeń.

## Wprowadzenie

Klient Mistral REST API dla uzupełnień czatu i osadzeń.

Należy skonfigurować właściwość **MistralOptions**, podając klucz API oraz opcjonalne ustawienia HTTP/ponownych prób/dziennika, a następnie wywoływać typowane metody, takie jak `CreateMessage/``CreateChatCompletion`, z obiektem żądania i odbierać typowaną klasę odpowiedzi.

Informacje ogólne na temat dostawcy można znaleźć w przewodniku [Mistral](#).

## Dokumentacja

- Właściwości — 5 właściwości dostępnych w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 2 zdarzenia do obsługi.

# Mistral | Wiadomości

Wyślij ustrukturyzowaną listę wiadomości wejściowych z treścią tekstową, a model wygeneruje kolejną wiadomość w rozmowie.

## Prosty przykład

Wyślij wiadomość Hello do Mistral.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);
Mistral->MistralOptions->ApiKey = "API_KEY";
ShowMessage(Mistral->_CreateMessage("mistral-large-latest", "Hello!"));
```

## Przykład monitu systemowego

Wysła wiadomość z systemowym poleceniem sterującym zachowaniem Mistral.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);
Mistral->MistralOptions->ApiKey = "API_KEY";
ShowMessage(Mistral->_CreateMessageWithSystem("mistral-large-latest",
    "You are a helpful assistant that responds in Spanish.",
    "What is the capital of France?"));
```

## Przykład strumieniowania

Należy użyć Server-Sent Events (SSE) do strumieniowania odpowiedzi w czasie rzeczywistym. W celu odbierania zdarzeń strumieniowych należy przypisać procedurę obsługi zdarzenia OnHTTPAPISSE.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);
Mistral->MistralOptions->ApiKey = "API_KEY";
Mistral->OnHTTPAPISSE = OnSSEEvent;
Mistral->_CreateMessageStream("mistral-large-latest", "Tell me a story.");

void __fastcall TForm1::OnSSEEvent(TObject *Sender, const String aEvent,
    const String aData, bool &Cancel)
{
    Memo1->Lines->Add(aData);
}
```

## Przykład trybu JSON

Żąda odpowiedzi w formacie JSON od modelu Mistral.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);
Mistral->MistralOptions->ApiKey = "API_KEY";
ShowMessage(Mistral->_CreateMessageJSON("mistral-large-latest",
    "List 3 colors as JSON"));
```

# Mistral | Vision

Mistral potrafi rozumieć obrazy przekazywane jako treść zakodowana w base64 wewnątrz wiadomości. Należy użyć punktu końcowego vision, aby wysłać obraz wraz z promptem tekstowym i otrzymać opis lub analizę.

## Przykład wizji

Wyślij obraz zakodowany w base64 do Mistral w celu analizy.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);
Mistral->MistralOptions->ApiKey = "API_KEY";
ShowMessage(Mistral->_CreateVisionMessage("mistral-large-latest",
    "Describe this image.", vBase64, "image/jpeg"));
```

# Mistral | Modele

Wyświetla dostępne modele Mistral i ich podstawowe informacje.

## Przykład wyświetlania listy modeli

Pobiera listę wszystkich dostępnych modeli Mistral.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);  
Mistral->MistralOptions->ApiKey = "API_KEY";  
ShowMessage(Mistral->_GetModels);
```



# Mistral | Embeddings

Uzyskaj wektorową reprezentację podanego wejścia, która może być używana do wyszukiwania semantycznego, grupowania i innych zadań uczenia maszynowego.

## Przykład Embeddings

Tworzy wektor osadzania dla tekstu wejściowego.

```
TsgcHTTP_API_Mistral *Mistral = new TsgcHTTP_API_Mistral(NULL);  
Mistral->MistralOptions->ApiKey = "API_KEY";  
ShowMessage(Mistral->_CreateEmbeddings("mistral-embed", "Hello world"));
```

# IoT

Internet rzeczy (IoT) odnosi się do koncepcji rozszerzenia łączności internetowej poza konwencjonalne platformy obliczeniowe, takie jak komputery osobiste i urządzenia mobilne, na różnorodne tradycyjnie „głupie” lub niepodłączone do internetu urządzenia fizyczne oraz przedmioty codziennego użytku. Wyposażone w elektronikę, łączność internetową i inne formy sprzętu (np. czujniki), urządzenia te mogą komunikować się i wchodzić w interakcje z innymi przez internet, a także być zdalnie monitorowane i sterowane.

Pakiet `sgcWebSockets` implementuje następujące klienty IoT:

**1. Amazon AWS IoT:** AWS IoT zapewnia bezpieczną, dwukierunkową komunikację między urządzeniami podłączonymi do Internetu, takimi jak czujniki, urządzenia wykonawcze, wbudowane mikrokontrolery lub inteligentne urządzenia domowe, a chmurą AWS. Umożliwia to zbieranie danych telemetrycznych z wielu urządzeń, ich przechowywanie i analizę. Pozwala również tworzyć aplikacje, dzięki którym użytkownicy mogą sterować tymi urządzeniami ze swoich telefonów lub tabletów.

**2. Azure IoT Hub:** IoT Hub to zarządzana usługa hostowana w chmurze, pełniąca rolę centralnego centrum wiadomości dla dwukierunkowej komunikacji między aplikacją IoT a zarządzanymi przez nią urządzeniami. Usługa Azure IoT Hub umożliwia budowanie rozwiązań IoT z niezawodną i bezpieczną komunikacją między milionami urządzeń IoT a zapleczem rozwiązania hostowanym w chmurze.

# Klient IoT Amazon MQTT

## Czym jest AWS IoT?

AWS IoT zapewnia bezpieczną, dwukierunkową komunikację między urządzeniami podłączonymi do internetu (takimi jak czujniki, siłowniki, mikrokontrolery lub inteligentne urządzenia) a chmurą AWS. Umożliwia to zbieranie danych telemetrycznych z wielu urządzeń, ich przechowywanie i analizę. Można również tworzyć aplikacje pozwalające użytkownikom sterować tymi urządzeniami z telefonów lub tabletów.

## Broker wiadomości

Zapewnia bezpieczny mechanizm umożliwiający urządzeniom i aplikacjom AWS IoT wzajemne publikowanie i odbieranie wiadomości. Można korzystać zarówno z protokołu MQTT bezpośrednio, jak i MQTT przez WebSocket do publikowania i subskrybowania.

Broker wiadomości AWS IoT to usługa brokera publikuj/subskrybuj umożliwiająca wysyłanie i odbieranie wiadomości do i z AWS IoT. Podczas komunikacji z AWS IoT klient wysyła wiadomość zaadresowaną do tematu, takiego jak Sensor/temp/room1.

Broker wiadomości z kolei wysyła wiadomość do wszystkich klientów, którzy zarejestrowali się do odbierania wiadomości dla danego tematu. Czynność wysyłania wiadomości jest nazywana publikowaniem. Czynność rejestrowania się do odbierania wiadomości dla filtru tematu jest nazywana subskrybowaniem.

Przestrzeń nazw tematów jest izolowana dla każdej pary konta AWS i regionu. Na przykład temat Sensor/temp/room1 dla jednego konta AWS jest niezależny od tematu Sensor/temp/room1 dla innego konta AWS. Dotyczy to również regionów. Temat Sensor/temp/room1 na tym samym koncie AWS w us-east-1 jest niezależny od tego samego tematu w us-east-2. AWS IoT nie obsługuje wysyłania i odbierania wiadomości między kontami AWS i regionami.

Broker wiadomości przechowuje listę wszystkich sesji klientów wraz z subskrypcjami dla każdej sesji. Gdy wiadomość jest publikowana na temat, broker sprawdza sesje z subskrypcjami pasującymi do tego tematu. Broker przekazuje następnie opublikowaną wiadomość do wszystkich sesji z aktywnie połączonym klientem.

## Klient MQTT

**TsgcIoTAmazon\_MQTT\_Client** to komponent służący do łączenia się z AWS IoT. Jeden klient może połączyć się tylko z jednym urządzeniem. Klient łączy się przy użyciu protokołu MQTT i uwierzytelnia się za pomocą certyfikatu klienta X.509.

Aby połączyć się z AWS IoT, klient musi mieć skonfigurowane następujące właściwości:

**Amazon.ClientId:** identyfikacja klienta, opcjonalna.

**Amazon.Endpoint:** nazwa serwera, z którym klient MQTT będzie się łączyć.

**Amazon.Port:** domyślnie używany jest port 8883. Jeśli port wynosi 443, do połączenia automatycznie stosowany jest ALPN (wymaga niestandardowej wersji Indy).

AWS IoT Core obsługuje urządzenia i klientów korzystających z protokołów MQTT i MQTT over WebSocket Secure (WSS) do publikowania i subskrybowania wiadomości. Poniższa tabela zawiera protokoły obsługiwane przez punkty końcowe urządzeń AWS IoT wraz z metodami uwierzytelniania i portami.

|                      |                                 |      |                |
|----------------------|---------------------------------|------|----------------|
| MQTT przez WebSocket | Signature Version 4             | 443  |                |
| MQTT przez WebSocket | Uwierzytelnianie niestandardowe | 443  |                |
| MQTT                 | Certyfikat klienta X.509        | 443  | x-amzn-mqtt-ca |
| MQTT                 | Certyfikat klienta X.509        | 8883 |                |
| MQTT                 | Uwierzytelnianie niestandardowe | 443  | mqtt           |

## Uwierzytelnianie certyfikatami

Należy utworzyć certyfikaty w konsoli Amazon AWS i ustawić ścieżkę, w której są przechowywane.

Korzystając z **OpenSSL** jako IOHandler, należy ustawić certyfikat w następujących ścieżkach

**Certificate.Enabled:** ustaw na True, jeśli chcesz używać certyfikatów.

**Certificate.CertFile:** ścieżka do certyfikatu klienta X.509.

**Certificate.KeyFile:** ścieżka do pliku klucza klienta X.509.

Używając **SChannel** jako IOHandler, należy najpierw przekonwertować certyfikat PEM wraz z kluczem do formatu PFX. Wymaga to plików binarnych OpenSSL:

```
openssl pkcs12 -inkey 884ccf73ff-private.pem.key -in 884ccf73ff-certificate.pem.crt -export -out 884ccf73ff-certificate.pfx
```

Następnie należy ustawić poniższe ścieżki (nie ma potrzeby ustawiania pliku klucza, ponieważ jest on już zawarty w certyfikacie).

**Certificate.Enabled:** ustaw na True, jeśli chcesz używać certyfikatów.

**Certificate.CertFile:** ścieżka do certyfikatu PFX

## Uwierzytelnianie SignatureV4

Należy utworzyć użytkownika w konsoli Amazon AWS i zapisać klucze dostępu oraz klucze sekretne, które będą używane do podpisywania żądania WebSocket.

**SignatureV4.Enabled:** należy ustawić wartość True, aby korzystać z tego typu uwierzytelniania.

**SignatureV4.Region:** region, w którym znajduje się urządzenie (przykład: us-east-1).

**SignatureV4.AccessKey:** klucz dostępu utworzony w konsoli Amazon lub uzyskany jako tymczasowe poświadczenie.

**SignatureV4.SecretKey:** klucz tajny utworzony w konsoli Amazon lub uzyskany jako tymczasowe poświadczenie

**SignatureV4.SessionToken:** (warunkowe) w przypadku korzystania z tymczasowych poświadczeń bezpieczeństwa należy tutaj podać token bezpieczeństwa.

**OpenSSL\_Options:** konfiguracja bibliotek OpenSSL.

**APIVersion:** pozwala określić, które API OpenSSL będzie używane.

**oslAPI\_1\_0:** używa API 1.0 OpenSSL; jest to najnowsza wersja obsługiwana przez Indy

**oslAPI\_1\_1:** używa interfejsu API 1.1 OpenSSL, wymaga niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**oslAPI\_3\_0:** korzysta z API OpenSSL 3.0, wymaga naszej niestandardowej biblioteki Indy i umożliwia używanie bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek openssl

**oslpNone:** jest to wartość domyślna; biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanej ścieżce.

**oslpDefaultFolder:** automatycznie ustawia ścieżkę openSSL, w której powinny znajdować się biblioteki dla wszystkich osobowości IDE.

**oslpCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.

**LibPathCustom:** gdy LibPath = oslpCustomFolder, należy podać pełną ścieżkę do folderu, w którym znajdują się biblioteki openSSL.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**oslsSymLinksDefault:** domyślnie jest włączone, z wyjątkiem platformy OSX64 (po MacOS Monterey próba załadowania biblioteki bez wersji kończy się błędem).

**oslsSymLinksLoadFirst:** Załaduj dowiązania symboliczne przed próbą załadowania bibliotek wersji.

**oslsSymLinksLoad:** Ładuj dowiązania symboliczne po próbie załadowania bibliotek wersji.

**oslsSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

*\*SignatureV4 wymaga Indy w wersji 10.5.7 lub nowszej*

## Uwierzytelnianie niestandardowe

Uwierzytelnianie niestandardowe umożliwia zdefiniowanie sposobu uwierzytelniania i autoryzacji klientów przy użyciu zasobów autoryzatora. Urządzenie przekazuje poświadczenia w polach nagłówka żądania lub parametrach zapytania (dla protokołów MQTT przez WebSockets) lub w polach nazwy użytkownika i hasła komunikatu MQTT CONNECT (dla protokołów MQTT i MQTT przez WebSockets).

**CustomAuthentication.Enabled:** należy ustawić na True, aby korzystać z tego rodzaju uwierzytelniania.

**CustomAuthentication.Parameters:** należy tu ustawić parametry zapytania, które będą przekazywane do serwera (domyślnie /mqtt)

**CustomAuthentication.Headers:** tu można umieścić niestandardowe pola nagłówka.

**CustomAuthentication.WebSockets:** jeśli ustawiono na true, połączenie będzie działać przez protokół WebSocket; w przeciwnym razie będzie działać przez zwykłe TCP.

**MQTTAuthentication.Enabled:** jeśli konieczne jest przekazywanie nazwy użytkownika i hasła w połączeniu MQTT, należy włączyć tę właściwość.

**MQTTAuthentication.Username:** nazwa użytkownika połączenia MQTT

**MQTTAuthentication.Password:** hasło połączenia MQTT.

Klient może opcjonalnie wysłać ClientId identyfikujący połączenie klienta; inne klienty mogą wówczas subskrybować powiadomienia o każdym zdarzeniu tego klienta (połączenie, subskrypcja, rozłączenie itd.).

## Autoryzacja

Jeśli nie można połączyć się przy użyciu portu 8883 i protokołu TCP jako transportu (co jest ustawieniem domyślnym), Amazon stosuje "zasady AWS IoT Core" w celu przyznania lub odmowy autoryzacji klientom i subskrypcjom. Najprawdopodobniej konieczne jest autoryzowanie identyfikatora klienta.

Należy zalogować się do konsoli Amazon AWS, przejść do IoT Core i otworzyć menu „Secure/Policies”, następnie wybrać zasadę przypisaną do urządzenia IoT (IoT Thing) i sprawdzić na końcu, jak skonfigurowane jest połączenie. Przykład:

```
{  
  "Effect": "Allow",  
  "Action": [  
    "iot:Connect"  
  ],  
  "Resource": [  
    "arn:aws:iot:us-east-1:222178873557:client/sdk-java",  
    "arn:aws:iot:us-east-1:222178873557:client/basicPubSub",  
    "arn:aws:iot:us-east-1:222178873557:client/sdk-nodejs-*"  
  ]  
}
```

Ta konfiguracja oznacza, że połączyć się będą mogli wyłącznie klienci o identyfikatorach: sdk-java, basicPubSub i sdk-nodejs-\*. Należy odpowiednio zmienić ustawienia i spróbować ponownie.

Jeśli problem nadal występuje, włącz dziennik i sprawdź w CloudWatch przyczynę braku możliwości połączenia.

## Inne właściwości

**MQTTHeartBeat:** jeśli włączone, próbuje utrzymać połączenie MQTT wysyłając ping co x sekund.

**Interval:** liczba sekund pomiędzy kolejnymi pingami.

**MQTTAuthentication:** jeśli włączone, uwzględnia nazwę użytkownika i hasło w połączeniu MQTT

**UserName:** nazwa użytkownika

**Hasło:** tajny ciąg znaków

**WatchDog:** jeżeli jest włączony, w przypadku wykrycia nieoczekiwanego rozłączenia podejmuje automatyczną próbę ponownego połączenia z serwerem.

**Interval:** liczba sekund przed kolejnymi próbami ponownego połączenia.

**Attempts:** maksymalna liczba prób ponownego połączenia; zero oznacza bez ograniczeń.

**LogFile:** jeśli włączone, zapisuje wiadomości gniazda do pliku dziennika (przydatne do debugowania). Dostęp do pliku dziennika nie jest bezpieczny wątkowo, gdy jest dostępny z wielu wątków.

**Enabled:** jeśli włączone, każda wiadomość odebrana i wysłana przez gniazdo będzie zapisywana do pliku.

**FileName:** pełna ścieżka do pliku.

## Implementacja

Implementacja Amazon MQTT opiera się na MQTT w wersji 3.1.1, jednak odbiega od specyfikacji w następujący sposób:

- W AWS IoT subskrybowanie tematu z poziomem QoS 0 oznacza, że wiadomość jest dostarczana zero lub więcej razy. Wiadomość może zostać dostarczona więcej niż raz. Wielokrotnie dostarczone wiadomości mogą być wysyłane z innym identyfikatorem pakietu. W takich przypadkach flaga DUP nie jest ustawiana.



- AWS IoT nie obsługuje publikowania i subskrybowania z QoS 2. Broker komunikatów AWS IoT nie wysyła PUBACK ani SUBACK w przypadku żądania QoS 2.
- Odpowiadając na żądanie połączenia, broker wiadomości wysyła wiadomość CONNACK. Wiadomość ta zawiera flagę informującą, czy połączenie wznowia poprzednią sesję. Wartość tej flagi może być nieprawidłowa, jeśli dwa klienty MQTT łączą się jednocześnie z tym samym identyfikatorem klienta.
- Gdy klient subskrybuje temat, może wystąpić opóźnienie między momentem wysłania przez broker komunikatu SUBACK a momentem, w którym klient zaczyna odbierać nowe pasujące wiadomości.
- Specyfikacja MQTT zawiera przepis umożliwiający wydawcy żądanie, aby broker zachował ostatnią wiadomość wysłaną do tematu i dostarczył ją wszystkim przyszłym subskrybentom. AWS IoT nie obsługuje zachowywania wiadomości. Jeśli zostanie wysłane żądanie zachowania wiadomości, połączenie zostanie rozłączone.
- Broker wiadomości używa identyfikatora klienta do identyfikacji każdego klienta. Identyfikator klienta jest przekazywany przez klienta do brokera wiadomości jako część ładunku MQTT. Dwóch klientów o tym samym identyfikatorze klienta nie może być jednocześnie połączonych z brokerem. Gdy klient łączy się z brokerem używając identyfikatora klienta, który jest już zajęty przez innego klienta, do obu klientów wysyłana jest wiadomość CONNACK, a aktualnie połączony klient jest rozłączany.
- W rzadkich przypadkach broker wiadomości może ponownie wysłać tę samą logiczną wiadomość PUBLISH z innym identyfikatorem pakietu.
- Broker wiadomości nie gwarantuje kolejności, w jakiej odbierane są wiadomości i potwierdzenia ACK.

## Połącz z AWS IoT

Najpierw należy zalogować się do konsoli AWS, zarejestrować nowe urządzenie i utworzyć certyfikat X.509 dla tego urządzenia. Po wykonaniu tych czynności można utworzyć nowy obiekt `TsgcIoTAmazon_MQTT_Client` i połączyć się z serwerem AWS IoT. Na przykład:

```
oClient = new TsgcIoTAmazon_MQTT_Client();
oClient->Amazon->Endpoint = "a2ohgdjqitsmij-ats.iot.us-west-2.amazonaws.com";
oClient->Amazon->ClientId = "sgcWebSockets";
oClient->Certificate->CertFile = "amazon-certificate.pem.crt";
oClient->Certificate->KeyFile = "amazon-private.pem.key";
oClient->OnMQTTConnect = OnMQTTConnectEvent;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const TmqttConnReturnCode
ReturnCode)
{
    ShowMessage("Connected to AWS");
}
```

## Tematy

Broker wiadomości używa tematów do kierowania wiadomości od klientów publikujących do klientów subskrybujących. Ukośnik (/) służy do oddzielania hierarchii tematów. Poniższa tabela zawiera listę symboli wieloznacznych, które można stosować w filtrze tematu podczas subskrypcji. Znak # musi być ostatnim znakiem tematu, do którego jest subskrypcja. Działa jako symbol wieloznaczny pasując do bieżącego drzewa i wszystkich poddrzew.

Na przykład subskrypcja tematu Sensor/# odbiera wiadomości publikowane w Sensor/, Sensor/temp, Sensor/temp/room1, ale nie wiadomości publikowane w Sensor.

+ Pasuje dokładnie do jednego elementu w hierarchii tematów. Na przykład subskrypcja Sensor/+/room1 odbiera wiadomości opublikowane w Sensor/temp/room1, Sensor/moisture/room1 i tak dalej.

```
oClient = new TsgcIoTAmazon_MQTT_Client();
...
oClient->OnSubscribe = OnSubscribeEvent;

vPacketIdentifier = oClient->Subscribe("Sensor/moisture/room1");

void OnMQTTSubscribe(TsgcWSConnection *Connection, Word aPacketIdentifier, TsgcWSSUBACKS *aCodes)
{
    if (vPacketIdentifier == aPacketIdentifier)
    {
        ShowMessage("Subscribed to topic Sensor/moisture/room1");
    }
}

// Client, can send a message using Publish method.
oClient->Publish("Sensor/moisture/room1", "{\"temp\"=10}");

// Messages received from server, are dispatched OnMQTTPublishEvent.
// For extended payload access (string, bytes or stream), use OnMQTTPublishEx.
void OnMQTTPublish(TsgcWSConnection *Connection, string aTopic, string aText)
{
    DoLog("Received Message: " + aTopic + " " + aText);
}
```

## Zarezerwowane tematy

Następujące metody służą do subskrybowania i publikowania w zarezerwowanych tematach.

**Subscribe\_ClientConnected(const aClientId: String):** AWS IoT publikuje do tego tematu, gdy klient MQTT o określonym identyfikatorze klienta łączy się z AWS IoT

**Subscribe\_ClientDisconnected(const aClientId: String):** AWS IoT publikuje do tego tematu, gdy klient MQTT o podanym identyfikatorze klienta rozłącza się z AWS IoT

**Subscribe\_ClientSubscribed(const aClientId: String):** AWS IoT publikuje do tego tematu, gdy klient MQTT o podanym identyfikatorze klienta subskrybuje temat MQTT

**Subscribe\_ClientUnSubscribed(const aClientId: String):** AWS IoT publikuje w tym temacie, gdy klient MQTT z określonym identyfikatorem klienta anuluje subskrypcję tematu MQTT

**Publish\_Rule(const aRuleName, aText: String):** Urządzenie lub aplikacja publikuje do tego tematu, aby bezpośrednio wyzwoić reguły

**Publish\_DeleteShadow(const aThingName, aText: String):** Urządzenie lub aplikacja publikuje w tym temacie, aby usunąć cień

**Subscribe\_DeleteShadow(const aThingName: String):** Urządzenie lub aplikacja subskrybuje ten temat w celu usunięcia cienia

**Subscribe\_ShadowDeleted(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu po usunięciu shadow.

**Subscribe\_ShadowRejected(const aThingName: String):** usługa Device Shadow wysyła wiadomości do tego tematu, gdy żądanie usunięcia cienia zostanie odrzucone.

**Publish\_ShadowGet(const aThingName, aText: String):** Aplikacja lub urządzenie publikuje pustą wiadomość do tego tematu, aby pobrać shadow

**Subscribe\_ShadowGet(const aThingName: String):** aplikacja lub urządzenie subskrybuje ten temat, aby pobrać cień

**Subscribe\_ShadowGetAccepted(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu po pomyślnym zrealizowaniu żądania dotyczącego cienia

**Subscribe\_ShadowGetRejected(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu, gdy żądanie dotyczące cienia zostaje odrzucone

**Publish\_ShadowUpdate(const aThingName, aText: String):** Urządzenie lub aplikacja publikuje do tego tematu w celu aktualizacji cienia

**Subscribe\_ShadowUpdateAccepted(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu po pomyślnym zastosowaniu aktualizacji cienia

**Subscribe\_ShadowUpdateRejected(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu, gdy aktualizacja shadow zostaje odrzucona

**Subscribe\_ShadowUpdateDelta(const aThingName: String):** Usługa Device Shadow wysyła wiadomości do tego tematu, gdy wykryta zostanie różnica między sekcjami reported i desired cienia

**Subscribe\_ShadowUpdateDocuments(const aThingName: String):** AWS IoT publikuje dokument stanu w tym temacie za każdym razem, gdy aktualizacja cienia zostanie pomyślnie wykonana

## Sesje trwałe

Sesja trwała reprezentuje bieżące połączenie z brokerem wiadomości MQTT. Gdy klient łączy się z brokerem wiadomości AWS IoT przy użyciu sesji trwałej, broker wiadomości zapisuje wszystkie subskrypcje dokonane przez klienta podczas połączenia. Po rozłączeniu klienta broker przechowuje niepotwierdzone wiadomości QoS 1 oraz nowe wiadomości QoS 1 publikowane w tematach, które subskrybuje klient. Po ponownym połączeniu klienta z sesją trwałą wszystkie subskrypcje zostają przywrócone, a wszystkie przechowywane wiadomości są wysyłane do klienta z maksymalną prędkością 10 wiadomości na sekundę.

Trwałą sesję MQTT tworzy się, ustawiając parametr **cleanSession** na **False** w zdarzeniu **OnMQTTBeforeConnect**. Jeśli dla klienta nie istnieje żadna sesja, tworzona jest nowa sesja trwała. Jeśli sesja już istnieje, zostaje wznowiona.

Urządzenia muszą sprawdzić atrybut **Session** w zdarzeniu **OnMQTTConnect**, aby ustalić, czy istnieje sesja trwała. Jeśli **Session ma wartość True**, sesja trwała jest obecna i przechowywane wiadomości są dostarczane do klienta. Jeśli **Session ma wartość False**, sesja trwała nie istnieje i klient musi ponownie subskrybować filtry tematyczne.

Trwałe sesje mają domyślny czas wygaśnięcia wynoszący 1 godzinę. Czas wygaśnięcia rozpoczyna się w momencie wykrycia przez brokera wiadomości rozłączenia klienta (rozłączenie MQTT lub przekroczenie czasu). Okres wygaśnięcia trwałej sesji można wydłużyć za pomocą standardowego procesu zwiększania limitów. Jeśli klient nie wznowi sesji w ciągu okresu wygaśnięcia, sesja zostaje zakończona, a wszelkie powiązane przechowywane wiadomości są odrzucane. Okres wygaśnięcia jest przybliżony — sesje mogą być przechowywane nawet o 30 minut dłużej (ale nie krócej) niż skonfigurowany czas trwania.

## Tymczasowe dane uwierzytelniające

AWS IoT Core może pracować z tymczasowymi poświadczeniami uzyskanymi za pośrednictwem pul tożsamości; istnieją 2 typy tożsamości:

- **UnAuthenticated:** wymaga jedynie ustawienia typu zasad w IAM
- **Authenticated:** wymaga ustawienia typu polityki w IAM oraz w politykach AWS IoT Core

## Nieuwierzytelniony

Jeśli używane są nieuwierzytelnione poświadczenia, należy dołączyć politykę do roli UnAuthenticated automatycznie tworzonej w menu IAM. Następnie należy skonfigurować klienta, ustawiając Access, Secret Key i Token zwrócone przez usługę Cognito.

Poniżej znajduje się kod w .NET umożliwiający uzyskanie nieuwierzytelnionych poświadczeń.

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials(
    "us-east-1:cc3c9c48-646d-44ef-bfd5-0c5fb2f0882f", // Identity pool ID
    Amazon.RegionEndpoint.USEast1 // Region
);

var identityPoolId = credentials.GetCredentialsAsync();

AmazonCognitoIdentityClient cognitoClient = new AmazonCognitoIdentityClient(
    credentials, // the anonymous credentials
    Amazon.RegionEndpoint.USEast1 // the Amazon Cognito region
);
```

```
GetIdRequest idRequest = new GetIdRequest();
idRequest.AccountId = "222178873557";
idRequest.IdentityPoolId = "us-east-1:cc3c9c48-646d-44ef-bfd5-0c5fb2f0882f";

GetIdResponse idResp = cognitoClient.GetId(idRequest);

string AccessKey = identityPoolId.Result.AccessKey;
string SecretKey = identityPoolId.Result.SecretKey;
string SessionToken = identityPoolId.Result.Token;

string IdentityId = idResp.IdentityId;
```

## Uwierzytelniono

Uwierzytelnione dane uwierzytelniające wymagają dołączenia polityki w roli uwierzytelnionej automatycznie tworzonej w menu IAM oraz dołączenia polityki użytkownika w zasadach AWS IoT Core.

Należy więc utworzyć nową politykę w menu polityk IoT Core i przy każdym uwierzytelnieniu nowego użytkownika dołączyć tę politykę do danego użytkownika.

Można użyć następującego polecenia AWS, aby dołączyć zasady lub utworzyć funkcję lambda.

```
aws iot attach-policy --policy-name PolicyName --target us-east-1:XXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

## Inicjowanie obsługi urządzeń

Usługa Fleet Provisioning obsługuje następujące operacje MQTT API:

- **CreateCertificateFromCsr:** tworzy certyfikat na podstawie żądania podpisania certyfikatu (CSR). AWS IoT dostarcza certyfikaty klienta podpisane przez główny urząd certyfikacji Amazon (CA). Nowy certyfikat ma status PENDING\_ACTIVATION. Po wywołaniu RegisterThing w celu udostępnienia urządzenia z tym certyfikatem, status certyfikatu zmienia się na ACTIVE lub INACTIVE zgodnie z opisem w szablonie.
- **CreateKeysAndCertificate:** Tworzy nowe klucze i certyfikat. AWS IoT udostępnia certyfikaty klienta podpisane przez główny urząd certyfikacji Amazon (CA). Nowy certyfikat ma status PENDING\_ACTIVATION. Po wywołaniu RegisterThing w celu aprowizacji urządzenia z tym certyfikatem, status certyfikatu zmienia się na ACTIVE lub INACTIVE zgodnie z opisem w szablonie.
- **RegisterThing:** Rejestruje urządzenie przy użyciu wstępnie zdefiniowanego szablonu.

### CreateCertificateFromCsr

Należy użyć metody `CreateCertificateFromCsr`, przekazując `CertificateSigningRequest` jako parametr, aby utworzyć certyfikat. Aby otrzymać odpowiedź na to żądanie, należy najpierw zasubskrybować następujące metody: `SubscribeCreateCertificateFromCsrResponse` i `SubscribeCreateCertificateFromCsrError`.

### CreateKeysAndCertificate

Należy użyć metody `CreateKeysAndCertificate`, aby utworzyć nowy certyfikat i klucze. Aby odebrać odpowiedź na to żądanie, należy najpierw zasubskrybować następujące metody: `SubscribeCreateKeysAndCertificateResponse` i `SubscribeCreateKeysAndCertificateError`

### RegisterThing

Metodą `RegisterThing` należy zarejestrować nową rzecz, przekazując jako parametry nazwę szablonu oraz ładunek w formacie JSON. Aby otrzymać odpowiedź na to żądanie, należy najpierw zasubskrybować następujące metody: `SubscribeRegisterThingResponse` oraz `SubscribeRegisterThingError`.

# IoT Azure MQTT Client

## Co to jest Azure IoT Hub?

IoT Hub to zarządzana usługa hostowana w chmurze, pełniąca rolę centralnego koncentratora wiadomości do dwukierunkowej komunikacji między aplikacją IoT a zarządzanymi przez nią urządzeniami. Azure IoT Hub umożliwia budowanie rozwiązań IoT z niezawodną i bezpieczną komunikacją między milionami urządzeń IoT a zapleczem rozwiązania hostowanym w chmurze. Z IoT Hub można połączyć praktycznie dowolne urządzenie.

IoT Hub obsługuje komunikację zarówno z urządzenia do chmury, jak i z chmury do urządzenia. IoT Hub obsługuje wiele wzorców komunikatów, takich jak telemetria urządzenie–chmura, przesyłanie plików z urządzeń oraz metody żądanie–odpowiedź służące do sterowania urządzeniami z chmury. Monitorowanie IoT Hub pomaga utrzymać kondycję rozwiązania poprzez śledzenie zdarzeń, takich jak tworzenie urządzeń, awarie urządzeń i połączenia urządzeń.

Możliwości IoT Hub pomagają budować skalowalne, w pełni funkcjonalne rozwiązania IoT, takie jak zarządzanie sprzętem przemysłowym w produkcji, śledzenie cennych zasobów w służbie zdrowia czy monitorowanie użytkowania budynków biurowych.

## Broker wiadomości

IoT Hub zapewnia bezpieczny kanał komunikacyjny dla urządzeń do wysyłania danych. IoT Hub i zestawy SDK urządzeń obsługują następujące protokoły do łączenia urządzeń:

- **MQTT**
- **MQTT** przez **WebSockets**

Wiele typów uwierzytelniania obsługuje różnorodne możliwości urządzeń:

- **SAS** uwierzytelnianie oparte na tokenach w celu szybkiego uruchomienia rozwiązania IoT.
- Uwierzytelnianie przy użyciu indywidualnych certyfikatów **X.509** — bezpieczna, oparta na standardach metoda uwierzytelniania.

## Klient MQTT

**TsgcIoTAzure\_MQTT\_Client** to komponent służący do połączenia z Azure IoT. Jeden klient może połączyć się tylko z jednym urządzeniem. Klient łączy się przy użyciu protokołu MQTT i uwierzytelnia się za pomocą SAS / certyfikatu klienta X.509.

Aby połączyć się z Azure IoT Hub, klient musi mieć skonfigurowane następujące właściwości:

**Azure.IoTHub:** nazwa serwera, z którym połączy się klient MQTT.

**Azure.DeviceId:** nazwa urządzenia w Azure IoT Hub.

Azure obsługuje wiele typów uwierzytelniania; domyślnie stosowane są tokeny SAS.

### Uwierzytelnianie SAS

**SAS.Enabled:** włącz jeśli uwierzytelnianie używa SAS.

**SAS.SecretKey:** token SAS z konta Azure IoT.

**SAS.KeyName:** nazwa klucza dostępu współdzielonego.

**SAS.Expiry:** ustawia liczbę minut przed wygaśnięciem tokenu SAS. Wartość domyślna to 1440 (24 godziny).

Jeśli posiadasz parametry połączenia, możesz odczytać ich wartości automatycznie za pomocą metody **ReadConnectionString**. Przykład:

```
ReadConnectionString('HostName=yourhub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=Yj7RRPnkSDTv+UCFLgwIP/FrbDymZv4qVAIoTLHUFR8=');
```

### Certyfikaty X509

Używanie **OpenSSL** jako IOHandler

**Certificate.Enabled:** włącz, jeśli uwierzytelnianie używa certyfikatów.



**Certificate.CertFile:** ścieżka do certyfikatu klienta X.509.

**Certificate.KeyFile:** ścieżka do pliku klucza klienta X.509.

**Certificate.Password:** hasło certyfikatu, jeśli zostało ustawione.

**Version:** Wersja TLS, domyślnie TLS 1.0

Użycie **SChannel** jako IOHandler

**Certificate.Enabled:** włącz, jeśli uwierzytelnianie używa certyfikatów.

**Certificate.CertFile:** ścieżka do certyfikatu PFX (certyfikat musi zostać wcześniej przekonwertowany do formatu PFX). [Więcej informacji.](#)

**Certificate.Password:** hasło certyfikatu, jeśli zostało ustawione.

**Version:** Wersja TLS, domyślnie TLS 1.0

Inne właściwości:

**MQTTHeartBeat:** jeśli włączone, próbuje utrzymać połączenie MQTT wysyłając ping co x sekund.

**Interval:** liczba sekund pomiędzy kolejnymi pingami.

**WatchDog:** jeżeli jest włączony, w przypadku wykrycia nieoczekiwanego rozłączenia podejmuje automatyczną próbę ponownego połączenia z serwerem.

**Interval:** liczba sekund przed kolejnymi próbami ponownego połączenia.

**Attempts:** maksymalna liczba prób ponownego połączenia; zero oznacza bez ograniczeń.

**LogFile:** jeśli włączone, zapisuje wiadomości gniazda do pliku dziennika (przydatne do debugowania). Dostęp do pliku dziennika nie jest bezpieczny wątkowo, gdy jest dostępny z wielu wątków.

**Enabled:** jeśli włączone, każda wiadomość odebrana i wysłana przez gniazdo będzie zapisywana do pliku.

**FileName:** pełna ścieżka do pliku.

Implementacja Azure MQTT opiera się na protokole MQTT w wersji 3.1.1, lecz odbiega od specyfikacji w następujący sposób:

- IoT Hub nie obsługuje wiadomości z QoS 2. Jeśli aplikacja urządzenia opublikuje wiadomość z QoS 2, IoT Hub zamknie połączenie sieciowe.
- IoT Hub nie przechowuje wiadomości Retain. Jeśli urządzenie wyśle wiadomość z flagą RETAIN ustawioną na 1, IoT Hub dodaje właściwość aplikacji x-opt-retain do wiadomości. W tym przypadku, zamiast przechowywać wiadomość retain, IoT Hub przekazuje ją do aplikacji zaplecza.
- IoT Hub obsługuje tylko jedno aktywne połączenie MQTT na urządzenie. Każde nowe połączenie MQTT dla tego samego identyfikatora urządzenia powoduje, że IoT Hub zrywa istniejące połączenie.

## Połączenie z Azure IoT Hub

Najpierw należy zalogować się na konto Azure, zarejestrować nowe urządzenie i utworzyć metodę uwierzytelniania dla tego urządzenia. Po wykonaniu tych kroków można utworzyć nowy `TsgcIoTAzure_MQTT_Client` i połączyć się z Azure IoT Hub.

Na przykład:

```
TsgcIoTAzure_MQTT_Client *oClient = new TsgcIoTAzure_MQTT_Client();
oClient->Azure->IoTHub = "youriothub.azure-devices.net";
oClient->Azure->DeviceId = "YourDeviceId";
oClient->SAS->Enabled = true;
oClient->SAS->SecretKey = "YourSecretKey";
oClient->OnMQTTConnect = OnMQTTConnectEvent;
oClient->Active = true;

private void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const TmqttConnReturnCode *ReturnCode)
{
    ShowMessage("Connected to Azure IoT Hub");
}
```

## Urządzenie do chmury

Przy wysyłaniu informacji z aplikacji urządzenia do zaplecza rozwiązania, IoT Hub udostępnia następujące opcje:

1. **Wiadomości z urządzenia do chmury** dla szeregów czasowych telemetrii i alertów.

```
oClient->Send_DeviceToCloud("{\"temp\": 10}", azuIoTQoS1);
```

Właściwości klucz-wartość można przysyłać za pomocą obiektu TStringList. Wystarczy wypełnić obiekt TStringList żdanymi właściwościami wiadomości i przekazać go jako argument.

```
TStringList *oProperties = new TStringList;
try
{
    oProperties->AddPair("prop_name1", "prop_value1");
    oProperties->AddPair("prop_name2", "prop_value2");
    oClient->Send_DeviceToCloud("{\"temp\": 10}", oProperties, azuIoTQoS1);
}
__finally
{
    oProperties->Free();
}
```

Jeśli konieczne jest ustawienie właściwości **ContentType** i **ContentEncoding** wiadomości, należy dodać te wartości do listy Properties. Nazwy tych właściwości są określone przez Azure.

|       |                  |
|-------|------------------|
| \$.ct | application/json |
| \$.ce | utf-8            |

```
TStringList *oProperties = new TStringList();
try
{
    oProperties->AddPair("$.ct", "application/json");
    oProperties->AddPair("$.ce", "utf-8");
    oClient->Send_DeviceToCloud("{\"temp\": 10}", oProperties, azuIoTQoS1);
}
__finally
{
    oProperties->Free();
}
```

**2. Zgłaszane właściwości bliźniaka urządzenia** służące do raportowania informacji o stanie urządzenia, takich jak dostępne możliwości, warunki lub stan długotrwałych przepływów pracy. Na przykład aktualizacje konfiguracji i oprogramowania.

```
oClient->Set_DeviceTwinsProperties("1", "{\"sgc\":1}");
```

## Cloud To Device

IoT Hub oferuje trzy opcje dla aplikacji urządzeń, które chcą udostępnić funkcjonalność aplikacji zaplecza:

1. **Metody bezpośrednie** do komunikacji wymagającej natychmiastowego potwierdzenia wyniku. Metody bezpośrednie są często stosowane do interaktywnego sterowania urządzeniami, na przykład do włączania wentylatora.

```
oClient->Subscribe_DirectMethod();
```

Na metody publiczne można odpowiadać przy użyciu następującej metody.

```
oClient->RespondPublicMethod(RequestId, Status, "Your Response", azuIoTQoS1);
```

2. **Żądane właściwości bliźniaka** dla długotrwałych poleceń mających na celu przełączenie urządzenia w określony pożądany stan. Na przykład ustaw interwał wysyłania telemetrii na 30 minut. Właściwości można odczytać za pomocą poniższej metody.

```
oClient->Get_DeviceTwinsProperties("1");
```

3. **Wiadomości z chmury do urządzenia** służące do jednokierunkowych powiadomień dla aplikacji na urządzeniu. Aby odbierać wiadomości, należy najpierw zasubskrybować odpowiedni temat.

```
oClient->Subscribe_CloudToDevice;
```

Wiadomości są odbierane w zdarzeniu **OnMQTTPublish** (ładunek tekstowy) lub **OnMQTTPublishEx** (ładunek jako TsgcWSMQTTPublishData z właściwościami Value, Bytes i Stream).

```
void AzureIoTMQTTPublish(TsgcWSConnection *Connection, string aTopic, string aText)
{
    DoLog("Received Message: " + aTopic + " " + aText);
}
```

## Przesyłanie plików

Centrum IoT ułatwia przesyłanie plików z podłączonych urządzeń, udostępniając im URI z podpisem dostępu współdzielonego (SAS) lub certyfikaty X509.

W przypadku wybrania opcji SAS należy ustawić następujące właściwości:

- **Azure.IoTHub:** przykład youriothub.azure-devices.net
- **Azure.DeviceId:** przykład: myDevice
- **SAS.SecretKey:** przykład: Yj7RRPnkSDTv+UCFLgwIP/FrbDymZv4qVAIoTLHUFR8=
- **SAS.KeyName:** przykład: iothubowner
- **SAS.Enabled:** wartość musi być ustawiona na true.

Jeśli wybrane zostaną certyfikaty X509, należy ustawić następujące właściwości:

- **Certificate.CertFile:** ścieżka do certyfikatu PEM.
- **Certificate.KeyFile:** ścieżka do pliku klucza PEM.

- **Certificate.Enabled:** wartość musi być ustawiona na true.

Należy użyć metody **UploadFile** w celu przesłania pliku na serwery Azure. Jeśli parametr Overwrite ma wartość true, istniejący plik zostanie zastąpiony. Jeśli parametr Overwrite ma wartość false, plik zostanie przesłany tylko wtedy, gdy nie istnieje; jeśli już istnieje, zostanie zgłoszony błąd (jest to opcja domyślna).

```
void UploadFileToAzure()
{
    TOpenDialog* oDialog = new TOpenDialog(NULL);
    try
    {
        if (oDialog->Execute())
        {
            AnsiString fileName = oDialog->FileName;
            AzureIoT::UploadFile(fileName.c_str());
        }
    }
    __finally
    {
        oDialog->Free();
    }
}
```

## Usługa aprowizacji urządzeń

Azure IoT umożliwia rejestrowanie urządzeń z poziomu kodu przy użyciu DPS. Biblioteka obsługuje obecnie rejestrowanie urządzenia poprzez przekazanie Scope Id i Registration Id jako parametrów.

```
TsgcIoTAzure_MQTT_Client* oClient = new TsgcIoTAzure_MQTT_Client(NULL);
try
{
    oClient->Certificate->CertFile = L"cert.pem";
    oClient->Certificate->KeyFile = L"key.pem";
    oClient->Certificate->Enabled = true;
    TsgcIoT_Azure_OperationRegistrationState* oResponse = new TsgcIoT_Azure_OperationRegistrationState();
    try
    {
        if (oClient->ProvisioningDeviceClient_Register(L"scope_id", L"registration_id", oResponse))
            ShowMessage(L"#Provisioning Register OK: " + oResponse->Status);
        else
            ShowMessage(L"#Provisioning Register Error: " + oResponse->Status);
    }
    __finally
    {
        delete oResponse;
    }
}
__finally
{
    delete oClient;
}
```

## Azure IoT Explorer

Aplikacja **Azure IoT Explorer** umożliwia interakcję z urządzeniami podłączonymi do centrum IoT Hub. Można w niej przeglądać odebrane wiadomości telemetryczne, zarejestrowane urządzenia i inne informacje. Aplikacja jest bezpłatna i można ją pobrać pod adresem:

<https://github.com/Azure/azure-iot-explorer/releases>

# HTTP

Protokół HTTP umożliwia pobieranie zasobów z serwerów, takich jak obrazy i dokumenty HTML. Jest to protokół klient-serwer, co oznacza, że klient żąda od serwera potrzebnych mu zasobów.

Gdy klient chce połączyć się z serwerem, wykonuje następujące kroki:

1. Otwórz nowe połączenie TCP
2. Wyślij wiadomość do serwera z żądanymi danymi

```
GET / HTTP/1.1
Host: server.com
Accept-Language: en-us
```

3. Odczytaj odpowiedź wysłaną przez serwer

```
HTTP/1.1 200 OK
Server: Apache
Content-Length: 120
Content-Type: text/html
```

...

## Komponenty

- **HTTP/2:** HTTP/2 (lub h2) to protokół binarny wprowadzający push, multipleksowanie strumieni i kontrolę ramek do sieci.
- **Klient HTTP/1:** komponent niewizualny dziedziczący po komponencie klienta TIdHTTP.
- **OAuth2:** OAuth2 umożliwia aplikacjom zewnętrznym uzyskanie ograniczonego dostępu do usługi HTTP.
- **JWT:** JWT umożliwia tworzenie danych z opcjonalnym podpisem i/lub szyfrowaniem, których ładunek zawiera JSON z określoną liczbą roszczeń.
- **Amazon SQS:** w pełni zarządzana usługa kolejkowania wiadomości dla mikroserwisów, systemów rozproszonych i aplikacji bezserwerowych.
- **Google Cloud Pub/Sub:** zapewnia przesyłanie wiadomości między aplikacjami i jest zaprojektowany w celu zapewnienia niezawodnej, wielokierunkowej, asynchronicznej komunikacji między aplikacjami.

- **Google Calendar:** umożliwia korzystanie z Google Calendar API V3: pobieranie kalendarzy, zdarzeń, synchronizację z własnym kalendarzem itd.
- **Google FCM:** wysyła powiadomienia za pośrednictwem Firebase Cloud Messaging.



# HTTP/2

HTTP/2 jest ewolucją protokołu HTTP 1.1. Jego głównym celem jest większa efektywność w sieci. Semantyka pozostaje taka sama, więc protokół jest zaprojektowany z myślą o zgodności ze starszymi protokołami.

## Ograniczenia HTTP 1.1

HTTP 1.1 jest ograniczony do przetwarzania jednego żądania na połączenie, dlatego klienci zazwyczaj używają więcej niż jednego połączenia do pobierania plików z serwerów. Rodzi to jednak problem, ponieważ przy zbyt wielu otwartych połączeniach TCP klienci rywalizują o zasoby serwera, a wydajność spada wraz ze wzrostem liczby klientów łączących się z serwerami.

## Główne funkcje

- HTTP/2 jest protokołem binarnym (dla przypomnienia: HTTP 1.1 jest protokołem tekstowym).
- HTTP/2 działa przez TLS i ALPN.
- Protokół jest multipleksowany (umożliwia wysyłanie więcej niż jednego żądania przez pojedyncze połączenie TCP).
- Serwer może przekazywać odpowiedzi do klientów.
- Skraca czasy podróży w obie strony, dzięki czemu klienci mogą ładować się szybciej.

HTTP/2 wprowadza inne udoskonalenia, więcej szczegółów: [RFC7540](#).

HTTP/2 wymaga naszej niestandardowej wersji Indy, ponieważ wymaga protokołu ALPN.

## Komponenty

- **[TsgcHTTP2Client](#)**: komponent klienta z pełną obsługą protokołu HTTP/2 (kod Pascal w 100% sgcWebSockets, bez zewnętrznych bibliotek).
- **[TsgcWebSocketHTTPServer](#)**: komponent serwerowy obsługujący w pełni protokół HTTP/2 (kod Pascal sgcWebSockets w 100%, bez zewnętrznych bibliotek). Domyślnie HTTP/2 jest wyłączony; można go włączyć za pomocą właściwości HTTPOptions, ustawiając Enable = true.
- **[TsgcWebSocketServer\\_HTTPAPI](#)**: komponent serwera obsługujący protokół HTTP/2 (implementacja Microsoft, wymaga systemu Windows 2016+ lub Windows 10+).
- **[Serwery DataSnap](#)**: serwer datasnap może również obsługiwać protokół HTTP/2.

## API

- **[Apple Push Notifications](#)**: wysyłanie powiadomień widocznych dla użytkownika na jego urządzenie z poziomu dostawcy serwera.



# TsgcHTTP2Client

TsgcHTTP2Client implementuje komponent klienta HTTP/2 i może łączyć się z serwerami HTTP/2.

## Wprowadzenie

TsgcHTTP2Client implementuje komponent klienta HTTP/2 i może łączyć się z serwerami HTTP/2. Aby skonfigurować ten komponent, należy wykonać poniższe kroki:

1. Utwórz nową instancję składnika **TsgcHTTP2Client**.
2. Wyślij żądanie do serwera i przetwórz odpowiedź przy użyciu zdarzenia OnHTTP2Response. Przykład:

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2Response = OnHTTP2ResponseEvent;
oClient->Get("https://www.google.com");

void OnHTTP2ResponseEvent(TObject *Sender, const
    TsgcHTTP2ConnectionClient *Connection, const TsgcHTTP2RequestProperty *Request,
    const TsgcHTTP2ResponseProperty *Response)
{
    ShowMessage(Response->DataString);
}
```

## Dokumentacja

- Właściwości — 19 właściwości dostępnych w komponencie.
- Metody — 19 metod publicznych.
- Zdarzenia — 13 zdarzeń do obsługi.

## Przewodniki

- **Żądania**
  - [Metoda żądania HTTP/2](#)
  - [HTTP/2 Server Push](#)
  - [Pobieranie pliku](#)
  - [HTTP/2 Partial Responses](#)
  - [HTTP/2 Headers](#)
- **Połączenie**
  - [Zamknięcie połączenia klienta](#)
  - [Utrzymanie aktywnego połączenia klienta](#)
  - [Przyczyna rozłączenia HTTP/2](#)
  - [Oczekujące żądania klienta](#)

- **Uwierzytelnianie**

- [Uwierzytelnianie klienta](#)
- [HTTP/2 i OAuth2](#)

- **Klasy**

- [TsgcHTTP2ConnectionClient](#)
- [TsgcHTTP2RequestProperty](#)
- [TsgcHTTP2ResponseProperty](#)

# TsgcHTTP2Client | Metoda żądania HTTP/2

Klient HTTP/2 może działać w trybie blokującym i nieblokującym; wewnętrznie komponent działa w wątku pomocniczym, a żądania są przetwarzane asynchronicznie, jednak można wywołać żądanie i poczekać na jego zakończenie.

Poniżej przedstawiono przykład pokazujący, jak klient może zażądać strony HTML od serwera HTTP/2 i jak można pracować w obu trybach.

## Tryb asynchroniczny

Pobierz następujący adres URL: <https://www.google.com> i otrzymaj powiadomienie, gdy klient odbierze pełną odpowiedź. Po wywołaniu metody **GETASYNC** przetwarzanie jest kontynuowane, a zdarzenie **OnHTTP2Response** jest wywoływane po odebraniu odpowiedzi.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2Response = OnHTTP2ResponseEvent;
oClient->GetAsync("https://www.google.com");
void OnHTTP2ResponseEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection,
    const TsgcHTTP2RequestProperty *Request, const TsgcHTTP2ResponseProperty *Response)
{
    ShowMessage(Response->Headers->Text + #13#10 + Response->DataString);
}
```

## Tryb blokujący

Pobierz następujący adres URL: <https://www.google.com> i czekaj, aż klient odbierze pełną odpowiedź. Po wywołaniu metody **GET** proces czeka, aż odpowiedź zostanie odebrana lub upłynie limit czasu.

Dostęp do surowych danych odpowiedzi można uzyskać za pomocą właściwości **Response** klienta HTTP/2. Umożliwia ona dostęp do surowych nagłówków, kodu statusu odpowiedzi, zestawu znaków i innych informacji.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
vResponse = oClient->Get("https://www.google.com");
if (oClient->Response->Status == 200)
    ShowMessage("Response from server: " + vResponse);
else
    ShowMessage("Response Code: " + IntToStr(oClient->Response->Status));
```

## Żądania | HTTP/2 Server Push

Server Push to możliwość wysyłania przez serwer wielu odpowiedzi na jedno żądanie klienta. Oznacza to, że oprócz odpowiedzi na oryginalne żądanie serwer może wypychać do klienta dodatkowe zasoby bez konieczności jawnego żądania każdego z nich przez klienta.



Za każdym razem, gdy serwer wysyła do klienta wiadomość PushPromise, wywoływane jest zdarzenie OnHTTP2PushPromise. Gdy klient odbierze PushPromise, oznacza to, że serwer wyśle ten zasób w kolejnych pakietach, a klient może go zaakceptować lub odrzucić.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2PushPromise = OnHTTP2PushPromiseEvent;
oClient->Get("https://http2.golang.org/serverpush");
...
void OnHTTP2PushPromiseEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection,
    const TsgcHTTP2_Frame_PushPromise *PushPromise, bool &Cancel)
{
    if (PushPromise->URL == "/serverpush/static/godocs.js")
    {
        Cancel = true;
    }
    else
    {
        Cancel = false;
    }
}
```

# TsgcHTTP2Client | Pobieranie pliku HTTP/2

Gdy klient żąda pliku od serwera, należy użyć zdarzenia `OnHTTP2Response`, aby załadować odpowiedź strumieniową.

## Pobieranie dużych plików

Podczas pobierania dużych plików (setki MB lub więcej) należy ustawić **`HTTP2Options.ReadTimeout`** na **0** (brak limitu czasu), aby transfer zakończył się bez przerwania przez domyślny 60-sekundowy limit czasu:

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->HTTP2Options->ReadTimeout = 0; // no timeout for large files
oClient->Get("https://server/largefile", oStream);
```

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2Response = OnHTTP2ResponseEvent;
oClient->Get("https://http2.golang.org/file/gopher.png");
...
void OnHTTP2ResponseEventEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection,
    const TsgcHTTP2RequestProperty *Request, const TsgcHTTP2ResponseProperty *Response)
{
    TFileStream *oStream = new TFileStream("file", fmOpenWrite | fmCreate);
    try
    {
        oStream->CopyFrom(Response->Data, Response->Data->Size);
    }
    __finally
    {
        oStream->Free();
    }
}
```

## TsgcHTTP2Client | HTTP/2 Częściowe odpowiedzi

Zazwyczaj po wysłaniu żądania HTTP serwer wysyła odpowiedź z żądanym plikiem, jednak czasami zamiast pojedynczej odpowiedzi serwer może wysyłać wiele odpowiedzi w postaci strumienia. W takich przypadkach można użyć zdarzenia **OnHTTP2ResponseFragment**, aby przechwycić te odpowiedzi i wyświetlić je użytkownikowi.

**Przykład:** wyślij żądanie do <https://http2.golang.org/clockstream>, a serwer będzie wysyłał odpowiedź strumieniową co sekundę.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2ResponseFragment = OnHTTP2ResponseFragmentEvent;
oClient->Get("https://http2.golang.org/clockstream");
...
void OnHTTP2ResponseFragmentEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection,
    const TsgcHTTP2RequestProperty *Request, const TsgcHTTP2ResponseFragmentProperty *Fragment)
{
    ShowMessage(Fragment->DataString);
}
```



## TsgcHTTP2Client | HTTP/2 Headers

TsgcHTTP2Client umożliwia dostosowanie nagłówków wysyłanych do serwera podczas łączenia się klienta

**Przykład:** jeśli chcesz dodać nagłówek HTTP „Client: sgcWebSockets”

```
void OnHTTP2BeforeRequest(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection,
    ref TStringList *Headers)
{
    Headers->Add("Client: sgcWebSockets");
}
```

Można również użyć właściwości Request.CustomHeaders, aby dodać własne nagłówki.

# TsgcHTTP2Client | Zamknięcie połączenia klienta

Połączenie można zamknąć za pomocą właściwości `Active` lub metod `Close/Disconnect`.

## Właściwość `Active`

Gdy połączenie jest aktywne i zostanie ustawione `Active := False`, połączenie zostanie natychmiast zamknięte bez wysyłania do serwera jakiegokolwiek wiadomości o rozłączeniu.

## Rozłącz

Można użyć metody `Disconnect` (z `TsgcHTTP2Client` lub `TsgcHTTP2ConnectionClient`) w celu rozłączenia gniazda.

## `Close`

Ta metoda wysyła do serwera wiadomość informującą o zamknięciu połączenia, z możliwością opcjonalnego podania informacji o przyczynie rozłączenia. Jest to poprawny sposób zamykania połączenia HTTP/2.

Metodę `Close` można wywołać z obiektów `TsgcHTTP2Client` lub `TsgcHTTP2ConnectionClient`.

Możliwe są następujące powody błędów:

- brak błędu
- błąd protokołu
- błąd wewnętrzny
- błąd kontroli przepływu
- limit czasu ustawień
- strumień zamknięty
- błąd rozmiaru ramki
- odmówiono strumienia
- cancel
- błąd kompresji
- błąd połączenia
- enhance your calm
- niewystarczające zabezpieczenia
- wymagane

# TsgcHTTP2Client | Utrzymywanie aktywnego połączenia klienta

Po nawiązaniu połączenia klienta z serwerem połączenie może niekiedy zostać przerwane z powodu słabego sygnału lub błędów połączenia. Dostępne są 2 właściwości pomagające utrzymać aktywne połączenie.

## HeartBeat

**HeartBeat** właściwość umożliwia **wysyłanie polecenia Ping** co **X sekund** w celu **utrzymania aktywnego połączenia**. Niektóre serwery zamykają połączenia TCP, jeśli między węzłami nie są wymieniane żadne dane. HeartBeat rozwiązuje ten problem, wysyłając polecenie ping w określonych odstępach czasu. Zazwyczaj jest to wystarczające do utrzymania aktywnego połączenia.

Właściwość HeartBeatType umożliwia dostosowanie sposobu działania HeartBeat:

1. **hbtAlways**: wysyła ping co x sekund zdefiniowanych w polu Interval.
2. **hbtOnlyIfNoMsgRcvInterval**: wysyła ping co x sekund tylko wtedy, gdy w ciągu ostatnich x sekund zdefiniowanych we właściwości Interval nie odebrano żadnych wiadomości.

**Przykład**: wysyłanie pinga co 30 sekund

```
oClient = new TsgcHTTP2Client();
oClient->HeartBeat->Interval = 30;
oClient->HeartBeat->Enabled = true;
oClient->Active = true;
```

## WatchDog

Jeśli WatchDog jest włączony, gdy klient wykryje rozłączenie, WatchDog będzie próbował ponownie nawiązać połączenie co X sekund, aż do przywrócenia aktywnego połączenia.

**Przykład**: ponowne połączenie co 10 sekund po rozłączeniu z nieograniczoną liczbą prób.

```
oClient = new TsgcHTTP2Client();
oClient->WatchDog->Interval = 10;
oClient->WatchDog->Attempts = 0;
oClient->WatchDog->Enabled = true;
oClient->Active = true;
```

## TsgcHTTP2Client | Przyczyna rozłączenia HTTP/2

Serwer HTTP/2 może rozłączyć klienta z kilku powodów; gdy serwer chce poinformować klienta o przyczynie rozłączenia, wysyła do niego wiadomość **GoAway** z informacjami o rozłączeniu.

Użyj zdarzenia `OnHTTP2GoAway`, aby przechwycić przyczynę rozłączenia przez serwer (jeśli klient chce zamknąć połączenie, może użyć metody `close`, aby wysłać przyczynę zamknięcia).

**TsgcHTTP2GoAwayProperty** Obiekt zawiera informacje o rozłączeniu

- **LastStreamId:** to ostatni strumień przetworzony przez serwer.
- **ErrorCode:** liczba całkowita identyfikująca kod błędu.
- **ErrorDescription:** opis błędu, jeden z poniższych:
  - brak błędu
  - błąd protokołu
  - błąd wewnętrzny
  - błąd kontroli przepływu
  - limit czasu ustawień
  - strumień zamknięty
  - błąd rozmiaru ramki
  - odmówiono strumienia
  - cancel
  - błąd kompresji
  - błąd połączenia
  - enhance your calm
  - niewystarczające zabezpieczenia
  - wymagane
- **AdditionalDebugData:** opcjonalny ciąg zawierający dodatkowe informacje o rozłączeniu.

## TsgcHTTP2Client | Oczekujące żądania klienta

Gdy klient wysyła kilka żądań, są one przetwarzane w wątku pomocniczym. Niekiedy połączenie może zostać zamknięte z różnych przyczyn, a żądania mogą jeszcze oczekiwać na obsługę. Zdarzenie `OnHTTP2PendingRequests` służy do obsługi tych oczekujących żądań. Jest wywoływane, gdy klient wykryje rozłączenie i nadal istnieją oczekujące żądania. Zdarzenie to posiada 2 parametry:

1. **Reconnect:** domyślnie wyłączone; jeśli ustawione na `true`, klient będzie automatycznie nawiązywał ponowne połączenie.
2. **Clear:** domyślnie włączone, jeśli ustawione na `false`, gdy klient połączy się ponownie, spróbuje ponownie wysłać oczekujące żądania do serwera.

# TsgcHTTP2Client | Uwierzytelnianie klienta

Klient HTTP/2 obsługuje 2 typy uwierzytelniania: uwierzytelnianie podstawowe (Basic Authentication) i uwierzytelnianie OAuth2.

Należy użyć zdarzenia **OnHTTP2Authorization**, aby obsługiwać oba typy uwierzytelniania.

## Uwierzytelnianie podstawowe

Jeśli serwer zwraca nagłówek żądający uwierzytelnienia Basic, należy ustawić nazwę użytkownika i hasło w OnHTTP2Authorization.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2Authorization = OnHTTP2AuthorizationEvent;
...
void OnHTTP2AuthorizationEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection, const string AuthType, const string Username, const string Password)
{
    if (AuthType == "Basic")
    {
        Username = "user";
        Password = "secret";
    }
}
```

## Bearer Token

Jeśli serwer zwróci nagłówek z żądaniem uwierzytelnienia tokenem Bearer, należy ustawić token w zdarzeniu OnHTTP2Authorization.

```
TsgcHTTP2Client *oClient = new TsgcHTTP2Client();
oClient->OnHTTP2Authorization = OnHTTP2AuthorizationEvent;
...
void OnHTTP2AuthorizationEvent(TObject *Sender, const TsgcHTTP2ConnectionClient *Connection, const string AuthType, const string Token)
{
    if (AuthType == "Bearer")
    {
        Token = "bearer token";
    }
}
```

## Wartość Bearer od strony trzeciej

Jeśli wartość Bearer jest już znana, ponieważ została uzyskana inną metodą, można ją przekazać jako nagłówek HTTP przy użyciu następujących właściwości żądania; należy je ustawić przed wywołaniem dowolnej metody żądania HTTP:

```
TsgcHTTP2Client.Request.BearerAuthentication  
= true
```

```
TsgcHTTP2Client.Request.BearerToken = "< value of the token >"
```

## OAuth2

Przeczytaj następujący artykuł, jeśli chcesz używać naszego [komponentu OAuth2 z klientem HTTP/2](#).

## TsgcHTTP2Client | HTTP/2 i OAuth2

OAuth2 to powszechna metoda autoryzacji stosowana przez wiele firm, takich jak Google. Gdy chcemy uwierzytelnić się na serwerach Google w celu korzystania z ich API, zazwyczaj wymagane jest uwierzytelnianie przy użyciu OAuth2.

sgcWebSockets obsługuje OAuth2 w kliencie HTTP/2. Dostępna jest właściwość o nazwie **Authentication.Token.OAuth**, do której należy przypisać instancję [TsgcHTTP\\_OAuth2](#).

### Jak połączyć się z GMail Google API

Aby nawiązać połączenie z API Google, należy utworzyć instancję TsgcHTTP\_OAuth2 i wypełnić następujące dane:

```
TsgcHTTP_OAuth1.AuthorizationServerOptions.AuthURL := 'https://accounts.google.com/o/oauth2/auth';
TsgcHTTP_OAuth1.AuthorizationServerOptions.TokenURL := 'https://accounts.google.com/o/oauth2/token';
```

```
TsgcHTTP_OAuth1.LocalServerOptions.IP := '127.0.0.1';
TsgcHTTP_OAuth1.LocalServerOptions.Port := 8080;
```

```
TsgcHTTP_OAuth1.OAuth2Options.ClientId := 'your client id';
TsgcHTTP_OAuth1.OAuth2Options.ClientSecret := 'your client secret';
```

Po wypełnieniu komponentu klienta OAuth2 należy utworzyć nową instancję TsgcHTTP2Client i przypisać komponent OAuth2 do klienta HTTP/2.

```
TsgcHTTP2Client1.Authentication.Token.OAuth := TsgcHTTP_OAuth1;
```

Na koniec wysyłane jest żądanie pobrania listy wiadomości z konta yourname@gmail.com.

```
oStream := TStringStream.Create('');
Try
  TsgcHTTP2Client1.Get('https://gmail.googleapis.com/gmail/v1/users/yourname@gmail.com/messages', oStream);
  ShowMessage(oStream.DataString);
Finally
  oStream.Free;
End;
```





# TsgcHTTP2ConnectionClient

TsgcHTTP2ConnectionClient jest opakowaniem dla połączeń HTTP/2 po stronie klienta; dostęp do tego obiektu można uzyskać w zdarzeniach klienta.

## Metody

- **Ping:** wysyła ping do serwera w celu podtrzymania aktywnego połączenia.
- **Close:** wysyła wiadomość do serwera z informacją o powodzie rozłączenia.
- **Disconnect:** zamyka połączenie bez wysyłania jakiegokolwiek wiadomości informacyjnej do serwera.
- **Metody HTTP/2:**

**GET:** Metoda GET żąda reprezentacji określonego zasobu. Żądania przy użyciu metody GET powinny służyć wyłącznie do pobierania danych.

**HEAD:** Metoda HEAD żąda odpowiedzi identycznej z odpowiedzią na żądanie GET, lecz bez treści odpowiedzi.

**POST:** Metoda POST służy do przesyłania danych do określonego zasobu, często powodując zmianę stanu lub efekty uboczne po stronie serwera.

**PUT:** metoda PUT zastępuje wszystkie bieżące reprezentacje zasobu docelowego ładunkiem żądania.

**DELETE:** Metoda HEAD żąda odpowiedzi identycznej jak w przypadku żądania GET, lecz bez treści odpowiedzi.

**CONNECT:** Metoda CONNECT ustanawia tunel do serwera identyfikowanego przez zasób docelowy.

**OPTIONS:** Metoda OPTIONS służy do opisanie opcji komunikacji dla zasobu docelowego.

**TRACE:** Metoda TRACE wykonuje test pętli zwrotnej wiadomości wzdłuż ścieżki do zasobu docelowego.

**PATCH:** Metoda PATCH służy do stosowania częściowych modyfikacji zasobu.

# TsgcHTTP2RequestProperty

Obiekt ten jest przekazywany jako argument zdarzenia OnHTTP2Response i umożliwia poznanie oryginalnego żądania odpowiedzi wysłanej przez serwer.

## Właściwości

- **Metoda:** identyfikuje metodę HTTP/2 (GET, POST...)
- **URL:** to żądany adres URL.
- **Request:** zawiera pola żądania.

# TsgcHTTP2ResponseProperty

Ten obiekt jest odbierany jako argument zdarzenia OnHTTP2Response; umożliwia poznanie odpowiedzi wysłanej przez serwer do klienta.

## Właściwości

- **Headers:** zawiera listę nieprzetworzone nagłówków odebranych od serwera.
- **Data:** zawiera surową treść odpowiedzi serwera na żądanie.
- **DataStream:** jest wynikiem konwersji pola Data na ciąg tekstowy.
- **DataUTF8:** konwersja danych Data do ciągu znaków UTF-8.
- **PushPromise:** jeśli jest przypisany, zawiera obiekt PushPromise wysłany przez serwer do klienta (oznacza to, że ten obiekt odpowiedzi nie był żądany przez klienta).

# HTTP2 | Apple Push Notifications

[https://developer.apple.com/documentation/usernotifications/setting\\_up\\_a\\_remote\\_notification\\_server](https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server)

Apple umożliwia wysyłanie powiadomień push do urządzeń Apple przy użyciu usługi Apple Push Notification Service (APNs).

Aby wysłać powiadomienie do urządzenia, dostawca musi wysłać żądanie HTTP/2 POST do APNs zawierające następujące informacje:

- Ładunek JSON z informacjami, które mają zostać wysłane.
- Token urządzenia identyfikujący urządzenie użytkownika.
- Niektóre nagłówki HTTP dotyczące sposobu dostarczania powiadomień.
- Certyfikat SSL lub token JWT do uwierzytelnienia żądania wobec APNs



## Co jest wymagane do wysyłania powiadomień

Aby wysłać powiadomienia do urządzenia przy użyciu Rad Studio, należy wykonać następujące kroki:

- Zarejestruj aplikację w APNs
- [Generowanie powiadomienia zdalnego](#)
- [Wysyłanie żądań powiadomień do APNs](#)
  - [Połączenie oparte na tokenach z APNs](#)
  - [Połączenie oparte na certyfikacie z APNs](#)

# APN | Generowanie zdalnego powiadomienia APNs

[https://developer.apple.com/documentation/usernotifications/setting\\_up\\_a\\_remote\\_notification\\_server/generating\\_a\\_remote\\_notification](https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/generating_a_remote_notification)

Powiadomienia Apple wykorzystują ładunek JSON do wysyłania obiektu powiadomienia. Maksymalny rozmiar ładunku wynosi 4096 bajtów.

## Przykłady ładunków JSON

Prosta wiadomość alertu

```
{
  "aps":{
    "alert":"Alert from sgcWebSockets!"
  }
}
```

Alert z tytułem i podtytułem.

```
{
  "aps" : {
    "alert" : {
      "title" : "Game Request",
      "subtitle" : "Five Card Draw"
      "body" : "Bob wants to play poker",
    },
    "category" : "GAME_INVITATION"
  },
  "gameID" : "12345678"
}
```

# APN | Wysyłanie żądań powiadomień do APNs

[https://developer.apple.com/documentation/usernotifications/setting\\_up\\_a\\_remote\\_notification\\_server](https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server)

Wyślij ładunek zdalnego powiadomienia i informacje o tokenie urządzenia do usługi Apple Push Notification (APNs).

## Jak połączyć się z APNs

Aby nawiązać pomyślne połączenie między dostawcą serwera a jednym z poniższych serwerów, należy użyć protokołu HTTP/2 oraz co najmniej TLS 1.2.

**Serwer deweloperski:** <https://api.sandbox.push.apple>

**Serwer produkcyjny:** <https://api.push.apple>

## Przykładowy kod

Utwórz nową instancję TsgcHTTP2Client i wywołaj metodę POST, aby wysłać powiadomienie do APNs.

```
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();
}
```

Aby wysyłać powiadomienia, należy ustawić zaufanie z APNs oparte na **tokenach** lub **certyfikatach**, korzystając z protokołu HTTP/2 i TLS 1.2 lub nowszego.

- [Połączenie z APNs oparte na tokenach](#)
- [Połączenie z APNs oparte na certyfikacie](#)



# APNs Trusted | Połączenie oparte na tokenach z APNs

[https://developer.apple.com/documentation/usernotifications/setting\\_up\\_a\\_remote\\_notification\\_server/establishing\\_a\\_token-based\\_connection\\_to\\_apns](https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/establishing_a_token-based_connection_to_apns)

Zabezpiecz komunikację za pomocą Apple Push Notification service (APNs) przy użyciu bezstanowych tokenów uwierzytelniających.

Najpierw należy uzyskać **Klucz szyfrowania** oraz **Identyfikator klucza** z konta Apple Developer. Po pomyślnej rejestracji otrzymany zostanie 10-znakowy ciąg będący identyfikatorem klucza oraz klucz podpisywania tokenu uwierzytelniania w postaci pliku z rozszerzeniem .p8.

Należy użyć **klienta JWT** sgcWebSockets, aby wygenerować JWT przy użyciu **ES256** jako algorytmu. Token nie powinien być generowany dla każdego żądania HTTP/2, nie należy go odświeżać wcześniej niż po 20 minutach i później niż po 60 minutach.

## Konfiguracja klienta JWT

Skonfiguruj klienta JWT następującymi wartościami:

- **JWTOptions.Header.Algorithm:** to algorytm szyfrowania użyty do zaszyfrowania tokenu. APNs obsługuje wyłącznie algorytm ES256.
- **JWTOptions.Header.kid:** jest 10-znakowym identyfikatorem klucza uzyskanym z konta dewelopera.
- **JWTOptions.Payload.iss:** wartość, którą stanowi 10-znakowy Team ID używany do tworzenia aplikacji firmy. Wartość tę należy uzyskać ze swojego konta dewelopera.
- **JWTOptions.Payload.iat:** Czas wydania tokenu; wartość wskazuje moment wygenerowania tego tokenu JSON. Podaj wartość jako liczbę sekund od Epoch w UTC. Wartość nie może być o więcej niż jedną godzinę wcześniejsza od bieżącego czasu.
- **JWTOptions.RefreshTokenAfter:** ustaw wartość w sekundach równą 40 minutom (60\*40).

Korzystanie z połączeń opartych na tokenach wymaga wysłania nagłówka **apns-topic** z wartością identyfikatora pakietu/identyfikatora aplikacji (np. com.example.application).

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

```
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;
```

```
oHTTP->Request->CustomHeaders->Clear();
```

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

# Połączenie z APNs oparte na certyfikatach

[https://developer.apple.com/documentation/usernotifications/setting\\_up\\_a\\_remote\\_notification\\_server/establishing\\_a\\_certificate-based\\_connection\\_to\\_apns](https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/establishing_a_certificate-based_connection_to_apns)

Komunikację z usługą Apple Push Notification service (APNs) można zabezpieczyć przy użyciu certyfikatu uzyskanego od firmy Apple.



Najpierw należy zalogować się na konto dewelopera i **utworzyć nowy certyfikat** dla usługi Apple Push Notification

Po pobraniu certyfikatu klient HTTP/2 sgcWebSockets umożliwia korzystanie z 2 procedur obsługi bezpieczeństwa IOHandler (wyłącznie w systemie Windows; dla innych platform obsługiwany jest jedynie OpenSSL).

- OpenSSL
- SChannel (tylko dla systemu Windows)

## OpenSSL

Jeśli używany jest OpenSSL, należy wdrożyć biblioteki OpenSSL razem z aplikacją. Przed ustawieniem certyfikatu za pomocą [TsgcHTTP2Client](#) certyfikat ten musi najpierw zostać przekonwertowany do formatu PEM, ponieważ OpenSSL nie umożliwia bezpośredniego importu certyfikatów P12.

Aby skonwertować pojedynczy certyfikat P12 do formatu PEM i pliku klucza prywatnego, należy użyć następujących poleceń

### utwórz plik certyfikatu PEM

```
openssl pkcs12 -in INFILE.p12 -out OUTFILE.crt -nokeys
```

### Utwórz plik klucza prywatnego

```
openssl pkcs12 -in INFILE.p12 -out OUTFILE.key -nodes -nocerts
```

Po uzyskaniu certyfikatu i klucza prywatnego w formacie PEM można skonfigurować [TsgcHTTP2Client](#) w sposób przedstawiony poniżej.

```
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;
```

## SChannel

W przypadku korzystania z SChannel nie ma potrzeby wdrażania żadnych bibliotek, a certyfikat pobrany od Apple można bezpośrednio zaimportować bez konieczności wcześniejszej konwersji do formatu PEM.

Ustaw właściwość UseLegacyCredentials na True podczas korzystania z SChannel jako IOHandler.

```
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;
```

## Błędy

Jeśli pojawi się błąd „**missing topic**”, najprawdopodobniej używany jest certyfikat uniwersalny (certyfikaty, które mogą być używane do powiadomień push, VoIP itp.), który wymaga ustawienia nazwy tematu na wartość identyfikatora pakietu / identyfikatora aplikacji (przykład: com.example.application). Należy ustawić nagłówek apns-topic z odpowiednią wartością we właściwości Request klienta HTTP/2.

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

# HTTP/1

**TsgcHTTP1Client** jest komponentem niewiizualnym, dziedziczącym po komponencie TIdHTTP indy, i dodającym kilka nowych właściwości.

Ten komponent znajduje się w jednostce sgcHTTP.

## TLSOptions

Umożliwia konfigurowanie sposobu łączenia się z bezpiecznymi serwerami SSL/TLS przy użyciu protokołu HTTP/1.

**ALPNProtocols:** lista protokołów ALPN, które zostaną wysłane do serwera.

**RootCertFile:** ścieżka do pliku certyfikatu głównego.

**CertFile:** ścieżka do pliku certyfikatu.

**KeyFile:** ścieżka do pliku klucza certyfikatu.

**Password:** jeśli certyfikat jest zabezpieczony hasłem, należy je tu podać.

**VerifyCertificate:** jeśli certyfikat musi być weryfikowany, należy włączyć tę właściwość. Do dostosowania weryfikacji SSL należy użyć zdarzenia **OnSSLVerifyPeer**.

**VerifyDepth:** właściwość całkowitoliczbowa określająca maksymalną liczbę ogniw dozwolonych podczas weryfikacji certyfikatu X.509.

**Wersja:** domyślnie używa TLS 1.0. Jeśli serwer wymaga wyższej wersji TLS, można ją tutaj wybrać.

**Proxy:** w tej sekcji można określić, czy połączenie ma odbywać się przez serwer proxy. Obsługiwane są następujące serwery proxy:

**pxyHTTP:** Serwer proxy HTTP.

**pxySocks4:** Serwer proxy SOCKS4.

**pxySocks4A:** Serwer proxy SOCKS4A.

**pxySocks5:** Serwer proxy SOCKS5.

**IOHandler:** wybierz bibliotekę, której chcesz użyć do połączenia przez TLS.

**iohOpenSSL:** korzysta z biblioteki OpenSSL i jest domyślnym wyborem dla komponentów Indy. Wymaga wdrożenia bibliotek OpenSSL dla win32/win64.

**iohSChannel:** używa Secure Channel — protokołu bezpieczeństwa zaimplementowanego przez Microsoft dla systemu Windows. Nie wymaga wdrożenia bibliotek OpenSSL. Działa wyłącznie w systemie Windows 32/64-bit.

**OpenSSL\_Options:** konfiguracja bibliotek OpenSSL.

**APIVersion:** pozwala określić, które API OpenSSL będzie używane.

**osslAPI\_1\_0:** używa API 1.0 OpenSSL; jest to najnowsza wersja obsługiwana przez Indy

**osslAPI\_1\_1:** używa interfejsu API 1.1 OpenSSL, wymaga niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**osslAPI\_3\_0:** korzysta z API OpenSSL 3.0, wymaga naszej niestandardowej biblioteki Indy i umożliwia używanie bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek OpenSSL

**osslNone:** wartość domyślna. Biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanych ścieżkach systemowych.

**osslDefaultFolder:** automatycznie ustawia ścieżkę OpenSSL, w której biblioteki powinny się znajdować dla wszystkich rodzajów projektów IDE.

**osslCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.

**LibPathCustom:** gdy LibPath = osslCustomFolder, należy tu podać pełną ścieżkę do katalogu, w którym znajdują się biblioteki OpenSSL.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**osslSymLinksDefault:** domyślnie dowiązania symboliczne są włączone, z wyjątkiem OSX64 (macOS Monterey i nowsze wersje nie działają przy próbie załadowania biblioteki bez numeru wersji).

**osslSymLinksLoadFirst:** wczytaj najpierw dowiązania symboliczne, przed próbą załadowania bibliotek z numerem wersji.

**osslSymLinksLoad:** ładuje dowiązania symboliczne po próbie załadowania bibliotek z wersją.

**osslSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

**MinVersion:** tutaj należy ustawić minimalną wersję, której klient użyje do połączenia z bezpiecznym serwerem. Domyślnie wartość wynosi `tlsUndefined`, co oznacza, że minimalna wersja jest taka sama jak ustawiona we właściwości `Version`. Przykład: aby klient łączył się wyłącznie przy użyciu TLS 1.2 lub TLS 1.3, należy ustawić następujące wartości.

```
SSLOptions.Version := tls1_3;
```

```
SSLOptions.OpenSSL_Options.MinVersion := tls1_2;
```

**X509Checks:** użyj tej właściwości, aby włączyć dodatkowe weryfikacje certyfikatu X509:

**Mode:** wybierz, które opcje będą walidowane

**osslX509chHostName:** weryfikuje certyfikat nazwy hosta.

**oslx509chIPAddress:** weryfikuje adres IP certyfikatu.

**HostName:** ustaw nazwę hosta, jeśli różni się od żądania.

**IPAddress:** ustawia adres IP, jeśli różni się od adresu z żądania.

**SChannel\_Options:** umożliwia korzystanie z certyfikatu z magazynu certyfikatów systemu Windows.

**CertHash:** to skrót certyfikatu. Skrót certyfikatu można znaleźć, uruchamiając polecenie dir w programie PowerShell.

**CipherList:** w tym miejscu można określić, które szyfry będą używane (oddzielone znakiem ":" ). Przykład: CALG\_AES\_256:CALG\_AES\_128

**CertStoreName:** nazwa magazynu, w którym przechowywany jest certyfikat. Należy wybrać jedną z następujących wartości:

**scsnMY** (wartość domyślna)

**scsnCA**

**scsnRoot**

**scsnTrust**

**CertStorePath:** ścieżka magazynu, w którym przechowywany jest certyfikat. Należy wybrać jedną z następujących opcji:

**scspStoreCurrentUser** (domyślne)

**scspStoreLocalMachine**

## Dziennik

Gdy właściwość Log jest włączona, zapisuje komunikaty gniazda do wskazanego pliku dziennika, co jest przydatne podczas debugowania.

**LogOptions.FileName:** pełna ścieżka do pliku.

## Uwierzytelnianie

Umożliwia uwierzytelnianie przy użyciu **OAuth2** lub **JWT**.

## Żądania asynchroniczne

Domyślnie HTTP1Client używa żądań blokujących, więc po wywołaniu metody żądania HTTP klient czeka na odpowiedź od serwera. Alternatywnie można użyć metod asynchronicznych do wykonywania żądań HTTP w wątku pomocniczym, co pozwala uniknąć blokowania wątku wywołującego. Zaimplementowane są następujące metody asynchroniczne:

- GetAsync
- PostAsync
- PutAsync
- OptionsAsync
- DeleteAsync

Po wywołaniu tych metod kod nie czeka na odpowiedź, lecz przechodzi do kolejnej linii, a odpowiedź może być obsługiwana za pomocą zdarzenia **OnAsyncResponse**.

```
void __fastcall OnAsyncResultEvent(TObject* Sender, const TsgcHTTPAsyncRequest* aRequest, const TIdHTTPResponse* aResponse)
```

Jeśli podczas przetwarzania żądania asynchronicznego wystąpi błąd, wyjątek zostanie zgłoszony w zdarzeniu **OnAsyncException**.

## Przykłady

Wysyła żądanie metodą **GET** do serwera HTTPS z użyciem **TLS 1.2**

```
TsgcHTTP1Client *oHTTP = new TsgcHTTP1Client();
try
{
    oHTTP->TLSOptions->Version = tls1_2;
    ShowMessage(oHTTP->Get("https://www.google.es"));
}
__finally
{
    oHTTP->Free();
}
```

Wysyłanie żądania metodą **GET** do serwera HTTPS z użyciem **openssl 1.1** i **TLS 1.3**

```
TsgcHTTP1Client *oHTTP = new TsgcHTTP1Client();
try
{
    oHTTP->TLSOptions->OpenSSL_Options->APIVersion = sslAPI_1_1;
    oHTTP->TLSOptions->Version = tls1_3;
    ShowMessage(oHTTP->Get("https://www.google.es"));
}
__finally
{
    oHTTP->Free();
}
```



Należy zażądać metody **Asynchronous POST** i odczytać odpowiedź przy użyciu zdarzenia **OnAsyncResultEvent**.

```
void __fastcall OnAsyncExceptionEvent(TObject* Sender, const TsgcThread* aThread, const Exception* E)
{
    Log(E->Message);
}
void __fastcall OnAsyncResultEvent(TObject* Sender, const TsgcHTTPAsyncRequest* aRequest, const TIdHTTPResponse* aResponse)
{
    if (aResponse->ResponseCode == 200)
        Log("ok", aRequest->Response);
    else
        Log("error", aRequest->Response);
}
TsgcHTTP1Client* oHTTP = new TsgcHTTP1Client(NULL);
oHTTP->OnAsyncResult = OnAsyncResultEvent;
oHTTP->OnAsyncException = OnAsyncExceptionEvent;
TStringStream* oRequest = new TStringStream("body");
TStringStream* oResponse = new TStringStream("");
oHTTP->PostAsync("https://localhost/test", oRequest, oResponse);
```

Wysyłanie żądania GET do serwera HTTPs przy użyciu **SChannel dla Windows**.

```
TsgcHTTP1Client *oHTTP = new TsgcHTTP1Client();
try
{
    oHTTP->TLSOptions->IOHandler = iohSChannel;
    oHTTP->TLSOptions->Version = tls1_2;
    ShowMessage(oHTTP->Get("https://www.google.es"));
}
__finally
{
    oHTTP->Free();
}
```

Żądanie metody **SSE** w celu pobierania zdarzeń danych

```
TsgcHTTP1Client *oHTTP = new TsgcHTTP1Client();
oHTTP->OnSSEMessage = OnSSEMessageEvent;
oHTTP->GetSSE("https://www.yoursite.com/sse");

void OnSSEMessageEvent(TObject *Sender, const string aMessage, bool &Cancel)
{
    ShowMessage(aMessage);
}
```

## Zdarzenia

### OnSSEMessage

Zdarzenie jest wywoływane po odebraniu nowej wiadomości SSE.

### **OnSSLVerifyPeer**

Jeśli weryfikacja certyfikatu jest włączona, w tym zdarzeniu można zweryfikować certyfikat serwera i zdecydować, czy go zaakceptować.

### **OnSSLGetHandler**

To zdarzenie jest wywoływane przed utworzeniem procedury obsługi SSL. Można tu utworzyć własną procedurę obsługi SSL (musi ona dziedziczyć po `TIdServerIOHandlerSSLBase` lub `TIdIOHandlerSSLBase`) i ustawić wymagane właściwości.

### **OnSSLAfterCreateHandler**

Jeśli nie utworzono niestandardowego obiektu SSL, zostanie on domyślnie utworzony przy użyciu procedury obsługi OpenSSL. Można uzyskać dostęp do właściwości procedury obsługi SSL i modyfikować je w razie potrzeby.

### **OnAsyncResult**

Zdarzenie jest wywoływane po otrzymaniu odpowiedzi na metodę asynchroniczną (przy użyciu metod `GetAsync`, `PutAsync`...). Należy użyć parametru `Response`, aby poznać wynik żądania.

### **OnAsyncException**

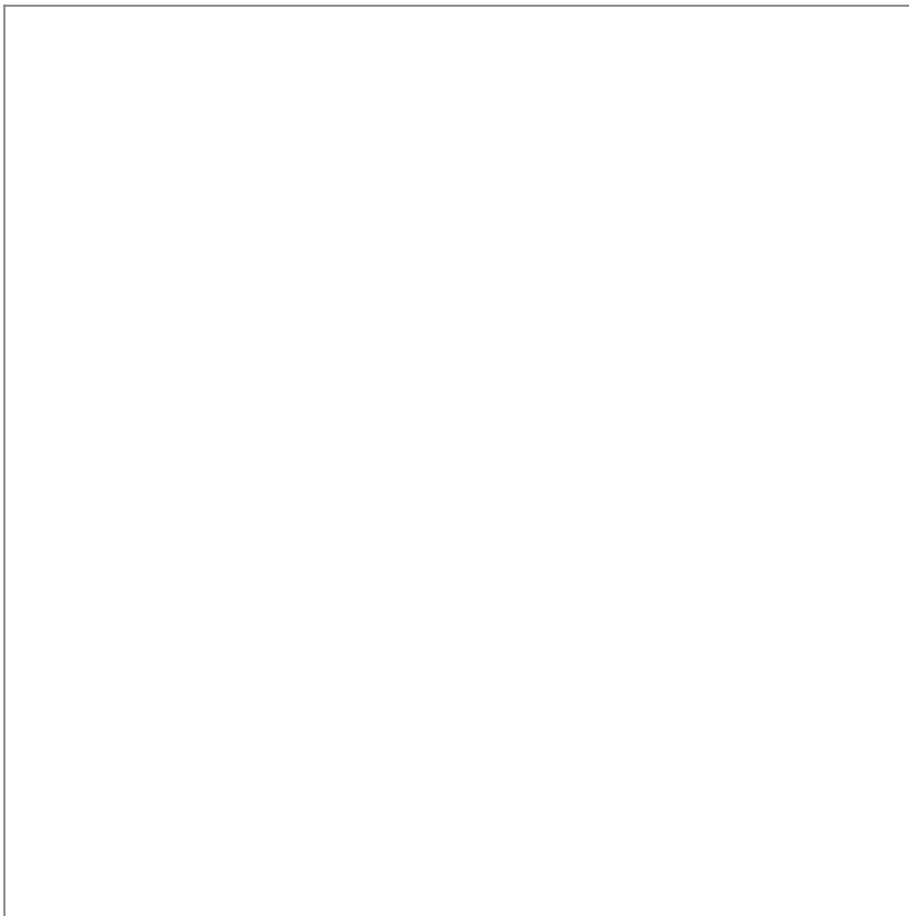
Jeśli podczas przetwarzania żądania asynchronicznego wystąpi błąd, to zdarzenie jest wywoływane z podniesionym wyjątkiem.

# HTTP | OAuth2

OAuth2 umożliwia aplikacjom trzecim uzyskanie ograniczonego dostępu do usługi HTTP, zarówno w imieniu właściciela zasobu, jak i na własny rachunek aplikacji. Dzięki OAuth2 dostawcy usług i aplikacje konsumenckie mogą współdziałać ze sobą w bezpieczny sposób.

W OAuth2 istnieją 4 role:

- **Resource Owner:** użytkownik.
- **Resource Server:** serwer przechowujący chronione zasoby i udostępniający do nich dostęp na podstawie tokenu dostępu.
- **Client:** zewnętrzna aplikacja ubiegająca się o uprawnienia.
- **Serwer autoryzacji:** wydaje token dostępu po uwierzytelnieniu użytkownika.



## Komponenty

- **TsgcHTTP\_OAuth2\_Client:** jest klientem z obsługą OAuth2, umożliwiającym łączenie się z serwerami OAuth2 w celu żądania uwierzytelnienia, np. przez Google, Facebook...
- **TsgcHTTP\_OAuth2\_Server:** implementacja serwerowa protokołu OAuth2, umożliwiająca ochronę zasobów serwera.
- **TsgcHTTP\_OAuth2\_Server\_Provider:** umożliwia implementację zewnętrznych dostawców OAuth2 (takich jak Azure AD, Google, Facebook...) na Twoim serwerze, dzięki czemu użytkownik może logować się przy użyciu danych uwierzytelniających Azure, Google, Facebook...

**Serwer i klient** OAuth2 obsługują **PKCE** (Proof Key for Code Exchange), które jest rozszerzeniem przepływu Authorization Code zapobiegającym atakom CSRF i wstrzykiwaniu kodu autoryzacyjnego (RFC 7636).

# TsgcHTTP\_OAuth2\_Client

Gdy klient potrzebuje nowego tokenu dostępu, automatycznie uruchamia serwer HTTP w celu przetworzenia odpowiedzi serwera autoryzacji.

## Wprowadzenie

Ten komponent umożliwia obsługę przepływu między klientem a innymi rolami. Zasadniczo po ustawieniu `Active := True` otwiera nową przeglądarkę internetową i żąda od użytkownika udzielenia autoryzacji. W przypadku powodzenia serwer autoryzacji wysyła token do aplikacji, który jest przetwarzany, a dzięki niemu klient może połączyć się z serwerem zasobów. Komponent uruchamia prosty serwer HTTP, który obsługuje odpowiedzi serwera autoryzacji, i używa klienta HTTP do żądania tokenów dostępu.

### GrantType

Klient obsługuje następujące typy autoryzacji:

**auth2Code:** Stosowany do uwierzytelniania i autoryzacji w większości typów aplikacji, w tym jednostronicowych, webowych i natywnych. Przepływ umożliwia aplikacjom bezpieczne uzyskiwanie `access_token`, które można wykorzystać do dostępu do chronionych zasobów, a także tokenów odświeżania do uzyskiwania kolejnych `access_token` oraz tokenów ID dla zalogowanego użytkownika.



**auth2CodePKCE:** jest to ten sam przepływ uwierzytelniania co `auth2Code` z włączonym PKCE. PKCE (Proof Key for Code Exchange) to rozszerzenie zabezpieczeń dla OAuth 2.0, zaprojektowane w celu zwiększenia bezpieczeństwa przepływów autoryzacji dla aplikacji natywnych i jednostronicowych. Ogranicza ryzyko ataków polegających na przechwyceniu, szczególnie w przypadku klientów publicznych, gdzie kod autoryzacji może być narażony na przechwycenie podczas transmisji. Opcja ta jest zazwyczaj stosowana w aplikacjach natywnych i mobilnych.

**auth2ClientCredentials:** Ten typ przydziału jest powszechnie stosowany w interakcjach serwer-serwer, które muszą działać w tle bez bezpośredniego udziału użytkownika. Aplikacje tego rodzaju są często określane jako demony lub konta usług.



**auth2DeviceCode:** Przyznanie autoryzacji urządzenia (RFC 8628) dla urządzeń z ograniczonymi możliwościami wprowadzania danych, takich jak telewizory Smart TV, konsole multimedialne i urządzenia IoT pozbawione przeglądarki lub mające ograniczone możliwości wejścia. Urządzenie wyświetla kod użytkownika i adres URI weryfikacyjny; użytkownik odwiedza ten adres na urządzeniu pomocniczym (telefon lub komputer) i wprowadza kod w celu autoryzacji.

## LocalServerOptions

Gdy klient potrzebuje nowego tokenu dostępu, automatycznie uruchamiany jest serwer HTTP w celu przetworzenia odpowiedzi z serwera autoryzacji. Ten serwer jest transparentny dla użytkownika i zazwyczaj działa na localhost. Domyślnie używa portu 8080, lecz można go zmienić w razie potrzeby.

- **IP:** adres IP serwera nasłuchującego, przykład: 127.0.0.1
- **Port:** domyślnie 8080. Gdy używany jest GrantType = auth2CodePKCE (dla aplikacji desktopowych i natywnych), można ustawić wartość portu na zero, a serwer wybierze losowy port.
- **RedirectURL:** (opcjonalne) umożliwia dostosowanie adresu URL przekierowania, przykład: http://localhost:8080/oauth/.
- **SSL:** należy włączyć tę właściwość, jeśli lokalny serwer działa na zabezpieczonym porcie (\*obsługiwane wyłącznie przez edycje Professional i Enterprise).
- **SSLOptions:** umożliwia dostosowanie właściwości SSL serwera (\*obsługiwane wyłącznie w wersjach Professional i Enterprise).
- **LogOptions:** umożliwia zapisywanie dziennika żądań/odpowiedzi odbieranych i wysyłanych przez wewnętrzny serwer HTTP (\*tylko dla edycji Professional i Enterprise).
  - **Enabled:** ustaw na True, aby włączyć zapis dziennika do pliku.
  - **FileName:** ustaw nazwę pliku, w którym będzie przechowywany dziennik.

## AuthorizationServerOptions

Tutaj należy ustawić adresy URL autoryzacji i tokenu dostępu, które zazwyczaj są podane w specyfikacji API. Scope to lista wszystkich zakresów żądanych przez klienta. Przykład:

- **AuthURL:** https://accounts.google.com/o/oauth2/auth
- **TokenURL:** https://accounts.google.com/o/oauth2/token
- **Scope:** https://mail.google.com/
- **RevocationURL:** adres URL punktu końcowego unieważniania tokenów (wymagany dla metody Revoke). Przykład: https://accounts.google.com/o/oauth2/revoke
- **IntrospectionURL:** adres URL punktu końcowego introspekcji tokenu (wymagany dla metody Introspect). Przykład: https://accounts.google.com/o/oauth2/introspect

## OAuth2Options

ClientId jest polem obowiązkowym informującym serwer o identyfikacji klienta. Sposób uzyskania ClientId należy sprawdzić w specyfikacji API. To samo dotyczy client secret.

Czasami serwer wymaga nazwy użytkownika i hasła do połączenia za pomocą uwierzytelniania Basic. W takim przypadku należy skonfigurować je w polach Username/Password. Przykład:

- **GrantType:** Typ przepływu autoryzacji
  - **auth2Code:** zaufane aplikacje, takie jak serwer webowy.
  - **auth2CodePKCE:** niezaufane aplikacje natywne lub mobilne.
  - **auth2ClientCredentials:** automatyczne aplikacje bez interakcji z użytkownikiem.
  - **auth2ResourceOwnerPassword:** umożliwia aplikacji zalogowanie użytkownika przez bezpośrednią obsługę jego hasła.
  - **auth2DeviceCode:** Device Authorization Grant (RFC 8628) dla urządzeń z ograniczonymi możliwościami wprowadzania danych, bez przeglądarki lub z ograniczonym interfejsem wejścia.
- **ClientId:** 180803918307-eqjtm20gqfhcs6gjkbbrreng022mqqc.apps.googleusercontent.com
- **ClientSecret:** \_by1iYYrvVHxC2Z8TbtNEYJN
- **Username:**
- **Hasło:**

## HTTPClientOptions

W tym miejscu można dostosować opcje klienta używane podczas połączenia z serwerem HTTP w celu żądania nowego tokenu.

**TLSOptions:** jeśli TLS jest włączone, tutaj można dostosować niektóre właściwości TLS.

**ALPNProtocols:** lista protokołów ALPN, które zostaną wysłane do serwera.

**RootCertFile:** ścieżka do pliku certyfikatu głównego.

**CertFile:** ścieżka do pliku certyfikatu.

**KeyFile:** ścieżka do pliku klucza certyfikatu.

**Password:** jeśli certyfikat jest zabezpieczony hasłem, należy je tutaj podać.

**VerifyCertificate:** jeśli certyfikat musi być weryfikowany, należy włączyć tę właściwość.

**VerifyDepth:** właściwość całkowitoliczbowa określająca maksymalną liczbę ogniw dozwolonych podczas weryfikacji certyfikatu X.509.

**Version:** domyślnie używa TLS 1.0; jeśli serwer wymaga wyższej wersji TLS, można ją tutaj wybrać.

**IOHandler:** wybierz bibliotekę, której chcesz użyć do połączenia przez TLS.

**iohOpenSSL:** korzysta z biblioteki OpenSSL i jest domyślną opcją dla komponentów Indy. Wymaga wdrożenia bibliotek OpenSSL dla win32/win64.

**iohSChannel:** używa kanału Secure Channel — protokołu bezpieczeństwa zaimplementowanego przez Microsoft dla systemu Windows; nie wymaga wdrażania bibliotek openssl. Działa wyłącznie w systemie Windows 32/64 bity.

**OpenSSL\_Options:** umożliwia zdefiniowanie, które API OpenSSL będzie używane.

**APIVersion:** pozwala określić, które API OpenSSL będzie używane.

**oslAPI\_1\_0:** używa API 1.0 OpenSSL; jest to najnowsza wersja obsługiwana przez Indy

**oslAPI\_1\_1:** używa interfejsu API 1.1 OpenSSL, wymaga niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**oslAPI\_3\_0:** korzysta z API OpenSSL 3.0, wymaga naszej niestandardowej biblioteki Indy i umożliwia używanie bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek openssl

**oslpNone:** jest to wartość domyślna; biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanej ścieżce.

**oslpDefaultFolder:** automatycznie ustawia ścieżkę openssl, w której powinny znajdować się biblioteki dla wszystkich osobowości IDE.

**oslpCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.

**LibPathCustom:** gdy LibPath = oslpCustomFolder, należy podać pełną ścieżkę do folderu, w którym znajdują się biblioteki openssl.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**oslsSymLinksDefault:** domyślnie jest włączone, z wyjątkiem platformy OSX64 (po MacOS Monterey próba załadowania biblioteki bez wersji kończy się błędem).

**oslsSymLinksLoadFirst:** Załaduj dowiązania symboliczne przed próbą załadowania bibliotek wersji.

**oslsSymLinksLoad:** Ładuj dowiązania symboliczne po próbie załadowania bibliotek wersji.

**oslsSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

**SChannel\_Options:** umożliwia korzystanie z certyfikatu z magazynu certyfikatów systemu Windows.

**CertHash:** to skrót certyfikatu. Skrót certyfikatu można znaleźć, uruchamiając polecenie dir w programie PowerShell.

**CertStoreName:** nazwa magazynu, w którym przechowywany jest certyfikat. Należy wybrać jedną z poniższych opcji:

**scsnMY** (wartość domyślna)

**scsnCA**

**scsnRoot**

**scsnTrust**



**CertStorePath:** ścieżka magazynu, w którym przechowywany jest certyfikat. Wybierz jedną z poniższych opcji:

**scspStoreCurrentUser** (domyślne)

**scspStoreLocalMachine**

**LogOptions:** jeśli podana jest nazwa pliku, zapisuje dziennik żądań/odpowiedzi HTTP klienta HTTP

## Dokumentacja

- Właściwości — 6 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 19 zdarzeń do obsługi.

## Klient OAuth2 dla aplikacji internetowych

Gdy klient OAuth2 musi uzyskać autoryzowany token łącząc się z aplikacją internetową, lokalny serwer (używany do uzyskania autoryzowanego tokenu) musi być skonfigurowany z parametrami aplikacji internetowej. Należy ustawić następujące właściwości **LocalServerOptions**:

- **IP:** Adres IP skonfigurowany wcześniej w konfiguracji OAuth2.
- **Port:** Port nasłuchiwania skonfigurowany wcześniej w konfiguracji OAuth2.
- **SSL:** jeśli serwer korzysta z bezpiecznego połączenia, należy włączyć tę opcję.

## Klient OAuth2 dla aplikacji desktopowych

Gdy klient OAuth2 jest aplikacją desktopową lub natywną, lokalny serwer (używany do uzyskania autoryzowanego tokenu) może nasłuchiwać na lokalnym adresie IP (przykład: 127.0.0.1), a port może być wybierany losowo. Dlatego **LocalServerOptions** powinny być skonfigurowane w następujący sposób

- **IP:** 127.0.0.1
- **Port:** po ustawieniu na zero serwer automatycznie wybiera losowy port.

Podczas tworzenia klienta OAuth2 dla aplikacji desktopowych lub natywnych należy ustawić **OAuth2Options.GrantType** na **auth2CodePKCE**, aby zapewnić dodatkowe bezpieczeństwo.

# OAuth2 | TsgcHTTP\_OAuth2\_Client\_Google

Ten komponent umożliwia łatwe logowanie się za pomocą konta Google.

## Konfiguracja

Moduł wymaga najpierw **skonfigurowania aplikacji OAuth2** na koncie Google. Po skonfigurowaniu wystarczy dodać kilka linii kodu do aplikacji, aby umożliwić użytkownikom logowanie za pomocą dowolnego konta Google.

**Lokalny serwer** służący do odczytu odpowiedzi od Google domyślnie nasłuchuje na **adresie IP 127.0.0.1 i porcie 0** (losowy port). W aplikacji Google należy zatem skonfigurować adres URL wywołania zwrotnego (CallBack URL). Adres IP i port można oczywiście zmienić.

Po skonfigurowaniu aplikacji OAuth2 na koncie Google należy utworzyć wystąpienie **TsgcHTTP\_OAuth2\_Client\_Google** i wywołać metodę **Authenticate** przekazując jako parametry **Client\_Id** oraz **Client\_Secret**. Ta **metoda oczekuje** (domyślnie do 60 sekund) na pomyślne zalogowanie się użytkownika. Zwraca obiekt umożliwiający sprawdzenie, czy użytkownik się uwierzytelnił, oraz odczytanie jego nazwy, identyfikatora i innych danych z profilu.

## Przykład

```
void GoogleSignIn()
{
    TsgcHTTP_OAuth2_Client_Google *oClient = new TsgcHTTP_OAuth2_Client_Google.Create(this);
    Try
    {
        TsgcOAuth2_Google_Data *oData = oClient.Authenticate("client_id", "client_secret");
        if oData->Authenticated() then
            ShowMessage(oData->UserProfile->_Name);
    }
    __Finally
        oClient->Free();
}
```

# TsgcHTTP\_OAuth2\_Client\_Microsoft

Ten komponent umożliwia łatwe logowanie przy użyciu konta Microsoft.

## Konfiguracja

Moduł wymaga najpierw **skonfigurowania aplikacji OAuth2** na koncie Microsoft. Po jej skonfigurowaniu wystarczy dodać kilka linii kodu w aplikacji, aby umożliwić użytkownikom logowanie za pomocą dowolnego konta Microsoft.

**Serwer lokalny** służący do odczytu odpowiedzi od Microsoft domyślnie nasłuchuje na **adresie IP 127.0.0.1 i porcie 8080** z obsługą SSL. Dlatego w aplikacji Microsoft należy skonfigurować adres URL wywołania zwrotnego jako **https://localhost:8080** (Microsoft dopuszcza tylko localhost jako lokalny adres IP). Adres IP i port można oczywiście zmienić.

Po skonfigurowaniu aplikacji OAuth2 na koncie Microsoft należy utworzyć instancję **TsgcHTTP\_OAuth2\_Client\_Microsoft** i wywołać metodę **Authenticate** przekazując jako parametry **TenantId**, **Client\_Id** i **Client\_Secret**. Ta **metoda czeka** (domyślnie do 60 sekund) na pomyślne zalogowanie się użytkownika. Zwraca obiekt pozwalający sprawdzić, czy użytkownik uwierzyłtnił się, oraz zawierający takie dane jak Name, Id i inne informacje z profilu użytkownika.

## Przykład

```
void GoogleSignIn()
{
    TsgcHTTPComponentClient_OAuth2_Microsoft *oClient = new TsgcHTTPComponentClient_OAuth2_Microsoft.Create(this);
    Try
    {
        TsgcOAuth2_Microsoft_Data *oData = oClient.Authenticate("tenant_id", "client_id", "client_secret");
        if oData->Authenticated() then
            ShowMessage(oData->UserProfile->DisplayName);
    }
    __Finally
        oClient->Free();
}
```

# OAuth2 | Authorization Code Grant (RFC 6749)

## Przegląd

Grant kodu autoryzacyjnego jest najpowszechniejszym przepływem OAuth2. Jest przeznaczony dla aplikacji webowych, desktopowych i mobilnych, w których klient może bezpiecznie przechowywać tajemnicę klienta. Przepływ polega na przekierowaniu użytkownika do serwera autoryzacji, gdzie użytkownik uwierzytelnia się i udziela uprawnień. Serwer zwraca następnie kod autoryzacyjny pod adres URI przekierowania, a klient wymienia ten kod na token dostępu.

## Przepływ

1. Klient przekierowuje użytkownika do adresu **AuthURL** z parametrami `client_id`, `redirect_uri`, `response_type=code`, `scope` i `state`.
2. Użytkownik uwierzytelnia się na serwerze autoryzacji i udziela uprawnień klientowi.
3. Serwer autoryzacji przekierowuje użytkownika z powrotem do `redirect URI` z parametrem `code`.
4. Klient wysyła metodą POST kod autoryzacji do **TokenURL** wraz z `client_id`, `client_secret` i `redirect_uri` w celu wymiany na token dostępu.
5. Serwer zwraca `access_token`, `token_type`, `expires_in` i opcjonalnie `refresh_token`.

## Konfiguracja

| Właściwość                                       | Opis                                                                                                        |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.GrantType</code>             | Ustaw na <b>auth2Code</b> .                                                                                 |
| <code>OAuth2Options.ClientId</code>              | Identyfikator klienta wydany przez serwer autoryzacji.                                                      |
| <code>OAuth2Options.ClientSecret</code>          | Sekret klienta wydany przez serwer autoryzacji.                                                             |
| <code>AuthorizationServerOptions.AuthURL</code>  | Adres URL punktu końcowego autoryzacji, do którego użytkownik jest przekierowywany w celu uwierzytelnienia. |
| <code>AuthorizationServerOptions.TokenURL</code> | Adres URL punktu końcowego tokenu, w którym kod autoryzacji jest wymieniany na token dostępu.               |
| <code>AuthorizationServerOptions.Scope</code>    | Zakres żądania dostępu (np. <code>openid</code> , <code>profile</code> , <code>email</code> ).              |
| <code>LocalServerOptions.IP</code>               | Adres IP lokalnego serwera przekierowań (np. <code>127.0.0.1</code> ).                                      |
| <code>LocalServerOptions.Port</code>             | Port lokalnego serwera przekierowania (np. <code>8080</code> ).                                             |

## Zdarzenia

| Zdarzenie                          | Opis                                                                                                                  |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <code>OnBeforeAuthorizeCode</code> | Wywoływane przed przekierowaniem klienta do punktu końcowego autoryzacji. Umożliwia dostosowanie żądania autoryzacji. |
| <code>OnAfterAuthorizeCode</code>  | Wywoływane po otrzymaniu kodu autoryzacji z serwera.                                                                  |
| <code>OnErrorAuthorizeCode</code>  | Wywoływane w przypadku błędu podczas kroku z kodem autoryzacji.                                                       |
| <code>OnBeforeAccessToken</code>   | Wywoływane przed wymianą kodu autoryzacyjnego przez klienta na token dostępu.                                         |
| <code>OnAfterAccessToken</code>    | Wywoływane po otrzymaniu tokenu dostępu. Zdarzenie służy do pobrania wartości tokenu.                                 |

OnErrorAccess-  
Token

Wywoływane, gdy podczas kroku wymiany tokenów wystąpi błąd.

## Przykład

```
OAuth2->OAuth2Options->GrantType = auth2Code;  
OAuth2->OAuth2Options->ClientId = "your-client-id";  
OAuth2->OAuth2Options->ClientSecret = "your-client-secret";  
OAuth2->AuthorizationServerOptions->AuthURL = "https://provider.com/oauth2/authorize";  
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";  
OAuth2->AuthorizationServerOptions->Scope->Text = "openid profile";  
OAuth2->LocalServerOptions->IP = "127.0.0.1";  
OAuth2->LocalServerOptions->Port = 8080;  
OAuth2->Start();
```

## Odświeżanie tokenu

Gdy token dostępu wygaśnie, można użyć tokenu odświeżania, aby uzyskać nowy bez konieczności interakcji użytkownika. Należy wywołać metodę Refresh z wartością tokenu odświeżania:

Zdarzenie OnAfterAccessToken zostanie wywołane z nowym tokenem dostępu. Jeśli odświeżenie nie powiedzie się, zostanie wywołane zdarzenie OnErrorAccessToken.

## Kiedy stosować

Należy użyć przyznania Authorization Code w następujących przypadkach:

- Aplikacje internetowe z zapleczem po stronie serwera, które mogą bezpiecznie przechowywać klucz tajny klienta.
- Aplikacje renderowane po stronie serwera (np. ASP.NET, PHP, Java).
- Aplikacje desktopowe, w których tajny klucz klienta może być w rozsądny sposób chroniony.

W przypadku klientów publicznych (aplikacje natywne, SPA), w których nie można bezpiecznie przechowywać tajnego klucza klienta, należy rozważyć użycie [kodu autoryzacyjnego z PKCE](#).

# OAuth2 | Kod autoryzacji z PKCE (RFC 7636)

## Przegląd

Przepływ Authorization Code z PKCE (Proof Key for Code Exchange) rozszerza standardowy przepływ Authorization Code o dodatkową warstwę zabezpieczeń. Jest przeznaczony dla aplikacji natywnych, mobilnych i jednostronicowych (SPA), w których nie można bezpiecznie przechowywać sekretu klienta. Zamiast polegać na sekrecie klienta, klient generuje kryptograficzny weryfikator kodu i wyzwanie, które potwierdzają, że podmiot wymieniający kod autoryzacji jest tym samym podmiotem, który zainicjował przepływ.

## Jak działa PKCE

1. Klient generuje losowy **code\_verifier** (32 bajty kryptograficznie losowych danych, zakodowane w Base64URL).
2. Klient oblicza **code\_challenge** jako skrót SHA-256 wartości code\_verifier zakodowany w Base64URL.
3. Klient wysyła code\_challenge i code\_challenge\_method=S256 wraz z żądaniem autoryzacji.
4. Po autoryzacji przez użytkownika klient przesyła oryginalny parametr code\_verifier wraz z żądaniem wymiany tokenu.
5. Serwer autoryzacji weryfikuje, czy  $\text{SHA256}(\text{code\_verifier}) == \text{code\_challenge}$  przed wydaniem tokenu.

Zapobiega to atakom przechwycenia kodu autoryzacji, ponieważ atakujący, który przechwyci kod, nie może go wymienić bez oryginalnego code\_verifier.

## Konfiguracja

| Właściwość                          | Opis                                                                                                        |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------|
| OAuth2Options.GrantType             | Ustaw na <b>auth2CodePKCE</b> .                                                                             |
| OAuth2Options.ClientId              | Identyfikator klienta wydany przez serwer autoryzacji.                                                      |
| OAuth2Options.ClientSecret          | Opcjonalne. Niektórzy dostawcy wymagają tego nawet w połączeniu z PKCE; inni nie.                           |
| AuthorizationServerOptions.AuthURL  | Adres URL punktu końcowego autoryzacji, do którego użytkownik jest przekierowywany w celu uwierzytelnienia. |
| AuthorizationServerOptions.TokenURL | Adres URL punktu końcowego tokenu, w którym kod autoryzacji jest wymieniany na token dostępu.               |
| AuthorizationServerOptions.Scope    | Zakres żądania dostępu.                                                                                     |
| LocalServerOptions.IP               | Adres IP lokalnego serwera przekierowań (np. 127.0.0.1).                                                    |
| LocalServerOptions.Port             | Port lokalnego serwera przekierowującego. Ustaw wartość <b>0</b> , aby losowo wybrać dostępny port.         |

## Przykład

```
OAuth2->OAuth2Options->GrantType = auth2CodePKCE;
OAuth2->OAuth2Options->ClientId = "your-client-id";
OAuth2->AuthorizationServerOptions->AuthURL = "https://provider.com/oauth2/authorize";
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";
OAuth2->AuthorizationServerOptions->Scope->Text = "openid profile";
```



```
OAuth2->LocalServerOptions->IP = "127.0.0.1";  
OAuth2->LocalServerOptions->Port = 0;  
OAuth2->Start();
```

## Losowy port

Gdy właściwość `LocalServerOptions.Port` jest ustawiona na **0**, komponent automatycznie wybiera losowy dostępny port. Jest to zalecane w aplikacjach desktopowych i mobilnych, ponieważ pozwala uniknąć konfliktów portów. Wybrany port jest uwzględniany w identyfikatorze URI przekierowania wysłanym do serwera autoryzacji.

## Odświeżanie tokenu

Przepływy PKCE zazwyczaj zwracają token odświeżania. Należy użyć metody `Refresh`, aby uzyskać nowy token dostępu:

## Kiedy stosować

Należy użyć nadania `Authorization Code` z PKCE w następujących przypadkach:

- Aplikacje desktopowe (natywne dla systemów Windows, macOS, Linux).
- Aplikacje mobilne (iOS, Android).
- Jednostronicowe aplikacje (SPA) działające w przeglądarce.
- Dowolny klient publiczny, który nie może bezpiecznie przechowywać tajnego klucza klienta.
- Jako bezpieczniejsza alternatywa dla standardowego grantu [Authorization Code](#), nawet dla klientów poufnych.

# OAuth2 | Client Credentials Grant (RFC 6749)

## Przegląd

Grant Client Credentials jest używany do uwierzytelniania maszynowego (M2M), w którym nie jest wymagana interakcja użytkownika. Klient uwierzytelnia się bezpośrednio na serwerze autoryzacyjnym przy użyciu własnych poświadczeń (`client_id` i `client_secret`) i otrzymuje token dostępu. Nie ma etapu kodu autoryzacyjnego ani logowania użytkownika.

Ten typ grantu jest odpowiedni, gdy klient działa w swoim własnym imieniu, a nie w imieniu użytkownika.

## Przepływ

1. Klient przesyła metodą POST wartości `client_id`, `client_secret`, `grant_type=client_credentials` oraz `scope` do punktu końcowego tokenu.
2. Serwer autoryzacji weryfikuje poświadczenia klienta.
3. Serwer zwraca wartości `access_token`, `token_type` i `expires_in`.

Należy pamiętać, że tokeny odświeżające zazwyczaj nie są wydawane przy tym typie grantu, ponieważ klient może po prostu zażądać nowego tokenu dostępu przy użyciu własnych poświadczeń.

## Konfiguracja

| Właściwość                                       | Opis                                                                                                                                    |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.GrantType</code>             | Ustaw na <b>auth2ClientCredentials</b> .                                                                                                |
| <code>OAuth2Options.ClientId</code>              | Identyfikator klienta wydany przez serwer autoryzacji.                                                                                  |
| <code>OAuth2Options.ClientSecret</code>          | Sekret klienta wydany przez serwer autoryzacji.                                                                                         |
| <code>AuthorizationServerOptions.TokenURL</code> | Adres URL punktu końcowego tokenu. Parametr <code>AuthURL</code> nie jest wymagany, ponieważ nie istnieje krok autoryzacji użytkownika. |
| <code>AuthorizationServerOptions.Scope</code>    | Zakres żądania dostępu.                                                                                                                 |

## Przykład

```
OAuth2->OAuth2Options->GrantType = auth2ClientCredentials;
OAuth2->OAuth2Options->ClientId = "your-client-id";
OAuth2->OAuth2Options->ClientSecret = "your-client-secret";
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";
OAuth2->AuthorizationServerOptions->Scope->Text = "api.read api.write";
OAuth2->Start();
```

## Kiedy stosować

Typ przyznania dostępu Client Credentials należy stosować w następujących przypadkach:

- Usługi backendowe i demony działające bez interakcji użytkownika.
- Komunikacja API serwer-serwer.

- Zaplanowane zadania, procesy wsadowe i zadania cron.
- Uwierzytelnianie między mikroserwisami.
- Każdy scenariusz, w którym aplikacja działa we własnym imieniu, a nie w imieniu użytkownika.

# OAuth2 | Grant poświadczeń hasła właściciela zasobu (RFC 6749)

## Przegląd

Uprawnienie Resource Owner Password Credentials (ROPC) umożliwia klientowi bezpośrednie zebranie nazwy użytkownika i hasła od użytkownika oraz wymianę ich na token dostępu. Dane uwierzytelniające użytkownika są wysyłane do punktu końcowego tokenu, a serwer zwraca token dostępu, jeśli są prawidłowe.

Ten typ grantu powinien być używany tylko wtedy, gdy istnieje wysoki poziom zaufania między właścicielem zasobu a klientem, na przykład gdy klient jest aplikacją pierwszej strony opracowaną przez tę samą organizację, która obsługuje serwer autoryzacji.

## Ostrzeżenie bezpieczeństwa

**Ten typ nadania jest uznawany za mniej bezpieczny niż inne przepływy OAuth2.** Ujawnia bezpośrednio poświadczenia użytkownika aplikacji klienckiej, co narusza zasadę delegowanej autoryzacji. Specyfikacja OAuth 2.1 uznała ten typ nadania za przestarzały. W miarę możliwości należy zamiast niego stosować [Authorization Code z PKCE](#).

## Przebieg

1. Użytkownik podaje nazwę użytkownika i hasło bezpośrednio do aplikacji klienckiej.
2. Klient wysyła metodą POST pola username, password, client\_id, client\_secret, grant\_type=password i scope do punktu końcowego tokenu.
3. Serwer autoryzacji weryfikuje poświadczenia.
4. Serwer zwraca access\_token, token\_type, expires\_in i opcjonalnie refresh\_token.

## Konfiguracja

| Właściwość                          | Opis                                                                                                                                |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| OAuth2Options.GrantType             | Ustaw na <b>auth2ResourceOwnerPassword</b> .                                                                                        |
| OAuth2Options.ClientId              | Identyfikator klienta wydany przez serwer autoryzacji.                                                                              |
| OAuth2Options.ClientSecret          | Sekret klienta wydany przez serwer autoryzacji.                                                                                     |
| OAuth2Options.UserName              | Nazwa użytkownika właściciela zasobu.                                                                                               |
| OAuth2Options.Password              | Hasło właściciela zasobu.                                                                                                           |
| AuthorizationServerOptions.TokenURL | Adres URL punktu końcowego tokena. Parametr AuthURL nie jest wymagany, ponieważ poświadczenia użytkownika są podawane bezpośrednio. |
| AuthorizationServerOptions.Scope    | Zakres żądania dostępu.                                                                                                             |

## Przykład

```
OAuth2->OAuth2Options->GrantType = auth2ResourceOwnerPassword;
OAuth2->OAuth2Options->ClientId = "your-client-id";
OAuth2->OAuth2Options->ClientSecret = "your-client-secret";
OAuth2->OAuth2Options->UserName = "user@example.com";
OAuth2->OAuth2Options->Password = "user-password";
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";
```

```
OAuth2->AuthorizationServerOptions->Scope->Text = "openid profile";  
OAuth2->Start();
```

## Odświeżanie tokenu

Jeśli serwer zwróci token odświeżania, można go użyć do uzyskania nowego tokenu dostępu bez ponownego wprowadzania poświadczeń użytkownika:

## Kiedy stosować

Typ przyznania Resource Owner Password Credentials należy stosować wyłącznie w przypadku:

- Starsze aplikacje, których nie można przenieść na bezpieczniejszy przepływ.
- Aplikacje własne, w których kontrolowane są zarówno klient, jak i serwer autoryzacji.
- Scenariusze migracji, w których następuje przejście z bezpośredniego uwierzytelniania nazwą użytkownika i hasłem do OAuth2.

We wszystkich innych przypadkach należy użyć [Kodu autoryzacji z PKCE](#) lub [Kodu autoryzacji](#).

# OAuth2 | Device Authorization Grant (RFC 8628)

## Przegląd

Grant Device Authorization (znany również jako przepływ Device Code) jest przeznaczony dla urządzeń z ograniczonymi możliwościami wprowadzania danych, które nie mogą łatwo wyświetlić przeglądarki ani przyjmować danych z klawiatury. Przykłady obejmują inteligentne telewizory, urządzenia IoT, odtwarzacze multimedialne, narzędzia CLI i kioski. Urządzenie wyświetla krótki kod, który użytkownik wprowadza na innym urządzeniu (np. telefonie lub komputerze), aby ukończyć autoryzację.

## Przepływ

1. Klient wysyła metodą POST `client_id` i `scope` do **DeviceAuthorizationURL**.
2. Serwer zwraca `device_code`, `user_code`, `verification_uri`, `expires_in` oraz `interval`.
3. Klient wywołuje zdarzenie **OnDeviceCode** z kodem użytkownika i identyfikatorem URI weryfikacji. Aplikacja wyświetla te dane użytkownikowi (np. „Przejdź na <https://provider.com/device> i wprowadź kod: ABCD-1234”).
4. Użytkownik przechodzi do identyfikatora URI weryfikacyjnego na innym urządzeniu i wprowadza kod użytkownika.
5. Użytkownik uwierzytelnia się i udziela uprawnień na urządzeniu pomocniczym.
6. W tym czasie klient automatycznie odpytuje punkt końcowy tokenu co `interval` sekund, używając kodu urządzenia.
7. Gdy użytkownik zakończy autoryzację, następne odpytanie zwraca `access_token`.
8. Jeśli kod urządzenia wygaśnie przed autoryzacją przez użytkownika, wywoływane jest zdarzenie **OnDeviceCodeExpired**.

## Konfiguracja

| Właściwość                                                     | Opis                                                                                   |
|----------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <code>OAuth2Options.GrantType</code>                           | Ustaw wartość <b>auth2DeviceCode</b> .                                                 |
| <code>OAuth2Options.ClientId</code>                            | Identyfikator klienta wydany przez serwer autoryzacji.                                 |
| <code>OAuth2Options.ClientSecret</code>                        | Opcjonalne. Niektórzy dostawcy tego wymagają, inni nie.                                |
| <code>AuthorizationServerOptions.TokenURL</code>               | Adres URL punktu końcowego tokenu używany do odpytywania.                              |
| <code>AuthorizationServerOptions.DeviceAuthorizationURL</code> | Punkt końcowy autoryzacji urządzenia, pod który klient wysyła żądanie kodu urządzenia. |
| <code>AuthorizationServerOptions.Scope</code>                  | Zakres żądania dostępu.                                                                |

## Zdarzenia

| Zdarzenie                        | Opis                                                                                                                                                          |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OnDeviceCode</code>        | Wywoływane po otrzymaniu kodu urządzenia i kodu użytkownika z serwera. Należy wyświetlić kod użytkownika i identyfikator URI weryfikacji użytkownikowi.       |
| <code>OnDeviceCodeExpired</code> | Wywoływane, gdy kod urządzenia wygasa przed ukończeniem autoryzacji przez użytkownika. Aplikacja powinna poprosić użytkownika o ponowne uruchomienie procesu. |

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| OnAfterAccessToken | Wywoływane, gdy użytkownik ukończy autoryzację i zostanie odebrany token dostępu. |
| OnErrorAccessToken | Wywoływane po wystąpieniu nieodwracalnego błędu podczas przepływu urządzenia.     |

## Przykład

### Obsługa zdarzenia kodu urządzenia

```
OAuth2->OAuth2Options->GrantType = auth2DeviceCode;
OAuth2->OAuth2Options->ClientId = "your-client-id";
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";
OAuth2->AuthorizationServerOptions->DeviceAuthorizationURL = "https://provider.com/oauth2/device/code";
OAuth2->AuthorizationServerOptions->Scope->Text = "openid profile";
OAuth2->Start();
```

## Zachowanie przy odpytywaniu

Po wywołaniu Start komponent automatycznie odpytuje punkt końcowy tokenu w interwale określonym przez serwer (zazwyczaj 5 sekund). Jeśli serwer odpowie błędem slow\_down, komponent zwiększa interwał odpytywania o 5 sekund. Odpytywanie trwa do momentu wystąpienia jednego z poniższych zdarzeń:

- Użytkownik kończy autoryzację i token jest zwracany.
- Kod urządzenia wygasa (wywoływane jest zdarzenie OnDeviceCodeExpired).
- Wystąpił nieodwracalny błąd (wywoływane jest OnErrorAccessToken).

## Kiedy stosować

Należy użyć nadania Device Authorization w następujących przypadkach:

- Inteligentne telewizory i urządzenia do strumieniowania.
- Narzędzia interfejsu wiersza poleceń (CLI).
- Urządzenia IoT o ograniczonych możliwościach wejściowych.
- Kioski i cyfrowe systemy informacji wizualnej.
- Dowolne urządzenie, na którym wpisanie pełnego adresu URL lub danych uwierzytelniających jest niepraktyczne.

# OAuth2 | DPoP — Demonstrating Proof of Possession (RFC 9449)

## Przegląd

DPoP (Demonstrating Proof of Possession) to mechanizm bezpieczeństwa wiążący tokeny dostępu z konkretnym klientem przez wymaganie dowodu posiadania klucza prywatnego. W przeciwieństwie do tokenów bearer, które może użyć każdy, kto je uzyska, tokeny powiązane z DPoP są bezużyteczne bez odpowiedniego klucza prywatnego. Zapewnia to ochronę przed kradzieżą tokenów i atakami typu replay.

DPoP nie jest samodzielnym typem przydziału, lecz mechanizmem zwiększającym bezpieczeństwo, który można stosować z dowolnym typem przydziału OAuth2 (kod autoryzacji, poświadczenia klienta itp.).

## Jak to działa

1. Klient generuje asymetryczną parę kluczy (ES256 lub RS256).
2. Klient tworzy **token JWT DPoP** o następującej strukturze:
  - Nagłówek: typ: dpop+jwt, alg: ES256, jwk: {public key}
  - Ładunek: htm (metoda HTTP), htu (URI HTTP), iat (data wydania), jti (unikalny identyfikator)
3. Klient wysyła dowód DPoP jako nagłówek HTTP (DPoP: <proof>) wraz z żądaniem tokenu.
4. Serwer autoryzacji wiąże wydany token z odciskiem JWK klucza publicznego klienta.
5. Serwer zwraca token\_type: DPoP zamiast token\_type: Bearer.
6. Przy kolejnych wywołaniach API klient wysyła zarówno nagłówek Authorization: DPoP <token>, jak i nowy nagłówek DPoP: <proof> specyficzny dla danego żądania.

## Generowanie kluczy

Parę kluczy ES256 można wygenerować bezpośrednio w Delphi przy użyciu metody **GenerateDPoP-KeyPair**. Metoda ta wewnętrznie korzysta z OpenSSL do utworzenia pary kluczy i automatycznie ustawia zarówno PrivateKey, jak i PublicKeyJWK:

```
OAuth2->DPoPOptions->Enabled = true;
OAuth2->DPoPOptions->Algorithm = dpopES256;
OAuth2->GenerateDPoPKeyPair(); // Generates PrivateKey + PublicKeyJWK automatically
OAuth2->Start();
```

## PublicKeyToJWK — konwersja dowolnego klucza lub certyfikatu do formatu JWK

Jeśli posiadasz już klucz publiczny, klucz prywatny lub certyfikat X.509 (format PEM), możesz go przekonwertować do formatu JWK za pomocą funkcji klasy **PublicKeyToJWK**. Akceptuje ona każde z tych danych wejściowych i zwraca poprawny JSON JWK:

### Dane wejściowe

### Nagłówek PEM

|                            |                                 |
|----------------------------|---------------------------------|
| Klucz publiczny EC lub RSA | -----BEGIN PUBLIC KEY-----      |
| Klucz prywatny EC          | -----BEGIN EC PRIVATE KEY-----  |
| Klucz prywatny RSA         | -----BEGIN RSA PRIVATE KEY----- |
| Certyfikat X.509           | -----BEGIN CERTIFICATE-----     |



## Z pliku klucza PEM

```
// From a public key file
OAuth2->DPoPOptions->PublicKeyJWK =
    OAuth2->PublicKeyToJWK(
        TFile::ReadAllText("public_key.pem"));

// From a private key file (extracts the public part)
OAuth2->DPoPOptions->PublicKeyJWK =
    OAuth2->PublicKeyToJWK(
        TFile::ReadAllText("private_key.pem"));
```

## Z certyfikatu karty inteligentnej

Karty inteligentne i moduły HSM chronią klucz prywatny — nie można go odczytać ani wyeksportować. Jednak **certyfikat** (zawierający klucz publiczny) zawsze można wyeksportować. Wyeksportować certyfikat jako PEM i przekazać go do `PublicKeyToJWK`:

```
// Export the certificate from the smart card
UnicodeString CertPEM = MySmartCard->ExportCertificatePEM();

// Convert the certificate to JWK
OAuth2->DPoPOptions->PublicKeyJWK =
    OAuth2->PublicKeyToJWK(CertPEM);

// No PrivateKey needed - use OnDPoPSign for signing via the card
OAuth2->DPoPOptions->Enabled = true;
OAuth2->OnDPoPSign = MyDPoPSignHandler;
```

To podejście działa z każdą kartą lub modulem HSM, który może wyeksportować swój certyfikat: tokenami PKCS#11, kartami inteligentnymi Windows CNG, kluczami bezpieczeństwa USB (YubiKey, Feitian) oraz chmurowymi modułami HSM (Azure Key Vault, AWS CloudHSM).

## Korzystanie z narzędzi wiersza poleceń OpenSSL

Można również wygenerować parę kluczy zewnętrznie:

## Opcje DPoP

| Właściwość                            | Opis                                                                                 |
|---------------------------------------|--------------------------------------------------------------------------------------|
| <code>DPoPOptions.Enabled</code>      | Ustaw na <b>True</b> , aby włączyć generowanie dowodów DPoP.                         |
| <code>DPoPOptions.Algorithm</code>    | Algorytm podpisywania. Obsługiwane wartości: <b>ES256</b> (zalecany), <b>RS256</b> . |
| <code>DPoPOptions.PrivateKey</code>   | Zakodowany w PEM klucz prywatny używany do podpisywania tokenu JWT dowodu DPoP.      |
| <code>DPoPOptions.PublicKeyJWK</code> | Klucz publiczny w formacie JWK (JSON Web Key), zawarty w nagłówku dowodu DPoP.       |

## Metody

| Metoda                           | Opis                                                                                                                                                                          |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>GenerateDPoPKeyPair</code> | Generuje parę kluczy ES256 przy użyciu OpenSSL i automatycznie ustawia zarówno <code>DPoPOptions.PrivateKey</code> (PEM), jak i <code>DPoPOptions.PublicKeyJWK</code> (JSON). |

|                                        |                                                                                                                                                                                                  |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GetDPoPProof(Method, URL, AccessToken) | Generuje dowód JWT DPoP dla podanej metody HTTP, adresu URL i tokenu dostępu. Zwraca podpisany ciąg JWT. Dowód zawiera roszczenie ath (skrót tokenu dostępu) w celu powiązania dowodu z tokenem. |
| GetDPoPJWKThumbprint                   | Zwraca skrót JWK (SHA-256) zgodny z RFC 7638 klucza publicznego, używany przez serwer do powiązania tokenu.                                                                                      |

## Obsługa nonce

Niektóre serwery autoryzacji wymagają dostarczonego przez serwer nonce w dowodzie DPoP. Gdy serwer odpowiada nagłówkiem DPoP-Nonce i statusem 400 lub 401, komponent automatycznie:

1. Wyodrębnia nonce z nagłówka odpowiedzi DPoP-Nonce.
2. Dołącza nonce do oświadczenia nonce następnego DPoP proof JWT.
3. Ponawia żądanie z zaktualizowanym dowodem.

## Niestandardowe podpisywanie za pomocą OnDPoPSign (HSM / karta inteligentna)

W niektórych środowiskach klucz prywatny jest przechowywany na sprzętowym module bezpieczeństwa (HSM) lub karcie inteligentnej (np. przez PKCS#11 lub Windows CNG) i **nie można go wyodrębnić**. Zdarzenie OnDPoPSign umożliwia zewnętrzne podpisywanie, podczas gdy komponent obsługuje wszystko pozostałe (strukturę dowodu, nonce, nagłówki).

Gdy zdarzenie OnDPoPSign jest przypisane, komponent wywołuje je przed próbą wewnętrznego podpisywania. Jeśli ustawisz `Handled := True` i podasz `Signature`, komponent używa Twojego podpisu i pomija wewnętrzne podpisywanie. W takim przypadku właściwość `DPoPOptions.PrivateKey` nie jest wymagana.

### Sygnatura zdarzenia

| Parametr    | Kierunek | Opis                                                                                                                                                |
|-------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| SignInInput | in       | Dane do podpisania: Base64URL(header) + "." + Base64URL(payload). Jest to standardowe wejście podpisywania JWS.                                     |
| Algorithm   | in       | Nazwa algorytmu: "ES256" lub "RS256".                                                                                                               |
| Signature   | wyjście  | Należy ustawić tu bajty podpisu zakodowane w formacie Base64URL, wygenerowane przez urządzenie podpisujące.                                         |
| Handled     | wyjście  | Ustaw na True, aby używać własnego podpisu. Jeśli wartość wynosi False, komponent używa wewnętrznego podpisywania za pomocą właściwości PrivateKey. |

### Przykład karty inteligentnej

Poniższy przykład pokazuje, jak używać DPoP z kartą podpisującą. Klucz prywatny nigdy nie opuszcza karty; do podpisania wysyłany jest jedynie skrót.

```
// Configure DPoP - no PrivateKey needed when using OnDPoPSign
OAuth2->DPoPOptions->Enabled = true;
OAuth2->DPoPOptions->Algorithm = dpopES256;
OAuth2->DPoPOptions->PublicKeyJWK =
    "{\"kty\":\"EC\",\"crv\":\"P-256\",\"x\":\"...\",\"y\":\"...\"}";
OAuth2->OnDPoPSign = OnDPoPSignHandler;
OAuth2->Start();
```

```

void __fastcall TForm1::OnDPoPSignHandler(TObject *Sender,
    const UnicodeString SigningInput, const UnicodeString Algorithm,
    UnicodeString &Signature, bool &Handled)
{
    // Hash and sign with smart card
    TBytes Hash = GetHashSHA256(SigningInput);
    TBytes Sig = MySmartCardAPI->SignHash(Hash);
    Signature = EncodeBase64URL(Sig);
    Handled = true;
}

```

Podejście to działa z dowolnym backendem podpisującym: tokenami PKCS#11, Windows CNG (Crypto-API Next Generation), Azure Key Vault, AWS CloudHSM lub dowolnym niestandardowym sprzętem zdolnym do generowania podpisów ECDSA/RSA.

## Przykład

```

// Configure the grant type
OAuth2->OAuth2Options->GrantType = auth2CodePKCE;
OAuth2->OAuth2Options->ClientId = "your-client-id";
OAuth2->AuthorizationServerOptions->AuthURL = "https://provider.com/oauth2/authorize";
OAuth2->AuthorizationServerOptions->TokenURL = "https://provider.com/oauth2/token";
OAuth2->AuthorizationServerOptions->Scope->Text = "openid profile";
// Enable DPoP
OAuth2->DPoPOptions->Enabled = true;
OAuth2->DPoPOptions->Algorithm = "ES256";
OAuth2->DPoPOptions->PrivateKey = LoadPrivateKeyFromFile("dpop_private.pem");
OAuth2->DPoPOptions->PublicKeyJWK = "{\"kty\":\"EC\",\"crv\":\"P-256\",\"x\":\"...\",\"y\":\"...\"}";
OAuth2->Start();
// Generate DPoP proof for API calls
UnicodeString Proof = OAuth2->GetDPoPProof("GET", "https://api.provider.com/resource");

```

# TsgcHTTP\_OAuth2\_Server

Ten komponent zapewnia implementację protokołu OAuth2 w komponentach po stronie serwera.

## Wprowadzenie

Ten komponent zapewnia implementację protokołu OAuth2 w komponentach po stronie serwera.

Komponenty serwera posiadają właściwość `Authorization.OAuth.OAuth2`, do której można przypisać instancję `TsgcHTTP_OAuth2_Server`. Gdy uwierzytelnianie jest włączone i właściwość `OAuth2` jest podłączona do komponentu serwera `OAuth2`, żądania `WebSocket` i `HTTP` wymagają tokenu `Bearer` do przetworzenia; w przeciwnym razie połączenie jest automatycznie zamykane.

```
TsgcHTTP_OAuth2_Server *OAuth2 = new TsgcHTTP_OAuth2_Server(this);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2;
```

Serwer obsługuje następujące typy autoryzacji:

- **auth2Code:** Stosowany do uwierzytelniania i autoryzacji w większości typów aplikacji, w tym jednostronicowych, webowych i natywnych. Przepływ umożliwia aplikacjom bezpieczne uzyskiwanie `access_token`, które można wykorzystać do dostępu do chronionych zasobów, a także tokenów odświeżania do uzyskiwania kolejnych `access_token` oraz tokenów ID dla zalogowanego użytkownika.
- **auth2ClientCredentials:** Ten typ przyznania jest powszechnie stosowany w interakcjach między serwerami, które muszą działać w tle bez bezpośredniej interakcji z użytkownikiem. Tego rodzaju aplikacje są często nazywane demonami lub kontami usług.
- **password** (Resource Owner Password Credentials): Umożliwia aplikacji logowanie użytkownika przez bezpośrednią obsługę jego danych uwierzytelniających. Klient wysyła nazwę użytkownika i hasło do punktu końcowego tokenu.
- **urn:ietf:params:oauth:grant-type:device\_code** (Device Code): Przyznanie autoryzacji urządzeniu zgodnie z [RFC 8628](#). Umożliwia urządzeniom z ograniczonym interfejsem wejściowym (smart TV, urządzenia IoT) uzyskanie autoryzacji użytkownika poprzez autoryzację na urządzeniu pomocniczym.

Typ autoryzacji można dostosować podczas rejestracji aplikacji; domyślnie obsługiwane są wszystkie typy autoryzacji.

## Endpoints

Domyślnie komponent jest skonfigurowany z następującymi punktami końcowymi do obsługi żądań autoryzacji i tokenów

**Autoryzacja:** `/sgc/oauth2/auth`

**Token:** `/sgc/oauth2/token`

**Revocation:** `/sgc/oauth2/revoke`

**Introspection:** `/sgc/oauth2/introspect`

**Autoryzacja urządzenia:** /sgc/oauth2/device

**Weryfikacja urządzenia:** /sgc/oauth2/device/verify

Jeśli więc serwer nasłuchuje na porcie 443, a domena to [www.esegece.com](http://www.esegece.com), punkty końcowe będą następujące:

**Autoryzacja:** <https://www.esegece.com/sgc/oauth2/auth>

**Token:** <https://www.esegece.com/sgc/oauth2/token>

**Revocation:** <https://www.esegece.com/sgc/oauth2/revoke>

**Introspection:** <https://www.esegece.com/sgc/oauth2/introspect>

**Device Authorization:** <https://www.esegece.com/sgc/oauth2/device>

**Weryfikacja urządzenia:** <https://www.esegece.com/sgc/oauth2/device/verify>

Punkty końcowe można skonfigurować we właściwości OAuth2Options.

Domyślnie włączone jest rozszerzenie **PKCE** (rozszerzenie przepływu kodu autoryzacji zapobiegające atakom CSRF i wstrzykiwaniu kodu autoryzacji).

## Konfiguracja

Przed rozpoczęciem procesu OAuth2 należy zarejestrować aplikacje, które będą dostępne; operację tę wykonuje się przy użyciu właściwości Apps komponentu serwera OAuth2.

### Rejestracja aplikacji

Użyj Apps.AddApp, aby dodać nową aplikację do serwera OAuth2; należy ustawić następujące parametry:

- **App Name:** to nazwa aplikacji. Przykład: MyApp
- **RedirectURI:** adres, na który będą przekierowywane odpowiedzi. Przykład: <http://127.0.0.1:8080>
- **ClientId:** informacja publiczna stanowiąca identyfikator klienta.
- **ClientSecret:** musi być przechowywany w poufności.

Opcjonalnie można ustawić następujące parametry:

- **ExpiresIn:** domyślnie wynosi 3600 sekund, co oznacza, że token wygaśnie po 1 godzinie; w razie potrzeby można ustawić większą wartość.
- **RefreshToken:** domyślnie tokeny odświeżania są obsługiwane; jeśli nie, należy ustawić ten parametr na false.
- **AllowedGrantTypes:** domyślnie obsługiwane są wszystkie typy przyznania dostępu (auth2Code i auth2ClientCredentials), jednak serwer można skonfigurować tak, aby zezwalał wyłącznie na Code Authorization lub wyłącznie na Client Credentials.

### Usuwanie aplikacji

Użyj metody Apps.RemoveApp, aby usunąć istniejącą aplikację.

### AddToken

Jeśli serwer został ponownie uruchomiony w trakcie wydawania tokenów, można odzyskać te tokeny za pomocą metody AddToken przed uruchomieniem serwera OAuth2 i po zarejestrowaniu aplikacji

- **AppName:** nazwa aplikacji.
- **Token:** token dostępu.
- **Expires:** moment wygaśnięcia tokenu.
- **RefreshToken:** token odświeżający.

## RemoveToken

Usuwa już wydany token.

## OAuth2Options

Właściwość OAuth2Options umożliwia konfigurację punktów końcowych serwera i opcjonalnych funkcji.

### Odwołanie

Unieważnianie tokenów zgodnie z [RFC 7009](#). Po włączeniu klienci mogą unieważniać wcześniej wydane tokeny dostępu lub odświeżania.

- **OAuth2Options.Revocation.Enabled:** ustawienie na True włącza punkt końcowy unieważniania.
- **OAuth2Options.Revocation.URL:** ścieżka URL punktu końcowego. Domyślnie: /sgc/oauth2/revoke

### Introspekcja

Introspekcja tokenu zgodnie z [RFC 7662](#). Po włączeniu serwery zasobów mogą odpytywać serwer autoryzacji w celu ustalenia aktywnego stanu i metadanych tokenu.

- **OAuth2Options.Introspection.Enabled:** ustaw na True, aby włączyć punkt końcowy introspekcji.
- **OAuth2Options.Introspection.URL:** ścieżka URL punktu końcowego. Domyślnie: /sgc/oauth2/introspect

### DeviceAuthorization

Device Authorization Grant zgodnie z [RFC 8628](#). Po włączeniu urządzenia z ograniczeniami wejścia mogą żądać autoryzacji, umożliwiając użytkownikowi autoryzację na urządzeniu pomocniczym.

- **OAuth2Options.DeviceAuthorization.Enabled:** należy ustawić na True, aby włączyć punkt końcowy autoryzacji urządzenia.
- **OAuth2Options.DeviceAuthorization.URL:** ścieżka URL punktu końcowego dla żądań kodu urządzenia. Domyślnie: /sgc/oauth2/device
- **OAuth2Options.DeviceAuthorization.VerificationURL:** ścieżka URL punktu końcowego strony weryfikacji użytkownika. Domyślnie: /sgc/oauth2/device/verify
- **OAuth2Options.DeviceAuthorization.ExpiresIn:** czas życia kodu urządzenia w sekundach. Wartość domyślna: 600 (10 minut).
- **OAuth2Options.DeviceAuthorization.Interval:** minimalny interwał odpytywania w sekundach, jaki klient powinien stosować podczas odpytywania punktu końcowego tokenu. Domyślnie: 5.

## Dokumentacja

- Właściwości — 2 właściwości dostępne na komponencie.
- Metody — 11 metod publicznych.
- Zdarzenia — 13 zdarzeń do obsługi.

# Przewodniki

- **QuickStart**
  - [Przykład serwera OAuth2](#)
  - [OAuth2 Dostosowanie HTML logowania](#)
  - [Punkty końcowe serwera OAuth2](#)
  - [OAuth2 Rejestracja aplikacji](#)
  - [OAuth2 Odzyskiwanie tokenów dostępu](#)
- **Uwierzytelnianie**
  - [Uwierzytelnianie serwera OAuth2](#)
  - [OAuth2 — wyłączenie uwierzytelniania dla wybranych URL](#)

# OAuth2 | Przykład serwerowy

Poniżej przedstawiono prosty przykład serwera OAuth2 z użyciem [TsgcWebSocketHTTPServer](#).

Najpierw należy utworzyć nowy `TsgcWebSocketHTTPServer` nasłuchujący na porcie 443 i używający certyfikatu z podpisem własnym w pliku `sgc.pem`.

```
oServer = new TsgcWebSocketHTTPServer(this);
oServer->Port := 80;
oServer->SSLOptions->Port = 443;
oServer->SSLOptions->CertFile = "sgc.pem";
oServer->SSLOptions->KeyFile = "sgc.pem";
oServer->SSLOptions->RootCertFile = "sgc.pem";
oServer->SSL = true;
```

Następnie utwórz nową instancję `TsgcHTTP_OAuth2_Server` i przypisz ją do wcześniej utworzonego serwera.

Zarejestruj nową aplikację, podając następujące wartości:

Name: MyApp

RedirectURI: http://127.0.0.1:8080

ClientId: client-id

ClientSecret: client-secret

```
OAuth2 = new TsgcHTTP_OAuth2_Server.Create(this);
OAuth2->Apps->AddApp("MyApp", "http://127.0.0.1:8080", "client-id", "client-secret");
oServer->Authentication->Enabled = true;
oServer->Authentication->OAuth->OAuth2 = OAuth2;
```

Następnie należy obsłużyć zdarzenie `OnOAuth2Authentication` komponentu serwera OAuth2 i zaimplementować własną metodę logowania użytkowników. Do akceptowania logowania zostanie użyta para „user/secret”.

```
void OnOAuth2Authentication(TsgcWSConnection *Connection, TsgcHTTPOAuth2Request *OAuth2, string aUser,
    string aPassword, ref bool Authenticated)
{
    if ((aUser == "user") and (aPassword == "secret"))
    {
        Authenticated = true;
    }
}
```

Na koniec uruchom serwer i użyj klienta OAuth2 do logowania; przykładowo można skorzystać z komponentu [TsgcHTTP\\_OAuth2\\_Client](#) dołączonego do biblioteki `sgcWebSockets`.





Żądanie nowego tokenu dostępu — zostanie wyświetlona nowa sesja przeglądarki internetowej, w której użytkownik musi zezwolić na połączenie i zalogować się.



Jeśli logowanie zakończy się pomyślnie, do klienta zostanie zwrócony nowy token. Następnie wszystkie żądania muszą zawierać ten token Bearer w nagłówkach HTTP.



## OAuth2 | Dostosowywanie strony logowania HTML

Gdy klient OAuth2 wysyła żądanie uzyskania nowego tokenu dostępu, w przeglądarce internetowej wyświetlana jest strona internetowa umożliwiająca zezwolenie na to połączenie i zalogowanie się przy użyciu nazwy użytkownika i hasła.

Strona HTML jest domyślnie zawarta w komponencie serwera, jednak kod ten można dostosować przy użyciu zdarzenia **OnAuth2BeforeDispatchPage**.

```
void OnOAuth2BeforeDispatchPage(TObject *Sender; TsgcHTTPOAuth2Request *OAuth2; ref string HTML)
{
    HTML = "your custom html";
}
```

Jeśli kod HTML zostanie dostosowany przez zastąpienie go całkowicie nowym kodem, należy co najmniej zachować formularz, za pomocą którego przesyłane są nazwa użytkownika i hasło:

```
<form action="">

<input type="hidden" name="request_type" value="sign-in" />

<input type="username" name="username" placeholder="Username" />

<input type="password" name="password" placeholder="Hasło" />

<input type="hidden" name="id" value="" />

<p></p>

<button>Sign In</button>

</form>
```

Parametr id, który jest ukryty, musi zachować tę samą wartość co oryginalny formularz, aby serwer mógł zidentyfikować żądanie.

# OAuth2 | Endpointy serwera

Domyślnie serwer OAuth2 używa następujących endpointów:

**Autoryzacja:** /sgc/oauth2/auth

**Token:** /sgc/oauth2/token

**Unieważnienie:** /sgc/oauth2/revoke (POST) — unieważnia tokeny zgodnie z [RFC 7009](#)

**Introspection:** /sgc/oauth2/introspect (POST) - Zwraca metadane tokenu zgodnie z [RFC 7662](#)

**Device Authorization:** /sgc/oauth2/device (POST) - Wydaje kody urządzenia zgodnie z [RFC 8628](#)

**Weryfikacja urządzenia:** /sgc/oauth2/device/verify (GET/POST) — strona weryfikacji użytkownika

Oznacza to, że jeśli serwer nasłuchuje na adresie IP 80.54.41.30 i porcie 8443, pełne punkty końcowe OAuth2 będą następujące:

**Autoryzacja:** https://80.54.41.30:8443/sgc/oauth2/auth

**Token:** https://80.54.41.30:8443/sgc/oauth2/token

**Revocation:** https://80.54.41.30:8443/sgc/oauth2/revoke

**Introspection:** https://80.54.41.30:8443/sgc/oauth2/introspect

**Device Authorization:** https://80.54.41.30:8443/sgc/oauth2/device

**Device Verification:** https://80.54.41.30:8443/sgc/oauth2/device/verify

Te punkty końcowe można łatwo zmodyfikować — wystarczy uzyskać dostęp do właściwości OAuth2Options komponentu i zmienić adresy URL autoryzacji i tokenu.

**Przykład:** jeśli punkty końcowe muszą być

**Authorization:** https://80.54.41.30:8443/authentication/auth

**Token:** https://80.54.41.30:8443/authentication/token

Ustaw właściwość OAuth2Options z następującymi wartościami:

**OAuth2Options.Authorization.URL** = /authentication/auth

**OAuth2Options.Token.URL** = /authentication/token

To samo podejście ma zastosowanie do pozostałych punktów końcowych:

**OAuth2Options.Revocation.URL** = /authentication/revoke

**OAuth2Options.Introspection.URL** = /authentication/introspect

**OAuth2Options.DeviceAuthorization.URL** = /authentication/device

**OAuth2Options.DeviceAuthorization.VerificationURL** = /authentication/device/verify

# OAuth2 | Rejestracja aplikacji

Przed wysłaniem nowego żądania OAuth2 przez klienta aplikacja musi zostać zarejestrowana na serwerze.

Rejestracja nowej aplikacji wymaga podania następujących informacji:

- **App Name:** to nazwa aplikacji. Przykład: MyApp
- **RedirectURI:** adres, na który będą przekierowywane odpowiedzi. Przykład: http://127.0.0.1:8080
- **ClientId:** informacja publiczna stanowiąca identyfikator klienta.
- **ClientSecret:** musi być przechowywany w poufności.

Opcjonalnie można ustawić następujące parametry:

- **ExpiresIn:** domyślnie wynosi 3600 sekund, co oznacza, że token wygaśnie po 1 godzinie; w razie potrzeby można ustawić większą wartość.
- **RefreshToken:** domyślnie tokeny odświeżania są obsługiwane; jeśli nie, należy ustawić ten parametr na false.

Jeśli nowy klient chce uwierzytelnić się przy użyciu OAuth2, aplikacja musi najpierw zostać zarejestrowana na serwerze, co można zrealizować za pomocą:

## 1. RegisterApp

Metoda ta wymaga podania nazwy aplikacji (App Name) i identyfikatora URI przekierowania (RedirectURI) i zwraca identyfikator klienta (ClientId) oraz tajny klucz klienta (ClientSecret).

## 2. Apps.AddApp

Metoda ta wymaga parametrów AppName, RedirectURI, ClientId i ClientSecret. Zazwyczaj stosowana, gdy serwer posiada już utworzone aplikacje i należy je załadować przed uruchomieniem serwera.

Obie metody wykonują to samo działanie, rejestrując aplikację na serwerze, jednak pierwsza jest najbardziej przydatna przy pierwszym rejestrowaniu aplikacji, natomiast druga, gdy chcemy załadować wszystkie zarejestrowane aplikacje przed uruchomieniem serwera (ponieważ są zapisane na przykład w bazie danych).

# OAuth2 | Odzyskiwanie tokenów dostępu

Jeśli serwer OAuth2 zostanie zniszczony (np. wskutek restartu), a istnieją ważne wydane tokeny dostępu, są one domyślnie tracone. Można je odzyskać za pomocą metody **AddToken**, która przechowuje tokeny na serwerze OAuth2.

Dodanie tokenu wymaga podania następujących informacji:

- **AppName:** nazwa aplikacji.
- **Token:** token dostępu.
- **Expires:** moment wygaśnięcia tokenu.
- **RefreshToken:** token odświeżający.

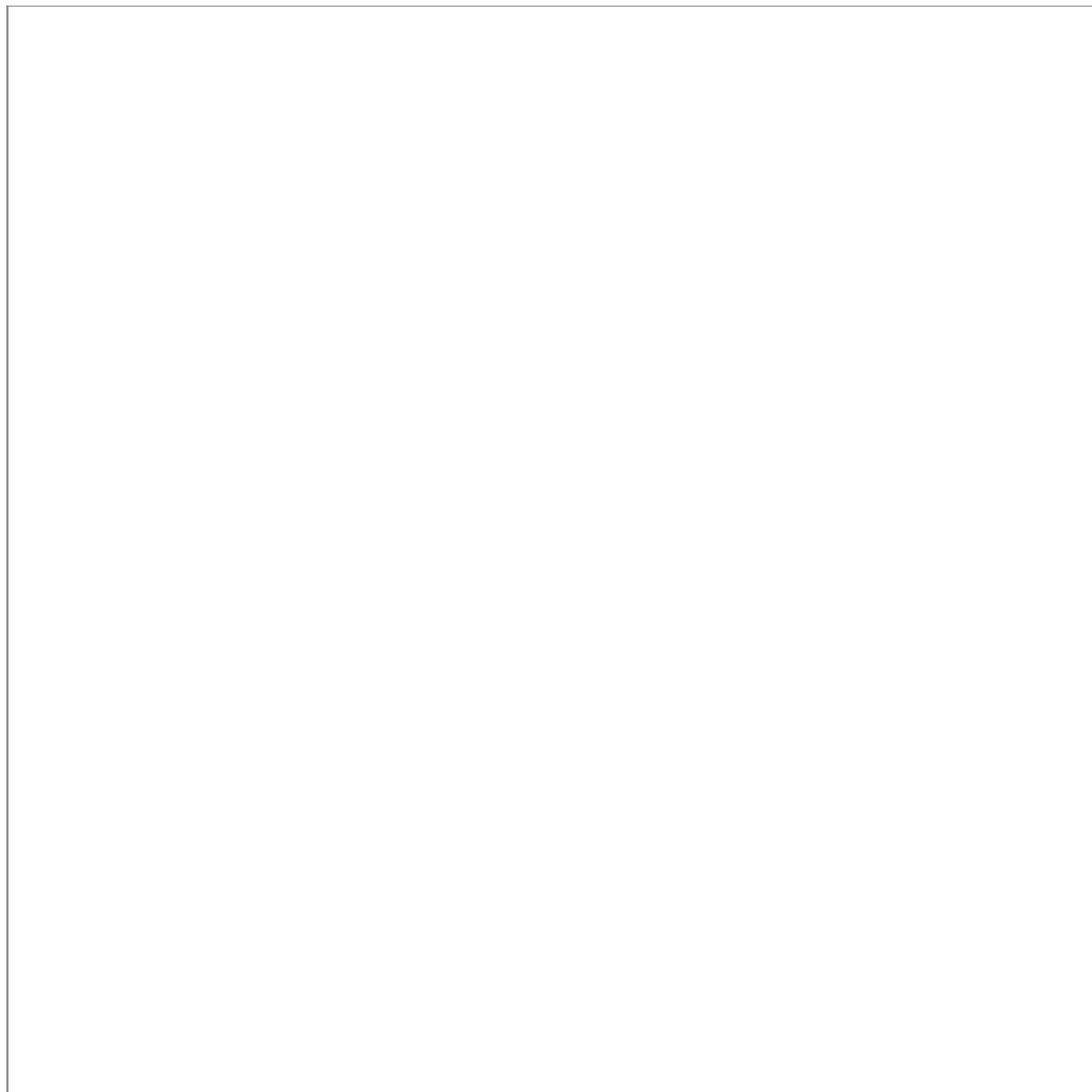
Wydane tokeny można zapisywać, obsługując zdarzenie **OAuth2AfterAccessToken**.

```
private void OnOAuth2AfterAccessToken(TObject *Sender, TsgcWConnection *Connection, TsgcHTTPOAuth2Request *OAuth2,
    string aResponse)
{
    // ... store OAuth2 Token data
}

OAuth2 = new TsgcHTTP_OAuth2_Server.Create(this);
OAuth2->Apps->AddApp("MyApp", "http://127.0.0.1:8080", "client-id", "client-secret");
OAuth2->AddToken("MyApp", "12146ce12b0e4813987f2794f768905cefc39da6fbd54f6d9b38387489280608", EncodeDate(2022,1,1),
    "ef3e3dfa56ec44109c3d345b1541f08e539ce21432d9433099b48a3d08d34bc0");
oServer->Authentication->Enabled = true;
oServer->Authentication->OAuth->OAuth2 = OAuth2;
```

## OAuth2 | Uwierzytelnianie serwera

Gdy klient OAuth2 żąda nowej autoryzacji, serwer wyświetla stronę internetową, na której użytkownik musi zezwolić na połączenie, a następnie się zalogować. Strona ta jest dostarczana przez bibliotekę `sgcWebSockets` i jest wyświetlana automatycznie, gdy klient żąda autoryzacji.



Jeśli użytkownik zezwoli na dostęp, zostanie wyświetlony formularz logowania, w którym należy podać nazwę użytkownika i hasło.

Te dane zostaną odebrane w zdarzeniu `OnOAuth2Authentication`, dlatego należy sprawdzić poprawność nazwy użytkownika i hasła, a jeśli są prawidłowe, ustawić parametr `Authenticated` na `true`.

```
void OnOAuth2Authentication(TsgcWSConnection *Connection, TsgcHTTPOAuth2Request *OAuth2, string aUser,
    string aPassword, ref bool Authenticated)
{
    if ((aUser == "user") and (aPassword == "secret"))
    {
        Authenticated = true;
    }
}
```

## OAuth2 | Brak uwierzytelnienia adresów URL

Domyślnie, gdy OAuth2 jest włączony po stronie serwera, wszystkie żądania HTTP wymagają uwierzytelnienia za pomocą tokenów Bearer.

Jeśli chcesz zezwolić na dostęp do niektórych adresów URL bez konieczności używania tokenu Bearer, możesz użyć zdarzenia **OnOAuth2BeforeRequest**

```
procedure OnOAuth2BeforeRequest(TObject *Sender; TsgcWSConnection *aConnection; TStringList *aHeaders;
    ref bool Cancel)
{
    if (DecodeGETFullPath(aHeaders) == "/Public.html")
    {
        Cancel = true;
    }
}
```



# OAuth2 Server | Authorization Code Grant

## Przegląd

Grant kodu autoryzacyjnego jest najczęstszym przepływem OAuth2 dla aplikacji po stronie serwera. Komponent [TsgcHTTP\\_OAuth2\\_Server](#) udostępnia dwa punkty końcowe obsługujące ten przepływ: punkt końcowy autoryzacji wyświetlający użytkownikowi stronę logowania oraz punkt końcowy tokenu wymieniający kod autoryzacyjny na token dostępu.

Ten przepływ jest odpowiedni dla aplikacji internetowych, desktopowych i mobilnych, w których klient może wchodzić w interakcję z przeglądarką.

## Przepływ

1. Klient przekierowuje użytkownika pod adres `/sgc/oauth2/auth?response_type=code&client_id=...&redirect_uri=...&scope=...&state=...`
2. Serwer wywołuje zdarzenie **OnOAuth2BeforeDispatchPage**, aby umożliwić dostosowanie strony logowania HTML.
3. Użytkownik przesyła poświadczenia. Serwer wywołuje zdarzenie **OnOAuth2Authentication** w celu weryfikacji nazwy użytkownika i hasła.
4. Jeśli uwierzytelnienie się powiedzie, serwer przekierowuje użytkownika do `redirect_uri` z dołączonym `?code=...&state=...`
5. Klient wysyła żądanie POST na `/sgc/oauth2/token` z parametrami `grant_type=authorization_code&code=...&redirect_uri=...&client_id=...&client_secret=...`
6. Serwer weryfikuje kod autoryzacji i dane uwierzytelniające klienta, wywołuje zdarzenie **OnOAuth2AfterAccessToken** i zwraca odpowiedź JSON zawierającą `access_token`, `token_type`, `expires_in` oraz opcjonalnie `refresh_token`.

## Konfiguracja

| Właściwość                                       | Opis                                                                                                                                                                                     |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.Authorization.Enabled</code> | Ustawienie na <code>True</code> włącza punkt końcowy autoryzacji. Domyślnie: <code>True</code> .                                                                                         |
| <code>OAuth2Options.Authorization.URL</code>     | Ścieżka URL punktu końcowego autoryzacji. Domyślnie: <code>/sgc/oauth2/auth</code>                                                                                                       |
| <code>OAuth2Options.Token.Enabled</code>         | Ustawić na <code>True</code> , aby włączyć punkt końcowy tokenu. Domyślnie: <code>True</code> .                                                                                          |
| <code>OAuth2Options.Token.URL</code>             | Ścieżka URL punktu końcowego tokenu. Domyślnie: <code>/sgc/oauth2/token</code>                                                                                                           |
| <code>OAuth2Options.PKCE</code>                  | Gdy wartość to <code>True</code> , serwer wymaga PKCE ( <code>code_challenge</code> i <code>code_verifier</code> ) dla żądań kodu autoryzacyjnego. Wartość domyślna: <code>True</code> . |

## Rejestracja aplikacji

Zanim klienci będą mogli korzystać z przepływu kodu autoryzacji, aplikacja musi zostać zarejestrowana na serwerze. Należy użyć metody `Apps.AddApp` i upewnić się, że właściwość `AllowedGrantTypes` zawiera wartość `auth2Code`.

```
OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",  
    "my-client-id", "my-client-secret", 3600, true,  
    TsgcOAuth2GrantTypes() << auth2Code);
```

## Zdarzenia

| Zdarzenie                        | Opis                                                                                                                                                      |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| OnOAuth2BeforeDispatchPage       | Wywoływane przed wysłaniem strony HTML logowania do użytkownika. Umożliwia dostosowanie zawartości HTML.                                                  |
| OnOAuth2Authentication           | Wywoływane, gdy użytkownik przesyła poświadczenia. Należy tu zweryfikować nazwę użytkownika i hasło, a następnie ustawić Authenticated na True lub False. |
| OnOAuth2AfterAccessToken         | Wywoływane po pomyślnym wystawieniu tokenu dostępu przez serwer. Przydatne do rejestrowania zdarzeń.                                                      |
| OnOAuth2AfterValidateAccessToken | Wywoływane, gdy klient wysyła żądanie z prawidłowym, nieprzeterminowanym tokenem. Przydatne do rejestrowania i audytowania.                               |

## Przykład

Poniższy przykład przedstawia kompletną konfigurację serwera z uprawnieniami opartymi na kodzie autoryzacyjnym, obejmującą rejestrację aplikacji i procedury obsługi zdarzeń.

```
// Create and configure OAuth2 server  
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);  
  
// Register an application  
OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",  
    "my-client-id", "my-client-secret", 3600, true,  
    TsgcOAuth2GrantTypes() << auth2Code);  
  
// Attach to HTTP server  
Server->Authentication->Enabled = true;  
Server->Authentication->OAuth->OAuth2 = OAuth2Server;  
  
void __fastcall TForm1::OAuth2ServerOAuth2Authentication(TObject *Sender,  
    const UnicodeString User, const UnicodeString Password, bool &Authenticated)  
{  
    Authenticated = (User == "admin") && (Password == "secret");  
}  
  
void __fastcall TForm1::OAuth2ServerOAuth2AfterAccessToken(TObject *Sender,  
    const UnicodeString Token, const UnicodeString RefreshToken, int ExpiresIn)  
{  
    Log("Access token issued: " + Token);  
}
```

## Obsługa PKCE

Gdy OAuth2Options.PKCE jest ustawiony na True (domyślnie), serwer wymusza Proof Key for Code Exchange. Klient musi dołączyć parametr code\_challenge do żądania autoryzacji oraz odpowiadający mu code\_verifier w żądaniu tokenu. Serwer weryfikuje, czy code\_verifier pasuje do oryginalnego code\_challenge przed wydaniem tokenu.

PKCE jest zalecane dla wszystkich klientów, szczególnie klientów publicznych (aplikacji natywnych i SPA), które nie mogą bezpiecznie przechowywać sekretu klienta.

# Serwer OAuth2 | Przyznanie poświadczeń klienta

## Przegląd

Grant Client Credentials jest używany w komunikacji machine-to-machine (M2M), gdzie interakcja z użytkownikiem nie jest wymagana. Klient uwierzytelnia się bezpośrednio w [TsgcHTTP\\_OAuth2\\_Server](#) przy użyciu własnych poświadczeń (client\_id i client\_secret) i otrzymuje token dostępu. Nie istnieje krok z kodem autoryzacyjnym, przekierowanie ani strona logowania użytkownika.

Ten typ przyznania jest odpowiedni, gdy aplikacja kliencka działa we własnym imieniu, a nie w imieniu konkretnego użytkownika.

## Przepływ

1. Klient wysyła żądanie POST do punktu końcowego tokenu z parametrami  
grant\_type=client\_credentials&client\_id=...&client\_secret=...&scope=...
2. Serwer weryfikuje poświadczenia klienta względem zarejestrowanej aplikacji.
3. Serwer wywołuje **OnOAuth2AfterAccessToken** i zwraca odpowiedź JSON z access\_token, token\_type i expires\_in.

Należy zauważyć, że dla tego typu przyznania nie jest wywoływane zdarzenie OnOAuth2Authentication, ponieważ nie ma użytkownika do uwierzytelnienia. Serwer automatycznie weryfikuje client\_id i client\_secret względem zarejestrowanej aplikacji.

## Konfiguracja

Należy zarejestrować aplikację z AllowedGrantTypes zawierającym auth2ClientCredentials. Dla tego przepływu wymagany jest tylko punkt końcowy tokena.

| Właściwość                  | Opis                                                                |
|-----------------------------|---------------------------------------------------------------------|
| OAuth2Options.Token.Enabled | Ustawić na True, aby włączyć punkt końcowy tokenu. Domyślnie: True. |
| OAuth2Options.Token.URL     | Ścieżka URL punktu końcowego tokenu. Domyślnie: /sgc/oauth2/token   |

## Rejestracja aplikacji

```
OAuth2Server->Apps->AddApp("MyServiceApp", "",  
    "service-client-id", "service-client-secret", 3600, false,  
    TsgcOAuth2GrantTypes() << auth2ClientCredentials);
```

Właściwość RedirectURI może zostać pozostawiona pusta, ponieważ przekierowanie nie jest wymagane. Właściwość RefreshToken jest zazwyczaj ustawiana na False, ponieważ klient może w dowolnym momencie zażądać nowego tokenu przy użyciu swoich poświadczeń.

## Zdarzenia

| Zdarzenie                | Opis                                                                                                 |
|--------------------------|------------------------------------------------------------------------------------------------------|
| OnOAuth2AfterAccessToken | Wywoływane po pomyślnym wystawieniu tokenu dostępu przez serwer. Przydatne do rejestrowania zdarzeń. |

OnOAuth2AfterValidateAccessToken Wywoływane, gdy klient wysyła żądanie z prawidłowym tokenem. Przydatne do celów audytu.

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);
OAuth2Server->Apps->AddApp("BackendService", "",
    "service-client-id", "service-client-secret", 7200, false,
    TsgcOAuth2GrantTypes() << auth2ClientCredentials);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;
void __fastcall TForm1::OAuth2ServerOAuth2AfterAccessToken(TObject *Sender,
    const UnicodeString Token, const UnicodeString RefreshToken, int ExpiresIn)
{
    Log("Service token issued: " + Token);
}
```

## Kiedy stosować

Przyznania poświadczeń klienta (Client Credentials grant) na serwerze należy używać do:

- Usługi backendowe i demony, które uwierzytelniają się bez interakcji z użytkownikiem.
- Komunikacja API serwer-serwer.
- Uwierzytelnianie między mikroserwisami.
- Zautomatyzowane procesy, zaplanowane zadania i operacje wsadowe.

# Serwer OAuth2 | Przyznanie poświadczeń właściciela zasobu

## Przegląd

Grant Resource Owner Password Credentials (ROPC) umożliwia klientowi bezpośrednie wysłanie nazwy użytkownika i hasła do punktu końcowego tokenu. [TsgcHTTP\\_OAuth2\\_Server](#) weryfikuje poświadczenia za pomocą zdarzenia **OnOAuth2Authentication** i wydaje token dostępu w przypadku pomyślnego uwierzytelnienia.

Ten przepływ nie obejmuje przekierowania przeglądarki ani strony logowania. Klient zbiera poświadczenia bezpośrednio od użytkownika i przesyła je do serwera.

## Ostrzeżenie bezpieczeństwa

**Ten typ grantu jest przestarzały w OAuth 2.1.** Należy go stosować wyłącznie wtedy, gdy inne przepływy (takie jak Authorization Code z PKCE) nie są możliwe, np. w aplikacjach legacy lub wysoce zaufanych klientach pierwszej strony. Zalecaną alternatywą jest przepływ Authorization Code z PKCE.

## Przepływ

1. Klient wysyła do punktu końcowego tokena żądanie POST z parametrami  
grant\_type=password&username=...&password=...&client\_id=...&client\_secret=...&scope=...
2. Serwer wywołuje **OnOAuth2Authentication** w celu weryfikacji nazwy użytkownika i hasła.
3. Po pomyślnym uwierzytelnieniu serwer wywołuje zdarzenie **OnOAuth2AfterAccessToken** i zwraca odpowiedź JSON zawierającą access\_token, token\_type, expires\_in oraz opcjonalnie refresh\_token.
4. Jeśli uwierzytelnianie się nie powiedzie, serwer zwraca odpowiedź z błędem.

## Konfiguracja

Zarejestrować aplikację z AllowedGrantTypes zawierającym auth2ResourceOwnerPassword.

| Właściwość                  | Opis                                                                |
|-----------------------------|---------------------------------------------------------------------|
| OAuth2Options.Token.Enabled | Ustawić na True, aby włączyć punkt końcowy tokenu. Domyślnie: True. |
| OAuth2Options.Token.URL     | Ścieżka URL punktu końcowego tokenu. Domyślnie: /sgc/oauth2/token   |

## Rejestracja aplikacji

```
OAuth2Server->Apps->AddApp("MyApp", "",  
    "my-client-id", "my-client-secret", 3600, true,  
    TsgcOAuth2GrantTypes() << auth2ResourceOwnerPassword);
```

## Zdarzenia

| Zdarzenie | Opis |
|-----------|------|
|-----------|------|

|                                  |                                                                                                                                                        |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| OnOAuth2Authentication           | Wywoływane gdy serwer otrzymuje nazwę użytkownika i hasło. Należy tutaj zweryfikować dane uwierzytelniające i ustawić Authenticated na True lub False. |
| OnOAuth2AfterAccessToken         | Wywoływane po pomyślnym wystawieniu tokenu dostępu przez serwer. Przydatne do rejestrowania zdarzeń.                                                   |
| OnOAuth2AfterValidateAccessToken | Wywoływane, gdy klient wysyła żądanie z prawidłowym tokenem.                                                                                           |

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);
OAuth2Server->Apps->AddApp("MyApp", "",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2ResourceOwnerPassword);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;
void __fastcall TForm1::OAuth2ServerOAuth2Authentication(TObject *Sender,
    const UnicodeString User, const UnicodeString Password, bool &Authenticated)
{
    Authenticated = ValidateUser(User, Password);
}
void __fastcall TForm1::OAuth2ServerOAuth2AfterAccessToken(TObject *Sender,
    const UnicodeString Token, const UnicodeString RefreshToken, int ExpiresIn)
{
    Log("Token issued for user: " + Token);
}
```

## Kiedy stosować

Grantu Resource Owner Password Credentials należy używać tylko gdy:

- Klient jest wysoce zaufaną aplikacją pierwszej strony.
- Inne typy przyznania (Authorization Code z PKCE) nie są możliwe do zastosowania.
- Migracja starszych aplikacji, które bezpośrednio zbierają poświadczenia użytkownika.

W przypadku nowych aplikacji zawsze należy preferować grant [Authorization Code](#) z PKCE.

# Serwer OAuth2 | Przyznanie tokenu odświeżania

## Przegląd

Przyznanie tokenu odświeżania (Refresh Token grant) umożliwia klientowi wymianę wcześniej wydane-go tokenu odświeżania na nowy token dostępu bez konieczności interakcji użytkownika.

[TsgcHTTP\\_OAuth2\\_Server](#) waliduje token odświeżania i wydaje nowy token dostępu (opcjonalnie również nowy token odświeżania).

Jest to przydatne w przypadku długotrwałych sesji, w których token dostępu ma krótki czas ważności, ale klient potrzebuje ciągłego dostępu do chronionych zasobów.

## Przepływ

1. Klient wysyła żądanie POST do punktu końcowego tokenu z parametrami  
`grant_type=refresh_token&refresh_token=...&client_id=...&client_secret=...&scope=...`
2. Serwer weryfikuje token odświeżania względem przechowywanych tokenów.
3. Serwer wywołuje zdarzenie **OnOAuth2AfterRefreshToken** i zwraca odpowiedź JSON z nowym `access_token`, `token_type`, `expires_in` i opcjonalnie nowym `refresh_token`.

## Konfiguracja

Tokeny odświeżania są włączane dla każdej aplikacji podczas jej rejestracji. Należy ustawić parametr `RefreshToken` na `True` w metodzie `Apps.AddApp`.

| Właściwość                                                 | Opis                                                                                                       |
|------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.Token.Enabled</code>                   | Ustawić na <code>True</code> , aby włączyć punkt końcowy tokenu. Domyślnie: <code>True</code> .            |
| <code>OAuth2Options.Token.URL</code>                       | Ścieżka URL punktu końcowego tokenu. Domyślnie: <code>/sgc/oauth2/token</code>                             |
| <code>RefreshToken</code> ( <code>AddApp</code> parameter) | Ustaw <code>True</code> , aby wydawać tokeny odświeżania dla tej aplikacji. Domyślnie: <code>True</code> . |

## Rejestracja aplikacji

Podczas rejestrowania aplikacji należy ustawić parametr `RefreshToken` na `True`, aby serwer wydawał tokeny odświeżania wraz z tokenami dostępu.

```
// RefreshToken = true (6th parameter)
OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);
```

## Zdarzenia

| Zdarzenie                              | Opis                                                                                                                                         |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OnOAuth2AfterRefreshToken</code> | Wywoływane po tym, jak serwer pomyślnie wyda nowy token dostępu na podstawie tokenu odświeżania. Przydatne do rejestrowania rotacji tokenów. |
| <code>OnOAuth2AfterAccessToken</code>  | Wywoływane po wystawieniu nowego tokenu dostępu.                                                                                             |

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);
OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;
void __fastcall TForm1::OAuth2ServerOAuth2AfterRefreshToken(TObject *Sender,
    const UnicodeString OldToken, const UnicodeString NewToken,
    const UnicodeString NewRefreshToken, int ExpiresIn)
{
    Log("Token refreshed. New token: " + NewToken);
}
```

## Rotacja tokenów

Gdy używany jest token odświeżania, serwer może wydać nowy token odświeżania wraz z nowym tokenem dostępu. Stary token odświeżania zostaje unieważniony. Mechanizm ten nazywany jest rotacją tokenów odświeżania i pomaga zapobiegać atakom powtórkowym.

## Odzyskiwanie tokenów po ponownym uruchomieniu

Jeśli serwer zostanie zrestartowany, gdy tokeny są nadal ważne, można je odzyskać za pomocą metody `Apps.AddToken` przed uruchomieniem serwera. Szczegółowe informacje można znaleźć w sekcji [OAuth2 — odzyskiwanie tokenów dostępu](#).



# Serwer OAuth2 | Device Authorization Grant (RFC 8628)

## Przegląd

Device Authorization Grant jest przeznaczony dla urządzeń z ograniczonym interfejsem wejściowym (smart TV, urządzenia IoT, narzędzia CLI), które nie mogą łatwo wyświetlić przeglądarki ani przyjąć danych z klawiatury. [TsgcHTTP\\_OAuth2\\_Server](#) udostępnia punkt końcowy autoryzacji urządzenia oraz stronę weryfikacji, dzięki czemu użytkownicy mogą autoryzować urządzenie z innego urządzenia, takiego jak telefon lub komputer.

## Przepływ

1. Urządzenie wysyła żądanie POST na `/sgc/oauth2/device` z parametrami `client_id=...` i `scope=...`
2. Serwer wywołuje zdarzenie **OnOAuth2DeviceAuthorization** i zwraca odpowiedź JSON zawierającą: `device_code`, `user_code`, `verification_uri`, `expires_in` i `interval`.
3. Urządzenie wyświetla użytkownikowi `user_code` i `verification_uri` (np. „Przejdź pod adres <https://example.com/sgc/oauth2/device/verify> i wpisz kod: ABCD-1234”).
4. Użytkownik przechodzi do URI weryfikacji na urządzeniu pomocniczym i wpisuje `user_code`.
5. Serwer wywołuje zdarzenie **OnOAuth2DeviceCodeVerification** w celu zweryfikowania `user_code` i autoryzacji urządzenia.
6. W tym czasie urządzenie odpytuje punkt końcowy tokena za pomocą `grant_type=urn:ietf:params:oauth:grant-type:device_code` i `device_code=...` oraz `client_id=...`
7. Serwer zwraca `authorization_pending` do momentu autoryzacji przez użytkownika. Po autoryzacji zwracany jest token dostępu.

## Konfiguracja

| Właściwość                                                     | Opis                                                                                                                   |
|----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.DeviceAuthorization.Enabled</code>         | Ustaw na <code>True</code> , aby włączyć punkt końcowy autoryzacji urządzenia. Wartość domyślna: <code>False</code> .  |
| <code>OAuth2Options.DeviceAuthorization.URL</code>             | Ścieżka URL punktu końcowego autoryzacji urządzenia. Domyślnie: <code>/sgc/oauth2/device</code>                        |
| <code>OAuth2Options.DeviceAuthorization.VerificationURL</code> | Adres URL strony weryfikacyjnej, na której użytkownik wprowadza kod. Domyślnie: <code>/sgc/oauth2/device/verify</code> |
| <code>OAuth2Options.DeviceAuthorization.ExpiresIn</code>       | Czas życia kodu urządzenia w sekundach. Domyślnie: 600 (10 minut).                                                     |
| <code>OAuth2Options.DeviceAuthorization.Interval</code>        | Minimalny interwał odpytywania w sekundach, który powinno stosować urządzenie. Domyślnie: 5.                           |

## Zdarzenia

| Zdarzenie | Opis |
|-----------|------|
|-----------|------|

|                                |                                                                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| OnOAuth2DeviceAuthorization    | Wywoływane, gdy urządzenie żąda kodu urządzenia. Umożliwia dostosowanie odpowiedzi lub rejestrowanie żądania.                                          |
| OnOAuth2DeviceCodeVerification | Wywoływane, gdy użytkownik przesyła kod weryfikacyjny na stronie weryfikacji urządzenia. Zweryfikuj kod użytkownika i zatwierdź lub odrzuć urządzenie. |
| OnOAuth2AfterAccessToken       | Wywoływane po wydaniu przez serwer tokenu dostępu dla urządzenia.                                                                                      |

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);
OAuth2Server->OAuth2Options->DeviceAuthorization->Enabled = true;
OAuth2Server->OAuth2Options->DeviceAuthorization->ExpiresIn = 600;
OAuth2Server->OAuth2Options->DeviceAuthorization->Interval = 5;
OAuth2Server->Apps->AddApp("DeviceApp", "",
    "device-client-id", "device-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;
void __fastcall TForm1::OAuth2ServerOAuth2DeviceCodeVerification(TObject *Sender,
    const UnicodeString UserCode, bool &Authorized)
{
    Authorized = true;
}
void __fastcall TForm1::OAuth2ServerOAuth2AfterAccessToken(TObject *Sender,
    const UnicodeString Token, const UnicodeString RefreshToken, int ExpiresIn)
{
    Log("Device authorized. Token: " + Token);
}
```

## Zachowanie przy odpytywaniu

Dopóki użytkownik nie autoryzuje urządzenia, urządzenie odpytuje punkt końcowy tokenu w skonfigurowanym interwale. Serwer odpowiada komunikatem:

- `authorization_pending` -- użytkownik nie zakończył jeszcze autoryzacji.
- `slow_down` -- urządzenie odpytuje zbyt często. Urządzenie powinno zwiększyć interwał o 5 sekund.
- `expired_token` -- kod urządzenia wygasł. Urządzenie musi ponownie uruchomić przepływ.
- `access_denied` -- użytkownik odmówił autoryzacji.

Po autoryzacji urządzenia przez użytkownika kolejne zapytanie zwraca token dostępu.

# Serwer OAuth2 | Unieważnianie tokenów (RFC 7009)

## Przegląd

Unieważnianie tokenów umożliwia klientom powiadomienie [TsgcHTTP\\_OAuth2\\_Server](#), że wcześniej wystawiony token dostępu lub token odświeżania nie jest już potrzebny. Serwer unieważnia token, umożliwiając jego dalsze użycie do uzyskania dostępu do chronionych zasobów.

Jest to przydatne, gdy użytkownik się wylogowuje, gdy aplikacja jest odinstalowywana lub gdy token mógł zostać naruszony.

## Punkt końcowy

POST do `/sgc/oauth2/revoke` z następującymi parametrami:

| Parametr                     | Opis                                                                                                   |
|------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>token</code>           | Token do unieważnienia (wymagany).                                                                     |
| <code>token_type_hint</code> | Opcjonalna wskazówka dotycząca typu tokenu: <code>access_token</code> lub <code>refresh_token</code> . |

## Zachowanie

Zgodnie z [RFC 7009](#) serwer zawsze zwraca HTTP 200 OK niezależnie od tego, czy token został znaleziony lub pomyślnie unieważniony. Zapobiega to ujawnieniu istnienia tokenu — klient nie może określić, czy token był ważny, obserwując status odpowiedzi.

Po odwołaniu tokenu odświeżania skojarzony token dostępu może również zostać unieważniony (zależnie od implementacji). Po odwołaniu tokenu dostępu skojarzony token odświeżania pozostaje ważny, chyba że zostanie jawnie odwołany.

## Konfiguracja

| Właściwość                                    | Opis                                                                                                       |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.Revocation.Enabled</code> | Ustaw wartość <code>True</code> , aby włączyć punkt końcowy unieważniania. Domyślnie: <code>False</code> . |
| <code>OAuth2Options.Revocation.URL</code>     | Ścieżka URL punktu końcowego unieważniania. Domyślnie: <code>/sgc/oauth2/revoke</code>                     |

## Zdarzenia

| Zdarzenie                             | Opis                                                                                                                                                                                                                                           |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OnOAuth2AfterRevokeToken</code> | Wywoływane po próbie unieważnienia tokenu. Dostarcza wartość tokenu, <code>token_type_hint</code> oraz parametr <code>Revoked</code> wskazujący, czy token został pomyślnie unieważniony. Przydatne do rejestrowania aktywności unieważniania. |

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);

OAuth2Server->OAuth2Options->Revocation->Enabled = true;

OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);

Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;

void __fastcall TForm1::OAuth2ServerOAuth2AfterRevokeToken(TObject *Sender,
    const UnicodeString Token, const UnicodeString TokenTypeHint, bool &Revoked)
{
    Log("Token revoked: " + Token + " (type: " + TokenTypeHint + ")");
    Revoked = true;
}
```

## Przykład żądania klienta

Klient unieważnia token, wysyłając żądanie POST do punktu końcowego unieważniania:

```
// Client-side: revoke an access token
HTTPClient->Post("https://example.com/sgc/oauth2/revoke",
    "token=eyJhbGciOi...&token_type_hint=access_token");
```

# Serwer OAuth2 | Introspekcja tokenów (RFC 7662)

## Przegląd

Introspekcja tokenów umożliwia serwerom zasobów odpytywanie [TsgcHTTP\\_OAuth2\\_Server](#) w celu określenia stanu aktywności i metadanych tokenu dostępu lub tokenu odświeżania. Jest to przydatne, gdy serwer zasobów musi zweryfikować token bez konieczności dekodowania go lokalnie.

## Punkt końcowy

Należy wysłać żądanie POST na `/sgc/oauth2/introspect` z następującymi parametrami:

| Parametr        | Opis                                                                        |
|-----------------|-----------------------------------------------------------------------------|
| token           | Token do introspekcji (wymagany).                                           |
| token_type_hint | Opcjonalna wskazówka dotycząca typu tokenu: access_token lub refresh_token. |

## Odpowiedź

Punkt końcowy introspekcji zwraca obiekt JSON z następującymi polami:

| Pole       | Opis                                                                                         |
|------------|----------------------------------------------------------------------------------------------|
| active     | Boolean. True, jeśli token jest aktualnie aktywny (ważny i nie wygasł ani nie unieważniony). |
| scope      | Zakres skojarzony z tokenem.                                                                 |
| client_id  | Identyfikator klienta aplikacji, która zażądała tokenu.                                      |
| token_type | Typ tokenu (np. Bearer).                                                                     |
| exp        | Czas wygaśnięcia tokenu (znacznik czasu Unix).                                               |
| username   | Nazwa użytkownika powiązana z tokenem (jeśli dotyczy).                                       |

Jeśli token jest nieprawidłowy, wygasł lub unieważniony, odpowiedź zawiera wyłącznie `{"active": false}`.

## Konfiguracja

| Właściwość                                       | Opis                                                                                      |
|--------------------------------------------------|-------------------------------------------------------------------------------------------|
| <code>OAuth2Options.Introspection.Enabled</code> | Ustaw na True, aby włączyć punkt końcowy introspekcji. Wartość domyślna: False.           |
| <code>OAuth2Options.Introspection.URL</code>     | Ścieżka URL punktu końcowego introspekcji. Domyślnie: <code>/sgc/oauth2/introspect</code> |

## Zdarzenia

| Zdarzenie                                 | Opis                                                                                                                                       |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OnOAuth2AfterIntrospectToken</code> | Wywoływane po żądaniu introspekcji tokena. Udostępnia wartość tokena oraz parametr <code>IsActive</code> , który można zmodyfikować w celu |

nadpisania domyślnego wyniku walidacji. Przydatne do niestandardowej logiki walidacji i audytowania.

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);

OAuth2Server->OAuth2Options->Introspection->Enabled = true;

OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);

Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;

void __fastcall TForm1::OAuth2ServerOAuth2AfterIntrospectToken(TObject *Sender,
    const UnicodeString Token, bool &IsActive)
{
    Log("Token introspected: " + Token + " Active: " + BoolToStr(IsActive));
}
```

## Przykład żądania do serwera zasobów

Serwer zasobów weryfikuje token, wysyłając żądanie POST do punktu końcowego introspekcji:

```
// Resource server: introspect a token
UnicodeString Response = HTTPClient->Post("https://example.com/sgc/oauth2/introspect",
    "token=eyJhbGciOiIuLl&token_type_hint=access_token");
// Response: {"active":true,"scope":"read write","client_id":"my-client-id",...}
```

# Serwer OAuth2 | Walidacja DPoP (RFC 9449)

## Przegląd

DPoP (Demonstrating Proof of Possession) zgodnie z [RFC 9449](#) umożliwia [TsgcHTTP\\_OAuth2\\_Server](#) wymaganie tokenów powiązanych z nadawcą. Gdy DPoP jest włączony, serwer weryfikuje proof JWT DPoP, wiąże tokeny z odciskiem JWK klienta i zwraca token\_type: DPoP zamiast Bearer.

Zapewnia to, że tokeny dostępu mogą być używane wyłącznie przez klienta, który je pierwotnie uzyskał, zapobiegając kradzieży i atakom powtórzenia tokenu.

## Jak to działa

1. Klient dołącza nagłówek DPoP z podpisanym dowodem JWT w żądaniach tokenu.
2. Serwer weryfikuje dowodowy JWT:
  - typ musi mieć wartość dpop+jwt
  - Algorytm (alg) musi być asymetryczny (np. ES256, RS256)
  - htm (metoda HTTP) i htu (URI HTTP) muszą być zgodne z żądaniem
  - iat (wydano o) musi być aktualny (sprawdzenie świeżości)
  - Podpis jest weryfikowany na podstawie JWK osadzonego w nagłówku dowodu
3. Serwer wyodrębnia odcisk palca JWK i wiąże go z wydanym tokenem.
4. Odpowiedź tokena zawiera token\_type: DPoP zamiast Bearer.
5. Przy kolejnych żądaniach zasobów serwer weryfikuje dowód DPoP, w tym roszczenie ath (hasz tokenu dostępu), aby upewnić się, że dowód jest powiązany z właściwym tokenem.

## Konfiguracja

| Właściwość                      | Opis                                                                                                |
|---------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>OAuth2Options.DPoP</code> | Wartość logiczna. Ustawienie na True włącza walidację tokenów powiązanych z DPoP. Domyślnie: False. |

## Weryfikacja podpisu

Serwer odtwarza publiczny klucz EC lub RSA z JWK zawartego w nagłówku dowodu DPoP i weryfikuje podpis JWT. Akceptowane są wyłącznie algorytmy asymetryczne (ES256, ES384, ES512, RS256, RS384, RS512, PS256, PS384, PS512). Algorytmy symetryczne (HS256 itp.) są odrzucane.

## Obsługa nonce

Serwer może dołączyć nagłówek DPoP-Nonce w odpowiedziach na tokeny. Gdy jest obecny, klient musi uwzględnić wartość nonce w polu nonce kolejnych dowodów DPoP JWT. Zapewnia to dodatkową warstwę ochrony przed atakami powtórkowymi.

## Zdarzenia

| Zdarzenie | Opis |
|-----------|------|
|-----------|------|

Wywoływane po zakończeniu standardowej walidacji DPoP. Umożliwia nie-  
OnOAuth2ValidateDPoP standardową logikę walidacji, np. sprawdzanie dodatkowych twierdzeń lub eg-  
zekwowanie polityki. Ustaw valid na False, aby odrzucić dowód.

## Przykład

```
TsgcHTTP_OAuth2_Server *OAuth2Server = new TsgcHTTP_OAuth2_Server(this);
OAuth2Server->OAuth2Options->DPoP = true;
OAuth2Server->Apps->AddApp("MyApp", "http://127.0.0.1:8080",
    "my-client-id", "my-client-secret", 3600, true,
    TsgcOAuth2GrantTypes() << auth2Code);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2 = OAuth2Server;
void __fastcall TForm1::OAuth2ServerOAuth2ValidateDPoP(TObject *Sender,
    const UnicodeString DPoPProof, const UnicodeString AccessToken,
    const UnicodeString Method, const UnicodeString URL, bool &Valid)
{
    Log("DPoP proof validated for: " + Method + " " + URL);
    Valid = true;
}
```

## Odrzucenie nieprawidłowego dowodu

Gdy weryfikacja dowodu DPoP nie powiedzie się, serwer zwraca HTTP 400 z odpowiedzią błędu:

```
{"error": "invalid_dpop_proof", "error_description": "..."} 
```

Typowe przyczyny odrzucenia:

- Brakujący lub nieprawidłowo sformułowany nagłówek DPoP.
- Nieprawidłowy podpis JWT.
- Użyto algorytmu symetrycznego (dozwolone są wyłącznie algorytmy asymetryczne).
- Niezgodne wartości pola htm lub htu.
- Wygasłe lub przyszłe roszczenie iat.
- Niezgodność ath (skrót tokenu dostępu) w żądaniach zasobów.
- Nieprawidłowy lub brakujący nonce, gdy serwer go wymaga.



# TsgcHTTP\_OAuth2\_Server\_Provider

Ten komponent umożliwia integrację zewnętrznych dostawców OAuth2 (takich jak Azure AD, Google, Facebook).

## Wprowadzenie

Ten komponent umożliwia integrację zewnętrznych dostawców OAuth2 (takich jak Azure AD, Google, Facebook...) z komponentem serwera (np. serwerem HTTP), dzięki czemu użytkownik może zalogować się przy użyciu poświadczeń Azure AD, a w przypadku pomyślnego uwierzytelnienia serwer HTTP może udzielić dostępu do chronionych zasobów.

Komponenty serwera mają właściwość `Authorization.OAuth.OAuth2Provider`, do której można przypisać instancję `TsgcHTTP_OAuth2_Server_Provider`. Jeśli uwierzytelnianie jest włączone i właściwość `OAuth2Provider` jest dołączona do komponentu serwera `OAuth2 Provider`, żądania WebSocket i HTTP wymagają pliku cookie lub tokenu Bearer, aby zostały przetworzone; w przeciwnym razie połączenie zostanie automatycznie zamknięte.

```
TsgcHTTP_OAuth2_Server_Provider *OAuth2Provider = new TsgcHTTP_OAuth2_Server_Provider(this);
Server->Authentication->Enabled = true;
Server->Authentication->OAuth->OAuth2Provider = OAuth2Provider;
```

## Zarejestruj dostawcę OAuth2

Przed uruchomieniem serwera należy skonfigurować dostawców OAuth2, których serwer będzie używał do uwierzytelniania. Do konfiguracji dostawców OAuth2 służy metoda **RegisterProvider**, która przyjmuje następujące parametry:

- **Name:** nazwa dostawcy — może być dowolna i służy wyłącznie do identyfikacji dostawcy w późniejszym czasie.
- **ClientId:** publiczny identyfikator klienta; wartość ta jest dostarczana przez dostawcę OAuth2.
- **ClientSecret:** to prywatny sekret klienta (musi być zachowany w poufności); wartość ta jest dostarczana przez dostawcę OAuth2.
- **AuthorizeURL:** adres URL, pod który klient OAuth2 przekieruje w celu logowania (połączenie korzysta z przeglądarki internetowej).
- **TokenURL:** jest adresem URL, którego serwer użyje do walidacji tokenu dostarczonego po pomyślnej autoryzacji (połączenie odbywa się między serwerami).
- **Scope:** jest wartością zakresu lub zakresów.
- **URL:** adres URL serwera HTTP, który będzie używany do przekierowania na adres URL autoryzacji.
- **CallbackURL:** adres URL skonfigurowany u dostawcy OAuth2, który przetworzy odpowiedź wysłaną z serwera OAuth2 po pomyślnej autoryzacji.

**Przykład:** aby skonfigurować Azure AD, wymagany jest tenant-id, który jest dodawany do adresów URL OAuth2, a także `ClientId`, `ClientSecret`, `Scope` i `CallbackURL`.

```
RegisterProvider(
'azure',
'90945b8d-f6b7-4b97-b2bd-21c3c90b5f3x',
```

```
'PN67Q~5m06c~X_GMyMf9zMntmm5l2dt~3jVq',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/authorize',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/token',  
'user.read',  
'/login',  
'https://localhost/callback'  
);
```

Aby usunąć istniejącego dostawcę, należy użyć metody **UnRegisterProvider**.

## Dokumentacja

- Właściwości — 2 właściwości dostępne w komponencie.
- Metody — 15 metod publicznych.
- Zdarzenia — 4 zdarzenia do obsługi.

## Przewodniki

- **Szybki start**
  - [Dostawca OAuth2 Azure AD](#)
- **Uwierzytelnianie**
  - [Prywatne punkty końcowe dostawcy OAuth2](#)
  - [Uwierzytelnianie dostawcy OAuth2](#)
  - [Żądania dostawcy OAuth2](#)

# Dostawca OAuth2 | Azure AD

Azure AD używa następujących adresów URL autoryzacji OAuth2

**Authorization:** `https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/authorize`

**Token:** `https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token`

Wartość **<tenant-id>** należy zastąpić własnymi wartościami.

Podczas tworzenia konfiguracji OAuth2 należy skonfigurować adres URL wywołania zwrotnego serwera. Ten adres URL będzie używany przez Azure do wysyłania odpowiedzi do serwera po pomyślnej autoryzacji.

**Przykład:** znajdź poniżej prosty przykład rejestracji dostawcy Azure AD.

## Wartości dostarczone przez Azure AD

ClientId: 90945b8d-f6b7-4b97-b2bd-21c3c90b5f3x

ClientSecret: PN67Q~5m06c~X\_GMyMf9zMntmm5l2dt~3jVq

tenant: a0ca2055-5dd1-467f-bf13-291f6fd715c6

scope: user.read

CallbackURL: `https://localhost/callback`

## Jak zarejestrować Azure AD

```
RegisterProvider(  
'azure',  
'90945b8d-f6b7-4b97-b2bd-21c3c90b5f3x',  
'PN67Q~5m06c~X_GMyMf9zMntmm5l2dt~3jVq',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/authorize',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/token',  
'user.read',  
'/login',
```

```
'https://localhost/callback'  
);
```

## Dostawca OAuth2 | Prywatne punkty końcowe

Za każdym razem, gdy serwer odbiera żądanie HTTP, wywoływane jest zdarzenie **OnOAuth2IsPrivateEndpoint** w celu sprawdzenia, czy punkt końcowy jest prywatny. Domyślnie nie jest prywatny.

```
void OnOAuth2IsPrivateEndpoint(TObject *Sender, const string aEndpoint, ref bool IsPrivate)
{
    if (aEndpoint == "/private")
    {
        IsPrivate = True;
    }
}
```

## Dostawca OAuth2 | Uwierzytelnianie

Komponent serwera dostawcy OAuth2 umożliwia uwierzytelnianie przy użyciu zewnętrznego dostawcy OAuth2 (np. Azure AD lub Google) w celu uzyskania dostępu do chronionych zasobów serwera. **Przykład:** serwer HTTP można skonfigurować tak, aby umożliwiał użytkownikom logowanie się przy użyciu poświadczeń Azure; po pomyślnym zalogowaniu użytkownicy będą mieli dostęp do chronionych zasobów serwera.

Proces uwierzytelniania odbywa się po stronie serwera, a tokeny OAuth2 nie są udostępniane klientom. Oznacza to, że gdy użytkownik loguje się na przykład za pomocą Azure i uwierzytelnianie się powiedzie, Azure zwraca token dostępu umożliwiający wysyłanie żądań do serwera Azure w celu uzyskania informacji o profilu użytkownika, e-mailach itp. (w zależności od zakresu). Ten token dostępu **NIE JEST UDOŚTĘPNIANY** klientowi (np. przeglądarce internetowej). Zamiast zwracać token dostępu klientowi, serwer tworzy losowe ID powiązane wewnętrznie z tokenem dostępu. Za każdym razem, gdy klient (przeglądarka) chce wywołać serwer OAuth2, używa tego publicznego ID, a serwer na jego podstawie pobiera token dostępu OAuth2 do obsługi żądań HTTP.

Poniżej przedstawiono przykład działania uwierzytelniania OAuth2. W przykładzie użyto konfiguracji Azure AD opisanej pod następującym linkiem: [OAuth2 Provider Azure AD](#).

### Uruchom serwer

Serwer rozpoczyna nasłuchiwanie na localhost na porcie 443. Serwer HTTP sgcWebSockets jest powiązany z komponentem dostawcy serwera OAuth2, a właściwość Authentication jest włączona.

Przed uruchomieniem serwera dostawca Azure OAuth2 jest rejestrowany przy użyciu następującego wywołania metody.

```
RegisterProvider(  
'azure',  
'90945b8d-f6b7-4b97-b2bd-21c3c90b5f3x',  
'PN67Q~5m06c~X_GMyMf9zMntmm5l2dt~3jVq',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/authorize',  
'https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/token',  
'user.read',  
'/login',  
'https://localhost/callback'  
);
```

### Logowania użytkowników

Użytkownik otwiera nową przeglądarkę internetową i przechodzi do punktu końcowego '/login'.

Serwer wykrywa, że punkt końcowy „/login” jest używany do logowania za pośrednictwem dostawcy Azure i przekierowuje do

<https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/authorize>

I rozpoczyna się przepływ uwierzytelniania OAuth2.

## **Uwierzytelnianie OAuth2**

Użytkownik jest przekierowywany do punktu końcowego uwierzytelniania serwera OAuth2, gdzie musi zalogować się przy użyciu swoich danych uwierzytelniających i zaakceptować warunki aplikacji OAuth2.

Jeśli autoryzacja się powiedzie, Azure AD wysyła kod na podany adres URL.

<https://localhost/callback>

## **Walidacja kodu OAuth2**

Serwer otrzymał teraz kod z Azure i nawiąże wewnętrzne połączenie z Azure (od serwera do serwera) w celu weryfikacji poprawności tokenu (co zapobiega próbom włamania do serwera).

Serwer łączy się z

<https://login.microsoftonline.com/a0ca2055-5dd1-467f-bf13-291f6fd715c6/oauth2/v2.0/token>

Po przekazaniu parametrów, takich jak odebrany kod i klucz tajny klienta, jeśli walidacja zakończy się sukcesem, Azure zwraca token dostępu, który można wykorzystać do uzyskania dostępu do chronionych zasobów Azure, takich jak odczyt profilu, adresu e-mail itp.

## **Pomyślny token dostępu**

Gdy serwer otrzyma prawidłowy AccessToken, wywoływane jest zdarzenie **OnOAuth2ProviderTokenValid**, w którym można skonfigurować sposób przechowywania AccessToken (jeśli ma być przechowywany), uzyskując dostęp do klasy parametrów TsgcHTTPOAuth2ProviderToken.

**AccessToken:** token OAuth2 zwrócony przez Azure.

**ID:** publiczny identyfikator przechowywany jako ciasteczko.

W tym zdarzeniu można skonfigurować działania po pomyślnym uwierzytelnieniu — na przykład, jeśli wymagane jest przekierowanie użytkownika pod prywatny adres URL, należy użyć poniższego kodu:

```
Response.Redirect.URL := 'https://localhost/private';
```

## Wysyłanie żądań do Azure

Teraz można wysyłać żądania do serwera Azure, używając publicznego identyfikatora przechowywanego jako plik cookie.

Przykład: aby odczytać dane profilu, należy użyć następującej metody.

```
Get('ID', 'https://graph.microsoft.com/v1.0/me');
```

Gdzie ID jest publicznym identyfikatorem.



## Dostawca OAuth2 | Żądania

Po pomyślnym uwierzytelnieniu można wysyłać żądania do serwera chronionego OAuth2 przy użyciu publicznego tokenu ID przechowywanego jako plik cookie.

Składnik serwera dostawcy OAuth2 udostępnia kilka metod do wysyłania żądań HTTP: GET, POST, DELETE...

Token można przekazać jako parametr lub przekazać klasę RequestInfo, jeśli używane są komponenty serwera Indy.

```
void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo, TidHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/private"
    {
        // return OAuth2 profile data
        AResponseInfo->ContentText = OAuth2Provider->Get(ARequestInfo, "https://graph.microsoft.com/v1.0/me");
        AResponseInfo->ContentType = "application/json";
        AResponseInfo->ResponseNo = 200;
    }
    else
    {
        AResponseInfo->ResponseNo = 404;
    }
}
```

# HTTP | JWT

JWT (JSON Web Token) zazwyczaj składa się z nagłówka + ładunku + podpisu.

## Nagłówek

Zawiera informacje o metadanych dotyczące JWT.

- **alg**: algorytm użyty do podpisania tokenu.
- **typ**: typ tokenu, zawsze JWT.

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Można znaleźć więcej nagłówków, jednak poprzednie będą zawsze obecne.

## Ładunek

Ładunek zawiera roszczenia JWT. Standardowe nagłówki są następujące:

- **iss**: wystawca, podmiot generujący i wystawiający JWT.
- **sub**: podmiot, encja identyfikowana przez ten token.
- **aud**: odbiorca, docelowa grupa odbiorców tego JWT.
- **exp**: czas wygaśnięcia — znacznik czasu w formacie UNIX, po którym token nie powinien być akceptowany.
- **iat**: issued at — określa datę wystawienia tokenu.

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

## Podpis

Podpis jest tworzony przy użyciu zakodowanego nagłówka, zakodowanego ładunku, sekretu i algorytmu kryptograficznego.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

## Obsługiwane algorytmy

Następujące algorytmy są obsługiwane zarówno przez składniki JWT klienta, jak i serwera.

- HS256
- HS384
- HS512
- RS256
- RS384
- RS512
- ES256
- ES384
- ES512

Wymagane są biblioteki **OpenSSL** do podpisywania i weryfikacji JWT.

## Komponenty

- **TsgcHTTP\_JWT\_Client:** Klient JWT umożliwiający kodowanie i podpisywanie JWT oraz wysyłanie go jako nagłówka Authorization w protokołach **HTTP** i **WebSocket**.
- **TsgcHTTP\_JWT\_Server:** Serwer JWT umożliwiający dekodowanie i walidację JWT odbieranych jako nagłówek autoryzacji w protokołach **HTTP** i **WebSocket**.

*\* Składniki JWT wymagają co najmniej wersji Indy 10.6.0.5169 lub sgcWebSockets Enterprise Edition.*

# TsgcHTTP\_JWT\_Client

Skonfiguruj, które biblioteki openSSL będą używane podczas korzystania z klienta JWT.

## Wprowadzenie

Komponent TsgcHTTP\_JWT\_Client umożliwia kodowanie i podpisywanie tokenów JWT dołączonych do [klienta WebSocket](#) lub [klienta HTTP/2](#); token będzie automatycznie wysyłany jako nagłówek Authorization Bearer Token.

## Konfiguracja

Wartości JWT można skonfigurować we właściwościach **JWTOptions**. Dostępne są 2 główne właściwości: **Header** i **Payload**; należy ustawić wartości dla każdej wymaganej właściwości.

Jeśli podpis jest szyfrowany za pomocą klucza prywatnego (algorytmy RS i ES), ustaw wartość we właściwości **PrivateKey** algorytmu.

Jeśli Podpis jest szyfrowany przy użyciu Sekretu (algorytmy HS), należy ustawić wartość we właściwości **Secret** Algorytmu.

## Opcje OpenSSL

Skonfiguruj, które biblioteki openSSL będą używane podczas korzystania z klienta JWT.

**OpenSSL\_Options:** konfiguracja bibliotek OpenSSL.

**APIVersion:** pozwala określić, które API OpenSSL będzie używane.

**oslpAPI\_1\_0:** używa API 1.0 OpenSSL; jest to najnowsza wersja obsługiwana przez Indy

**oslpAPI\_1\_1:** używa API 1.1 OpenSSL, wymaga naszej niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**oslpAPI\_3\_0:** używa API 3.0 OpenSSL, wymaga naszej niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek openSSL

**oslpNone:** jest to wartość domyślna; biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanej ścieżce.

**oslpDefaultFolder:** automatycznie ustawia ścieżkę openSSL, w której powinny znajdować się biblioteki dla wszystkich osobowości IDE.

**oslpCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.

**LibPathCustom:** gdy LibPath = oslpCustomFolder, należy podać pełną ścieżkę do folderu, w którym znajdują się biblioteki openssl.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**oslsSymLinksDefault:** domyślnie jest włączone, z wyjątkiem platformy OSX64 (po MacOS Monterey próba załadowania biblioteki bez wersji kończy się błędem).

**oslsSymLinksLoadFirst:** Załaduj dowiązania symboliczne przed próbą załadowania bibliotek wersji.

**oslsSymLinksLoad:** Ładuj dowiązania symboliczne po próbie załadowania bibliotek wersji.

**oslsSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

## Nagłówki niestandardowe

Właściwości Header i Payload zawierają najczęściej używane nagłówki służące do generowania JWT, ale można dodać kolejne nagłówki, wywołując metodę **AddKeyValue** i przekazując klucz oraz wartość jako parametry.

**Przykład:** jeśli chcesz dodać nowy rekord do nagłówka JWT ze swoją nazwą, użyj poniższej metody

```
Header->AddKeyValue("name", "John Smith");
```

Po skonfigurowaniu właściwości, aby wygenerować JWT, wystarczy wywołać metodę **Sign**, która zwróci wartość JWT.

## Klient WebSocket i JWT

[TsgcWebSocketClient](#) umożliwia użycie JWT podczas łączenia się z serwerami WebSocket — wystarczy utworzyć nowego klienta JWT i przypisać go do właściwości **Authentication.Token.JWT**.

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();  
oClient->URL = "ws://www.esegece.com:2052";
```

```
TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();  
oJWT->JWTOptions->Header.alg = jwtRS256;  
oJWT->JWTOptions->Payload.sub = "1234567890";
```

```
oJWT->JWTOptions->Payload.iat = 1516239022;
```

```
oClient->Authentication->Enabled = true;  
oClient->Authentication->URL->Enabled = false;  
oClient->Authentication->Token->Enabled = true;  
oClient->Authentication->Token->JWT = oJWT;  
oClient->Active = true;
```

## Klienci HTTP i JWT

[TsgcHTTP2Client](#) oraz [TsgcHTTP1Client](#) umożliwiają użycie JWT podczas łączenia się z serwerami HTTP/2 — wystarczy utworzyć nowy klient JWT i przypisać go do właściwości **Authentication.Token.JWT**.

```
TsgcHTTP2Client oHTTP = new TsgcHTTP2Client();
```

```
TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client();  
oJWT->JWTOptions->Header->alg = jwtRS256;  
oJWT->JWTOptions->Payload->sub = "1234567890";  
oJWT->JWTOptions->Payload->iat = 1516239022;
```

```
oHTTP->Authentication->Token->JWT = oJWT;  
oHTTP->Get("https://your.api.com");
```

## Wygaśnięcie

Token autoryzacyjny może być **tworzony na nowo przy każdym** wysyłaniu żądania HTTP przy użyciu klienta HTTP lub może być **ponownie używany wielokrotnie do momentu wygaśnięcia**.

**Przykład:** wywołanie Apple APNs przy użyciu tokenów wymaga, aby token był ponownie używany przez co najmniej 20 minut i maksymalnie 1 godzinę. Należy użyć właściwości **RefreshTokenAfter**, aby ustawić liczbę sekund, po których token wygaśnie, np. po 30 minutach.

```
RefreshTokenAfter = 60 * 40.
```

## Utwórz sygnaturę JWT

Możliwe jest **ręczne tworzenie podpisów JWT** do stosowania w aplikacjach, które nie korzystają z protokołu WebSocket ani HTTP, lub gdy używane są komponenty firm trzecich i potrzebny jest wyłącznie token JWT.

Aby uzyskać podpis JWT, należy utworzyć nową instancję klienta JWT i ręcznie wypełnić właściwości. Po ustawieniu wszystkich właściwości należy wywołać metodę **Sign**, która zwróci token JWT.

```
TsgcHTTP_JWT_Client oJWT = new TsgcHTTP_JWT_Client(this);
// ... header
oJWT->JWTOptions->Header->alg = jwtHS256;
oJWT->JWTOptions->Algorithms->HS->Secret = "79F66F1E-E998-436B-8A0A-3E5DEFA4FD9E";
// ... payload
oJWT->JWTOptions->Payload->jti = "9B66FB94-B761-42B1-A1AF-3C44233DBE87";
oJWT->JWTOptions->Payload->iat = 1630925658;
oJWT->JWTOptions->Payload->iss = "2886EC7547B7BA6A9009";
oJWT->JWTOptions->Payload->exp = 1630933158;
// ... custom payload values
oJWT->JWTOptions->Payload->ClearKeyValues;
oJWT->JWTOptions->Payload->AddKeyValue("origin", "www->yourwebsite->com");
oJWT->JWTOptions->Payload->AddKeyValue("ip", "69.39.46.178");
// ... get JWT Token
ShowMessage(oJWT->Sign());
```

## Dokumentacja

- Właściwości — 2 właściwości dostępne na komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 0 zdarzeń do obsługi.

# TsgcHTTP\_JWT\_Server

Komponent TsgcHTTP\_JWT\_Server.

## Wprowadzenie

Komponent TsgcHTTP\_JWT\_Server umożliwia **dekodowanie** i **walidację** tokenów JWT odbieranych w **nagłówku uzgadniania WebSocket** przy korzystaniu z protokołu WebSocket lub jako **nagłówek HTTP** przy korzystaniu z protokołu HTTP.

## Konfiguracja

W właściwości **JWTOptions** komponentu można skonfigurować następujące właściwości:

Jeśli podpis jest weryfikowany przy użyciu klucza publicznego (algorytmy RS i ES), należy ustawić wartość we właściwości **PublicKey** algorytmu.

Jeśli podpis jest walidowany przy użyciu klucza tajnego (algorytmy HS), należy ustawić wartość we właściwości **Secret** algorytmu.

Aby sprawdzić tokeny JWT, wystarczy **dołączyć instancję TsgcHTTP\_JWT\_Server** do właściwości **Authentication.JWT.JWT** serwera WebSocket/HTTP.

```
TsgcWebSocketHTTPServer oServer = new TsgcWebSocketHTTPServer();
oServer->Port = 80;
TsgcHTTP_JWT_Server oJWT = new TsgcHTTP_JWT_Server();
oJWT->JWTOptions->Algorithms->RS->PublicKey->Text = "public key here";
oServer->Authorization->Enabled = true;
oServer->Authorization->JWT->JWT = oJWT;
oServer->Active = true;
```

**Checks** właściwość umożliwia włączenie pewnych sprawdzeń w ładunku JWT; domyślnie weryfikuje poprawność dat wystawienia.

## Dokumentacja

- Właściwości — 2 dostępne właściwości komponentu.
- Metody — 3 metody publiczne.
- Zdarzenia — 7 zdarzeń do obsługi.



# WebAuthn

**WebAuthn** (Web Authentication) to standard sieciowy opracowany przez World Wide Web Consortium (W3C) i sojusz FIDO, umożliwiający **bezpieczne, uwierzytelnianie bez hasła w sieci web**. Jest częścią szerszego frameworku FIDO2 i ma na celu **zmniejszenie zależności od tradycyjnych haseł**, które są często podatne na phishing, credential stuffing i inne ataki.

U podstaw **WebAuthn umożliwia użytkownikom uwierzytelnianie przy użyciu kryptografii klucza publicznego**. Zamiast nazwy użytkownika i hasła, użytkownicy rejestrują **unikalną parę kluczy publiczny-prywatny w aplikacji internetowej** (Relying Party). Klucz prywatny jest bezpiecznie przechowywany na uwierzytelniaczu — takim jak sprzętowy klucz bezpieczeństwa, smartfon lub wbudowane urządzenie biometryczne — podczas gdy klucz publiczny jest przechowywany na serwerze.

Podczas uwierzytelniania serwer wystawia wyzwanie, które musi zostać podpisane prywatnym kluczem użytkownika. Podpisane wyzwanie jest zwracane i weryfikowane przy użyciu przechowywanego klucza publicznego, zapewniając integralność i autentyczność odpowiedzi. Takie podejście zapobiega przechwyceniu lub ponownemu użyciu danych uwierzytelniających.

WebAuthn obsługuje szeroki zakres urządzeń uwierzytelniających, co czyni go elastycznym rozwiązaniem zarówno dla deweloperów, jak i użytkowników. Umożliwia także uwierzytelnianie wieloskładnikowe (MFA) w połączeniu z innymi czynnikami, takimi jak kody PIN lub dane biometryczne, znacząco poprawiając bezpieczeństwo bez uszczerbku dla użyteczności.

## Serwer

- **TsgcWSAPIServer\_WebAuthn**: Komponent zapewnia proste, lecz wydajne rozwiązanie do implementacji serwera Relying Party WebAuthn, umożliwiające uwierzytelnianie bez hasła w aplikacji internetowej. Serwer przeszedł pełny **test zgodności FIDO** przy użyciu narzędzia do samodzielnej walidacji zgodności.

## Klient

- **Klient JavaScript**: pełni rolę pomostu między użytkownikiem, przeglądarką i uwierzytelniaczem podczas rejestracji i uwierzytelniania

# TsgcWSAPIServer\_WebAuthn

Wyłącznie OpenSSL 3.

## Wprowadzenie

Komponent **TsgcWSAPIServer\_WebAuthn** zapewnia proste, lecz wydajne rozwiązanie do implementacji serwera Relying Party WebAuthn, umożliwiając uwierzytelnianie bez hasła w aplikacji internetowej. Aplikacja WebAuthn składa się z serwera WebAuthn obsługującego rejestrację i uwierzytelnianie po stronie serwera oraz aplikacji klienckiej, która zazwyczaj jest aplikacją JavaScript.

WebAuthn wymaga stosowania bezpiecznych połączeń (SSL/TLS), dlatego biblioteki [OpenSSL](#) muszą zostać wdrożone i skonfigurowane na serwerze.

**Obsługiwane jest wyłącznie API OpenSSL w wersji 3.0.0 lub nowszej**, starsze wersje OpenSSL mogą nie działać.

## Konfiguracja

Komponent **TsgcWSAPIServer\_WebAuthn** musi być dołączony do serwera HTTP, [TsgcWebSocketHTTP\\_Server](#) lub [TsgcWebSocketServer\\_HTTPAPI](#), za pomocą właściwości `Server`. Możliwa jest konfiguracja punktów końcowych serwera obsługujących opcje rejestracji i uwierzytelniania oraz opcje WebAuthn, takich jak obsługiwane algorytmy, źródła i inne.

### Opcje punktów końcowych

W tym miejscu można skonfigurować punkty końcowe serwera, które będą obsługiwać żądania HTTP/JavaScript w celu użycia WebAuthn jako uwierzytelniacza. Komponent jest już skonfigurowany z domyślnymi punktami końcowymi, ale można je wszystkie zmienić według własnych potrzeb.

- **AuthenticationOptions:** domyślnie `/sgcWebAuthn/Authentication/Options`
- **AuthenticationVerify:** domyślnie `/sgcWebAuthn/Authentication/Verify`
- **RegistrationOptions:** domyślnie jest `/sgcWebAuthn/Registration/Options`
- **RegistrationVerify:** domyślnie `/sgcWebAuthn/Registration/Verify`
- **Webauthn:** zawiera bibliotekę JavaScript używaną domyślnie. Tę właściwość można wyłączyć i użyć własnej biblioteki WebAuthn.
- **Test:** domyślnie wyłączone, należy używać wyłącznie do testowania funkcjonalności WebAuthn.

**Przykład:** jeśli serwer nasłuchuje pod domeną `www.test.com`, żądanie opcji uwierzytelniania domyślnie będzie kierowane na adres `http://www.test.com/sgcWebAuthn/Authentication/Options`

## Opcje WebAuthn

W tej właściwości można skonfigurować główne opcje komponentu serwera WebAuthn.

- **RelyingParty:** właściwość **obowiązkowa**, w której należy zdefiniować **nazwę DNS serwera**. Przykład: jeśli serwer działa pod domeną `www.test.com`, należy ustawić tę właściwość na `"www.test.com"`.

WebAuthn używa źródeł (origins) do wymuszania ograniczeń zasady tego samego źródła, co jest niezbędne do zapobiegania phishingowi i atakom cross-site. Podczas procesów rejestracji i uwierzytelniania WebAuthn źródło jest ściśle weryfikowane przez przeglądarkę i uwierzytelniacz.

- **Origins:** Jeśli żądania mogą pochodzić z różnych źródeł, użyj właściwości `Origin`, aby ustawić dodatkowe źródła. Przykład: jeśli żądania mogą pochodzić z `login.test.com` i `www.test.co.uk`, skonfiguruj właściwość `Origins` z wartościami: `https://login.test.com` i `https://www.test.co.uk`
- **TopOrigins:** Zazwyczaj WebAuthn opiera się na źródle ramki wywołującej (tj. tej, która wywołuje `navigator.credentials.create()` lub `navigator.credentials.get()`). Jednak strony internetowe mogą być osadzone w ramach `iframe`, które mogą pochodzić z innego źródła niż strona najwyższego poziomu. Stwarza to potencjał do nadużyć lub ataków typu clickjacking. Aby temu zaradzić, specyfikacja WebAuthn Level 2 wprowadza `TopOrigin`, gdzie można zdefiniować `TopOrigins`.

W WebAuthn parametr `crossOrigin` to wartość logiczna wskazująca, czy operacja WebAuthn jest wykonywana z kontekstu cross-origin, na przykład z elementu `iframe` osadzonego z innego źródła niż kontekst przeglądania najwyższego poziomu.

Ten parametr został wprowadzony, aby pomóc przeglądarkom i urządzeniom uwierzytelniającym bezpiecznie obsługiwać żądania uwierzytelniania w środowiskach osadzonych, co jest częstym scenariuszem w nowoczesnych aplikacjach internetowych.

- **AllowCrossOrigins:** jeśli ustawione na `true`, wskazuje, że żądania z ramki `iframe` z innego źródła (np. `iframe` pod adresem `https://auth.example.com` osadzony na stronie pod adresem `https://app.example.org`) są dozwolone. Domyślnie jest wyłączone.

WebAuthn obsługuje wiele **algorytmów kryptograficznych** do generowania i weryfikacji poświadczeń klucza publicznego. Algorytmy te są stosowane podczas rejestracji poświadczeń (za pomocą `navigator.credentials.create()`) i uwierzytelniania (za pomocą `navigator.credentials.get()`), zapewniając

bezpieczne podpisywanie i walidację wyzwań przy użyciu asymetrycznych par kluczy. Serwer jest domyślnie skonfigurowany z algorytmami **ES256** i **RS256**, które są najczęściej stosowane. W dowolnym momencie można zmienić obsługiwane algorytmy za pomocą właściwości **Algorithms**. Obsługiwane są następujące algorytmy:

- ES256
- ES384
- ES512
- RS256
- RS384
- RS512
- PS256
- PS384
- PS512
- RS1
- EdDSA

W WebAuthn **atestacja jest opcjonalnym mechanizmem** umożliwiającym uwierzytelniaczowi (np. urządzeniu lub kluczowi bezpieczeństwa) dostarczenie informacji o producencie, modelu i charakterystykach bezpieczeństwa podczas tworzenia poświadczeń. Informacje te pomagają Relying Party (RP) zdecydować, czy ufać uwierzytelniaczowi.

Różne formaty atestacji definiują sposób strukturyzowania i weryfikowania tych danych. Trzy najczęściej stosowane formaty to android-key, packed oraz inne, takie jak fido-u2f, apple lub none. Domyślnie wszystkie formaty atestacji są włączone. Poniżej znajduje się lista obsługiwanych formatów atestacji:

- **NoneAttestation:** w tym przypadku nie są zwracane żadne dane atestacyjne. Daje pierwszeństwo prywatności użytkownika poprzez unikanie ujawniania identyfikatorów urządzenia. Powszechnie stosowane w aplikacjach, które nie potrzebują informacji o pochodzeniu urządzenia.
- **PackedAttestation:** to elastyczny, kompaktowy format używany przez wiele uwierzytelniających. Uwierzytelniający zwraca certyfikat atestacji i podpis. Może być: **Pełna atestacja:** podpisana kluczem i certyfikatem dostarczonym przez producenta, lub **Samopodpisana atestacja:** podpisana za pomocą prywatnego klucza poświadczenia. Najszerzej stosowana na różnych platformach (np. YubiKey, Windows Hello).
- **TPMAttestation:** Używane przez urządzenia z modułem Trusted Platform Module (TPM). Potwierdzenie jest podpisywane kluczami z TPM i zawiera łańcuch certyfikatów. Stosowane w komputerach stacjonarnych i laptopach klasy korporacyjnej z układami TPM (np. maszyny z systemem Windows).

- **AndroidKeyAttestation:** Używane przez urządzenia z systemem Android z magazynem kluczy Android Keystore. Klucz jest generowany sprzętowo, a atestacja zawiera informacje podpisane przez łańcuch certyfikatów wydany przez producenta urządzenia. Używane przez telefony z systemem Android z magazynami kluczy wspieranymi sprzętowo (TEE lub StrongBox).
- **AppleAttestation:** Używane przez urządzenia uwierzytelniające platformy Apple, takie jak Touch ID i Face ID. Atestacja jest generowana przez wewnętrzne API Apple i zawiera specjalny format certyfikatu. Stosowane w przeglądarce Safari z użyciem biometrii Apple.
- **FidoU2FAttestation:** Starszy format atestacji używany przez uwierzytelniniki FIDO U2F. Zwraca certyfikat i podpis zgodny z U2F. Stosowany przez starsze klucze bezpieczeństwa (np. wczesne YubiKey) obsługujące FIDO U2F.

W API WebAuthn pole **AllowCredentials** jest **polem opcjonalnym** używanym podczas procesu uwierzytelniania (przez `navigator.credentials.get()`). **Określa listę identyfikatorów poświadczeń, które są uprawnione do uwierzytelnienia użytkownika** dla konkretnej jednostki zależnej (RP). Mechanizm ten umożliwia RP kontrolowanie, które poświadczenia są uznawane za prawidłowe przy próbie logowania. Właściwość `credentials` zawiera następujące pola:

- **AllowCredentials:** jeśli włączone (domyślnie `false`), określa listę identyfikatorów danych uwierzytelniających, które mogą uwierzytelniać użytkownika
- **ExcludeCredentials:** dla podanej nazwy użytkownika wyświetla wszystkie istniejące poświadczenia już zapisane w składniku serwera.
- **Limit:** maksymalna liczba poświadczeń, które zostaną wysłane, gdy `ExcludeCredentials` ma wartość `true`.

## Protokół WebAuthn

- **Rejestracja WebAuthn:** Serwer generuje wyzwanie i wysyła je do klienta, który używa uwierzytelniacza (np. klucza bezpieczeństwa lub urządzenia biometrycznego) do utworzenia pary kluczy. Klucz publiczny jest odsyłany i przechowywany przez serwer do przyszłego uwierzytelniania. Poniżej znajdują się dodatkowe informacje o zdarzeniach przepływu rejestracji:
  - **Żądanie rejestracji WebAuthn**
  - **Odpowiedź rejestracji WebAuthn**
  - **Wynik rejestracji WebAuthn**

- **Uwierzytelnianie WebAuthn:** Serwer wysyła wyzwanie do klienta, który podpisuje je przy użyciu wcześniej zarejestrowanego klucza prywatnego przechowywanego w urządzeniu uwierzytelniającym. Podpisana odpowiedź jest weryfikowana przez serwer przy użyciu przechowywanego klucza publicznego w celu potwierdzenia tożsamości użytkownika. Poniżej znajdują się dodatkowe informacje o zdarzeniach przepływu uwierzytelniania:
  - [Żądanie uwierzytelniania WebAuthn](#)
  - [Odpowiedź uwierzytelniania WebAuthn](#)
  - [Wynik uwierzytelniania WebAuthn](#)
- **MDS:** FIDO Alliance Metadata Service (MDS) to centralne repozytorium instrukcji metadanych używane przez strony ufające do weryfikacji poświadczenia uwierzytelniacza i potwierdzenia autentyczności modelu urządzenia.



- **Autoryzacja:** Klient może żądać od serwera tokenu Bearer podczas przepływu uwierzytelniania. Token ten może być następnie używany do otwierania nowego połączenia WebSocket lub HTTP bez konieczności logowania się za pomocą kluczy dostępu.
  - [Autoryzacja WebAuthn WebSocket](#)
  - [Autoryzacja WebAuthn HTTP](#)

## Dokumentacja

- Właściwości — 4 właściwości dostępne w składniku.
- Metody — 8 metod publicznych.
- Zdarzenia — 15 zdarzeń do obsługi.

# WebAuthn | Rejestracja

**Rejestracja WebAuthn** obejmuje **klienta** (przeglądarkę), **urządzenie uwierzytelniające** (urządzenie zabezpieczające) oraz **stronę ufającą** (RP; serwer Delphi/C++Builder TsgcWSAPIServer\_WebAuthn).

## Opcje rejestracji

Tworzenie nowych danych uwierzytelniających użytkownika zazwyczaj wymaga zainicjowania przez klienta przepływu rejestracji za pomocą nowego żądania HTTP do **punktu końcowego opcji rejestracji** skonfigurowanego na serwerze TsgcWSAPIServer\_WebAuthn. Domyślnie punkt końcowy to /sgcWebAuthn/Registration/Options; jeśli serwer nasłuchuje pod domeną <https://www.test.com>, klient powinien wysłać nowe żądanie pod adres URL <https://www.test.com/sgcWebAuthn/Registration/Options>.

Klient wysyła nowe żądanie, przekazując jako ładunek następujący JSON (używając „test” jako nazwy użytkownika)

```
{"username":"test","algorithms":[]}
```

Serwer odczytuje żądanie i zwraca odpowiedź z nowym wyzwaniem.

```
{
  "rp": {
    "name": "localhost",
    "id": "localhost"
  },
  "user": {
    "id": "36b9d6a84204487382fee62e7e67a80d",
    "name": "test",
    "displayName": "test"
  },
  "challenge":
    "6c6c468c99f24bf29a85a15b661f75f385654f97309c46bab2909c926e17ccbe",
  "pubKeyCredParams": [
    {
      "type": "public-key",
      "alg": "-7"
    },
    {
      "type": "public-key",
      "alg": "-257"
    }
  ]
}
```

```
],
"timeout": 60000,
"excludeCredentials": [],
"authenticatorSelection": {
  "residentKey": "preferred",
  "requireResidentKey": false,
  "userVerification": "preferred"
},
"attestation": "direct",
"hints": [],
"extensions": {
  "credProps": true
}
}
```

Odpowiedź zwraca następujące dane:

- **challenge**: Losowe dane binarne zakodowane w base64.
- **rp**: informacje o Relying Party (aplikacja/strona internetowa).
- **user**: Identyfikator użytkownika i nazwa wyświetlana.
- **pubKeyCredParams**: Dozwolone algorytmy (np. ES256, RS256).
- **authenticatorSelection**: Preferencje dotyczące typu uwierzytelniającego (platformowy lub między platformami, opcje weryfikacji użytkownika).
- **timeout**: Sugerowana wartość limitu czasu.
- **attestation**: Preferencja przekazywania potwierdzenia („none”, „direct”, „indirect”).

## Weryfikacja rejestracji

Teraz klient otrzymuje odpowiedź z serwera, odczytuje ją, a uwierzytelniacz zwraca dane kryptograficzne do przeglądarki klienta. Klient wysyła nowe żądanie HTTP z następującymi danymi do punktu końcowego weryfikacji rejestracji skonfigurowanego w serwerze TsgcWSAPIServer\_WebAuthn; domyślnie jest to /sgcWebAuthn/Registration/Verify

, więc jeśli serwer nasłuchuje w domenie <https://www.test.com>, klient powinien wysłać nowe żądanie pod adres URL <https://www.test.com/sgcWebAuthn/Registration/Verify>.

Authenticator odpowiada z powrotem do kodu JavaScript w przeglądarce za pomocą:

- **id**: Identyfikator poświadczeń (unikalny identyfikator pary kluczy wygenerowanej przez authenticator).
- **rawId**: Binarna postać identyfikatora poświadczeń.



- **type:** Typ poświadczenia ("public-key").
- **response:** Zawiera attestationObject i clientDataJSON.

Poniżej znajduje się przykład:

```
{
  "id": "yeA4BVRlrAfLG-KzqsL_rli4ffhuKHK8uoEkVoab065UkS82Zqlh9VFQHIYwOuOo",
  "rawId": "yeA4BVRlrAfLG-KzqsL_rli4ffhuKHK8uoEkVoab065UkS82Zqlh9VFQHIYwOuOo",
  "response": {
    "attestationObject": "o2NmbXRmcGFja2VkZ2F0dFN0....",
    "clientDataJSON": "eyJ0eXBlljoid2ViYXV0aG4uY3Jl....",
    "transports": [
      "nfc",
      "usb"
    ],
    "publicKeyAlgorithm": -7,
    "publicKey": "MFkwEwYHKoZIzj0CAQYIKoZIzj....",
    "authenticatorData": "SZYN5YgOjGh0NBcPZHZgW4_k...."
  },
  "type": "public-key",
  "clientExtensionResults": {
    "credProps": {
      "rk": true
    }
  },
  "authenticatorAttachment": "cross-platform"
}
```

Serwer odczytuje żądanie JSON od klienta, a następnie dekoduje, weryfikuje i przechowuje klucz publiczny oraz identyfikator poświadczeń.

Dekoduj attestationObject i clientDataJSON:

- Wyodrębnij klucz publiczny z attestationObject.
- Sprawdź, czy wyzwanie w clientDataJSON odpowiada oryginalnemu wysłanemu przez serwer.
- Należy zapewnić właściwą weryfikację źródła (myapp.example.com).

Przechowaj poświadczenie:

- Jeśli weryfikacja zakończy się powodzeniem, wywoływane jest zdarzenie **OnWebAuthnRegistrationSuccessful**, umożliwiające bezpieczne przechowanie identyfikatora poświadczenia i wyodrębnionego klucza publicznego w bazie danych w celu przyszłych uwierzytelnień.

- W przypadku wystąpienia błędu wywoływane jest zdarzenie **OnWebAuthnRegistrationError**, które pozwala uzyskać szczegółowe informacje o błędzie.

## Przepływ rejestracji

Więcej informacji na temat procesu rejestracji można znaleźć pod następującymi linkami:

- [Żądanie rejestracji WebAuthn](#)
- [Odpowiedź rejestracji WebAuthn](#)
- [Wynik rejestracji WebAuthn](#)

## WebAuthn — rejestracja | żądanie

To żądanie opcji rejestracji jest niezbędne do bezpiecznego zainicjowania rejestracji WebAuthn. Prosi serwer o:

- Wygeneruj wyzwanie,
- Wyszukaj lub utwórz identyfikator użytkownika,
- Zwraca prawidłowy obiekt `PublicKeyCredentialCreationOptions` (zawierający RP, użytkownika, algorytmy itp.).

Przeglądarka musi pobrać opcje tworzenia poświadczeń z serwera, aby zainicjować bezpieczną rejestrację poświadczeń WebAuthn. Zazwyczaj odbywa się to za pośrednictwem żądania HTTPS POST z przeglądarki do punktu końcowego serwera (np. `/sgcWebAuthn/Registration/Options`), który przygotowuje parametry takie jak wyzwanie, RP ID i informacje o użytkowniku.

Treść żądania zawiera dane identyfikujące użytkownika i ewentualnie preferencje konfiguracyjne. Zazwyczaj wygląda następująco:

```
{
  "username": "alice@example.com",
  "displayName": "Alice Smith",
  "authenticatorSelection": {
    "authenticatorAttachment": "platform", // opcjonalnie: platform lub
    cross-platform
    "userVerification": "preferred" // opcjonalnie: required | preferred |
    discouraged
  },
  "attestation": "none", // lub "direct", "indirect"
  "residentKey": "discouraged" // opcjonalnie
}
```

Składnik serwera WebAuthn nasłuchuje na punkcie końcowym skonfigurowanym we właściwości **EndpointOptions.RegistrationOptions** na początkowe żądanie przeglądarki w celu uzyskania `PublicKeyCredentialCreationOptions`. Gdy serwer odbiera nowe żądanie HTTP, wywoływane jest zdarzenie **OnWebAuthnRegistrationOptionsRequest**, umożliwiające dostęp do żądania wysłanego przez klienta i anulowanie żądania przez ustawienie parametru `Accept` na `False`.

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



## WebAuthn Registration | Odpowiedź

Serwer odpowiada na żądanie opcji rejestracji klienta (np. POST /sgcWebAuthn/Registration/Options) łańdunkiem JSON, który wygląda następująco (po zakodowaniu pól binarnych w base64url):

```
{
  "publicKey": {
    "rp": {
      "name": "esegece software",
      "id": "esegece.com"
    },
    "user": {
      "id": "c3ViamVjdC1pZAA", // base64url-encoded ArrayBuffer (user handle)
      "name": "webauthn@esegece.com",
      "displayName": "Delphi Developer"
    },
    "challenge": "Xz8x2K6nY3gZ...", // base64url-encoded challenge
    "pubKeyCredParams": [
      { "type": "public-key", "alg": -7 }, // ES256
      { "type": "public-key", "alg": -257 } // RS256
    ],
    "timeout": 60000,
    "attestation": "none",
    "authenticatorSelection": {
      "authenticatorAttachment": "platform",
      "userVerification": "preferred",
      "residentKey": "discouraged"
    }
  }
}
```

Poniżej znajduje się opis pól:

- **rp:** Informacje o Relying Party:
  - **name:** Przyjazna nazwa wyświetlana (np. "Example Inc.").
  - **id:** Identyfikator Relying Party (musi odpowiadać pochodzeniu witryny lub być rejestrowanym sufiksem).
- **user:** Informacje identyfikujące konto użytkownika:
  - **id:** Unikalny, stały tablica bajtów (np. UUID użytkownika).
  - **name:** Unikalny identyfikator użytkownika (nazwa użytkownika lub adres e-mail).
  - **displayName:** Wyświetlana nazwa użytkownika.
- **challenge:** Kryptograficznie losowy nonce (16–64 bajtów) zakodowany w formacie base64url.
- **pubKeyCredParams:** Algorytmy akceptowane przez stronę ufającą (RP):
  - **Najczęstsze:** -7 (ES256), -257 (RS256)

- **timeout:** Opcjonalny limit czasu operacji tworzenia (w ms).
- **attestation:** Jedna z wartości: „none”, „indirect”, „direct”.
- **authenticatorSelection:**
  - **authenticatorAttachment:** "platform" (wbudowany) lub "cross-platform" (np. YubiKey).
  - **userVerification:** "required", "preferred" lub "discouraged".
  - **residentKey:** "required", "preferred" lub "discouraged".

Przed wysłaniem odpowiedzi do klienta wywoływane jest zdarzenie **OnWebAuthnRegistrationOptionsResponse**, umożliwiające dostosowanie odpowiedzi.

```
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");
    }
}
```

## WebAuthn Rejestracja | Wynik

Celem jest weryfikacja autentyczności i integralności danych zwróconych przez klienta, upewnienie się, że poświadczenie jest powiązane z oczekiwanym użytkownikiem, oraz bezpieczna rejestracja poświadczenia klucza publicznego do przyszłego uwierzytelniania.

Serwer musi zweryfikować odpowiedź klienta zgodnie z następującymi krokami:

- Przetwarzanie odpowiedzi klienta: klient wysłał do serwera odpowiedź w następującej postaci.

```
{
  "id": "base64url-encoded credential ID",
  "rawId": "base64url-encoded ID bytes",
  "response": {
    "clientDataJSON": "base64url",
    "attestationObject": "base64url"
  },
  "type": "public-key"
}
```

- Serwer weryfikuje obiekt clientDataJSON oraz attestationObject
- Weryfikuje oświadczenie zaświadczenia (Attestation Statement), jeśli jest zdefiniowane
- Jeśli wszystkie dane są prawidłowe, poświadczenie jest przechowywane i klucz publiczny może być zaufany przy przyszłych logowaniach.

## Rejestracja zakończona pomyślnie

Jeśli odpowiedź wysłana przez klienta jest prawidłowa, wywoływane jest zdarzenie **OnWebAuthnRegistrationSuccessful**, a rekord poświadczeń można bezpiecznie zapisać w bazie danych do przyszłych walidacji logowania.

```
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();
}
```

```
}
```

## Błąd rejestracji

W przypadku wystąpienia błędu podczas walidacji odpowiedzi klienta wywoływane jest zdarzenie **OnWebAuthnRegistrationError**, a przyczyna błędu jest dostępna w parametrze `aError`.

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



# WebAuthn | Uwierzytelnianie

**Uwierzytelnianie WebAuthn** umożliwia użytkownikom logowanie się przy użyciu wcześniej zarejestrowanych poświadczeń klucza publicznego. Polega na walidacji podpisanego wyzwania przy użyciu przechowywanego klucza publicznego użytkownika z rejestracji.

## Opcje uwierzytelniania

Uwierzytelnianie wymaga, aby klient zainicjował przepływ uwierzytelniania przy użyciu nowego żądania HTTP do **punktu końcowego opcji uwierzytelniania** skonfigurowanego na serwerze TsgcWSAPIServer\_WebAuthn. Domyślnie punktem końcowym jest /sgcWebAuthn/Authentication/Options, więc jeśli serwer nasłuchuje pod adresem <https://www.test.com>, klient powinien wysłać nowe żądanie do adresu URL <https://www.test.com/sgcWebAuthn/Authentication/Options>.

Klient wysyła potwierdzenie (odpowiedź uwierzytelniania) do serwera przez POST

```
{"username":"test","user_verification":"preferred"}
```

Serwer generuje PublicKeyCredentialRequestOptions

```
{
  "challenge":
    "9d0d61edf30b45f8b88aef7087f9117716e2b7d8b0ee4460b06142f39dd0ec9f",
  "timeout": 60000,
  "rpId": "localhost",
  "allowCredentials": [
    {
      "id": "yeA4BVRlrAfLG-
      KzqsL_rli4ffhuKHK8uoEkVoab065UkS82Zqlh9VFQHIYwOuOo",
      "type": "public-key",
      "transports": [
        "nfc",
        "usb"
      ]
    }
  ],
  "userVerification": "preferred",
  "hints": [],
  "attestation": "none",
}
```

```
"attestationFormats": [],  
"extensions": {}  
}
```

## Weryfikacja uwierzytelniania

Uwierzytelnianie wymaga, aby klient zainicjował przepływ uwierzytelniania, wysyłając nowe żądanie HTTP do punktu końcowego **Authentication Verify Endpoint** skonfigurowanego na serwerze TsgcWSAPIServer\_WebAuthn. Domyślnie punkt końcowy to /sgcWebAuthn/Authentication/Verify, więc jeśli serwer nasłuchuje pod domeną https://www.test.com, klient powinien wysłać nowe żądanie pod adres https://www.test.com/sgcWebAuthn/Authentication/Verify.

Przeglądarka prosi użytkownika o użycie jego urządzenia uwierzytelniającego (np. odcisk palca, Yubi-Key). Urządzenie uwierzytelniające podpisuje wyzwanie kluczem prywatnym powiązanym z identyfikatorem poświadczenia. Zwrócone poświadczenie zawiera:

- **rawId**: Identyfikator poświadczenia (base64url).
- **response.authenticatorData**: Metadane z uwierzytelnacza (dane binarne).
- **response.clientDataJSON**: Zawiera oryginalne wyzwanie, origin i typ.
- **response.signature**: podpis obejmujący authenticatorData + skrót (hash) clientDataJSON.

Poniżej znajduje się przykład JSON żądania klienta:

```
{  
  "id": "yeA4BVRlrAfLG-  
KzqsL_rlI4ffhuKHK8uoEkVoab065UkS82Zqlh9VFQHIYwOuOo",  
  "rawId": "yeA4BVRlrAfLG-  
KzqsL_rlI4ffhuKHK8uoEkVoab065UkS82Zqlh9VFQHIYwOuOo",  
  "response": {  
    "authenticatorData":  
    "SZYN5YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2MFAAAABw",  
    "clientDataJSON": "eyJ0eXB1Ijoid2ViYXV0aG4uZ.....",  
    "signature": "MEQCIAJRqvvys8.....",  
    "userHandle": "36b9d6a84204487382fee62e7e67a80d"  
  },  
  "type": "public-key",  
  "clientExtensionResults": {},  
  "authenticatorAttachment": "cross-platform"  
}
```

Serwer odczytuje żądanie od klienta, sprawdza, czy poświadczenie jest zapisane, i weryfikuje podpis. Jeśli podpis jest prawidłowy, wywoływane jest zdarzenie **OnWebAuthnAuthenticationSuccessful**.

## Przepływ uwierzytelniania

Więcej informacji na temat procesu przepływu uwierzytelniania można znaleźć pod następującymi linkami:

- [WebAuthn Authentication Request](#)
- [Odpowiedź uwierzytelniania WebAuthn](#)
- [Wynik uwierzytelniania WebAuthn](#)

## WebAuthn Authentication | Request

Gdy użytkownik próbuje się zalogować, przeglądarka wysyła do serwera żądanie opcji uwierzytelniania (zwanym również „opcjami asercji”).

```
{  
  "username": "alice@example.com"  
}
```

To żądanie umożliwia serwerowi:

- Wygeneruj wyzwanie (bezpieczną, losową wartość).
- Wyszukuje, które poświadczenia (identyfikatory kluczy publicznych) są prawidłowe dla danego użytkownika.
- Odpowiedz parametrami kryptograficznymi, których przeglądarka użyje w `navigator.credentials.get()`.

Gdy składnik WebAuthn Server odbiera to żądanie (zazwyczaj wysyłane przez przeglądarkę jako żądanie HTTPS POST do punktu końcowego serwera, np. `/sgcWebAuthn/Authentication/Options`), wywoływane jest zdarzenie **OnWebAuthnAuthenticationOptionsRequest**, w którym można dodać poświadczenia skojarzone z nazwą użytkownika (jeśli istnieją).

```
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();  
        }  
    }  
}
```

## WebAuthn Authentication | Odpowiedź

Po tym, jak przeglądarka (klient) wyśle żądanie do serwera z nazwą użytkownika (lub podobnym identyfikatorem), serwer odpowiada opcjami uwierzytelnienia, których klient użyje do rozpoczęcia procesu uwierzytelnienia za pomocą API przeglądarki `navigator.credentials.get()`.

Ta odpowiedź dostarcza parametry i ograniczenia wymagane przez przeglądarkę do wygenerowania asercji uwierzytelniania WebAuthn przy użyciu urządzenia uwierzytelniającego użytkownika (np. klucza bezpieczeństwa, urządzenia biometrycznego).

Przykładowa odpowiedź JSON:

```
{
  "challenge": "Z3lVbWV5YXBpbmdvZG90IQ",
  "timeout": 60000,
  "rpId": "example.com",
  "allowCredentials": [
    {
      "type": "public-key",
      "id": "dXNlckNyZWRJZA",
      "transports": ["usb", "nfc", "ble"]
    }
  ],
  "userVerification": "preferred"
}
```

Poniżej znajduje się opis pól:

- **challenge:** Kryptograficzne wyzwanie (generowane losowo) w celu zapobiegania atakom powtórzeniowym. Musi być unikalne dla każdego żądania.
- **timeout:** Opcjonalnie. Czas (w ms), przez który przeglądarka powinna czekać przed anulowaniem operacji.
- **rpId:** Identyfikator strony ufającej (Relying Party) — zazwyczaj nazwa domeny.
- **allowCredentials:** Lista akceptowalnych poświadczeń (identyfikatorów poświadczeń) wcześniej zarejestrowanych przez użytkownika.
- **userVerification:** Informuje przeglądarkę, jak weryfikować użytkownika: "required", "preferred" lub "discouraged".
- **extensions (opcjonalne):** Dodatkowe możliwości lub dane żądane od uwierzytelniacza.

Przed wysłaniem odpowiedzi do klienta wywoływane jest zdarzenie **OnWebAuthnAuthenticationOptionsResponse**, umożliwiając dostosowanie odpowiedzi.

## WebAuthn Authentication | Result

Celem jest weryfikacja podpisanego potwierdzenia dostarczonego przez przeglądarkę, które dowodzi, że użytkownik posiada klucz prywatny pierwotnie zarejestrowany. W ten sposób użytkownik jest bezpiecznie logowany. Po interakcji użytkownika z urządzeniem uwierzytelniającym (np. odciskiem palca, kluczem bezpieczeństwa) przeglądarka wysyła żądanie POST do serwera z wynikiem uwierzytelniania. Poniżej znajduje się przykład w formacie JSON:

```
{
  "id": "credential-id",
  "rawId": "base64url-encoded-credential-id",
  "type": "public-key",
  "response": {
    "clientDataJSON": "base64url",
    "authenticatorData": "base64url",
    "signature": "base64url",
    "userHandle": "optional"
  }
}
```

Gdy serwer otrzyma to żądanie pod skonfigurowanym punktem końcowym (np. /sgcWebAuthn/Authentication/Verify), musi zweryfikować następujące kroki:

- Pobieranie poświadczeń z bazy danych: należy użyć identyfikatora poświadczenia, aby odnaleźć zarejestrowany klucz publiczny użytkownika i metadane (np. licznik podpisów).
- Dekodowanie i parsowanie clientDataJSON
- Weryfikacja danych uwierzytelniających
- Weryfikacja podpisu

Jeśli wszystkie weryfikacje przebiegną pomyślnie, uwierzytelnianie zakończy się powodzeniem i zostanie wywołane zdarzenie **OnWebAuthnAuthenticationSuccessful**.

Jeśli którykolwiek ze sprawdzeń zakończy się niepowodzeniem, wywoływane jest zdarzenie **OnWebAuthnAuthenticationError** z podaniem przyczyny błędu.

# WebAuthn | MDS

**Metadata Service (MDS)** to scentralizowana usługa dostarczana przez **FIDO Alliance**, która agreguje i publikuje **Metadata Statements** dotyczące urządzeń uwierzytelniających certyfikowanych w ramach programów certyfikacji FIDO. Oświadczenia te zawierają szczegółowe informacje o bezpieczeństwie, zgodności i aspektach operacyjnych urządzeń uwierzytelniających.

- Punkt końcowy usługi jest często określany jako **MDS3** (trzecia wersja protokołu Metadata Service).
- Strony ufające (np. witryny internetowe lub aplikacje implementujące WebAuthn) **pobierają instrukcje metadanych** z MDS, aby podejmować świadome decyzje dotyczące zaufania do urządzeń uwierzytelniających.

MDS dodaje kluczową warstwę zaufania i weryfikacji bezpieczeństwa dla stron ufających korzystających z WebAuthn:

- **Walidacja uwierzytelniającego:** Umożliwia weryfikację zgodności uwierzytelniającego ze standardami FIDO oraz pomaga sprawdzić AAGUID przedstawiony w poświadczeniu WebAuthn.
- **Wykrywanie naruszenia i unieważnienia:** Dostarcza aktualnych informacji o naruszonych lub unieważnionych uwierzytelniaczach, umożliwiając stronom uzależnionym blokowanie niezabezpieczonych urządzeń.
- **Zapewnienie bezpieczeństwa:** Pomaga w egzekwowaniu zasad bezpieczeństwa, takich jak zezwalanie wyłącznie na uwierzytelniające spełniające określony poziom certyfikacji FIDO lub siłę weryfikacji użytkownika.
- **Interoperacyjność:** Zapewnia spójne zachowanie i oczekiwania dotyczące bezpieczeństwa w różnych przeglądarkach, platformach i urządzeniach korzystających z różnych urządzeń uwierzytelniających.

## Konfiguracja

Konfigurację MDS można przeprowadzić za pomocą właściwości `WebAuthnOptions.MDS`; poniżej przedstawiono główne właściwości:

- **Enabled:** jeśli true (wartość domyślna), żądania webauthn będą weryfikowane względem skonfigurowanego pliku MDS.
- **MDS\_FileName:** ścieżka, w której przechowywany jest plik MDS. Można go pobrać z następującego adresu URL: <https://mds3.fidoalliance.org/>
- **RootCert\_FileName:** ścieżka, w której przechowywany jest certyfikat główny. Musi być zdefiniowany, aby zweryfikować łańcuch certyfikatów. Można go pobrać ze strony <https://valid.r3.roots.globalsign.com/>
- **Leaf\_CertificateCRL:** jeśli ustawiona na true (domyślnie false), certyfikat liścia obiektu blob zostanie zweryfikowany względem listy CRL (Lista odwołania certyfikatów).
- **CRL\_FileName:** ścieżka do pliku CRL. Jeśli nie jest zdefiniowana, a opcja `Leaf_CertificateCRL` jest włączona, zostanie podjęta próba automatycznego pobrania pliku CRL.





## Autoryzacja WebAuthn

Jeśli serwer ma po pomyślnym uwierzytelnieniu wysłać token bearer, który może być używany do otwierania nowego połączenia WebSocket lub HTTP, należy przekazać parametr `token = true`. Przykład:

```
{
  "username": "alice@example.com", "token": true
}
```

Po pomyślnym uwierzytelnieniu serwer wyśle odpowiedź o następującej postaci:

```
{
  "verified": "ok",
  "authentication": {
    "token":
      "C760C1C39E3D4E829693A13F18F5CFDE537B516336FC48F7BAB0276176F9E6DE"
  }
}
```

Zdarzenie **OnWebAuthnUnauthorized** jest wywoływane, gdy żądanie nie jest autoryzowane i połączenie zostanie rozłączone. W tym miejscu można skonfigurować, które punkty końcowe wymagają uwierzytelniania WebAuthn, a które nie.

## WebAuthn Autoryzacja | HTTP

Po uzyskaniu nowego tokenu wystarczy wysłać nagłówek autoryzacji z tym tokenem Bearer. Właściwość CustomHeaders komponentu TsgcHTTP1Client umożliwia skonfigurowanie tokenu Bearer. Przykład:

```
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);
}
```

## WebAuthn Authorization | WebSocket

Po uzyskaniu nowego tokenu wystarczy wysłać nagłówek autoryzacji z tym tokenem Bearer. Do dodania tokenu Bearer do żądania połączenia można użyć zdarzenia OnHandShake. Przykład:

```
void __fastcall OnClientHandshake(TsgcWSConnection *Connection, TStringList *Headers)
{
    Headers->Add("Authorization: Bearer C760C1C39E3D4E829693A13F18F5CFDE537B516336FC48F7BAB0276176F9E6DE");
}
```

# Webauthn | Klient JavaScript

**WebAuthn** (Web Authentication API) to standard W3C umożliwiający bezpieczne uwierzytelnianie bez hasła przy użyciu **kryptografii klucza publicznego**. Zamiast haseł użytkownicy rejestrują się i uwierzytelniają przy użyciu **sprzętowych urządzeń uwierzytelniających** (takich jak czytniki linii papilarnych, Face ID, YubiKeys itp.) lub uwierzytelniatorów platformowych (wbudowanych, jak Touch ID).

Poniżej przedstawiono, jak obsługiwać rejestrację i uwierzytelnianie przy użyciu klienta JavaScript.

## Rejestracja WebAuthn

Jak działa rejestracja WebAuthn

- **Użytkownik inicjuje rejestrację:**
  - Użytkownik podaje nazwę użytkownika i klika **Zarejestruj**.
- **Przeglądarka żąda opcji od serwera:**
  - Frontend wysyła żądanie POST w celu uzyskania opcji rejestracji.
- **Przeglądarka tworzy poświadczenia:**
  - Za pomocą `API navigator.credentials.create()` przez `SimpleWebAuthnBrowser.startRegistration()` tworzone jest poświadczenie.
- **Serwer weryfikuje rejestrację:**
  - Przeglądarka odsyła dane uwierzytelniające do serwera.
  - Serwer weryfikuje rejestrację i przechowuje klucz publiczny dla tego użytkownika.

Komponent [TsgcWSAPIServer\\_WebAuthn](#) zawiera plik HTML służący do testowania protokołu WebAuthn. Plik HTML zawiera minimalny interfejs użytkownika oraz kod JavaScript umożliwiający interakcję z WebAuthn.

## Omówienie pliku sgcWebAuthn.html

Przegląd struktury:

- **Dane wejściowe nazwy użytkownika:** Przechwytuje identyfikator użytkownika.

- **Przyciski:**
  - Register – inicjuje rejestrację WebAuthn.
  - Authenticate – inicjuje logowanie (obsługiwane analogicznie).
- **Konsola debugowania:** Wyświetla informacje debugowania w czasie rzeczywistym (JSON z WebAuthn).

## 1. Interfejs HTML do wprowadzania danych

```
<input type="text" id="username" name="username" autocomplete="username webauthn" />
<button id="btnRegBegin"><strong>Register</strong></button>
```

## 2. JavaScript: procedura obsługi kliknięcia przycisku

```
document.querySelector('#btnRegBegin').addEventListener('click', async () => {
  const username = document.getElementById("username").value;
  if (username == "") {
    document.getElementById('Error').innerText = 'Please enter a username to register';
    return;
  }

  const resp = await fetch('<#webauthn_registration_options>', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ username, algorithms: [] })
  });

  const options = await resp.json();
  const attResp = await startRegistration(options); // WebAuthn API
```

## 3. Odpowiedź serwera (fałszywy punkt końcowy w HTML)

```
fetch('/sgcWebAuthn/Registration/Options', ...)
fetch('/sgcWebAuthn/Registration/Verify', ...)
```

## 4. Finalizowanie rejestracji

```
const verificationResp = await fetch('/webauthn/register/verify', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(attResp)
});

const verificationJSON = await verificationResp.json();
if (verificationJSON && verificationJSON.verified) {
  document.getElementById('Success').innerHTML = `Authenticator registered!`;
}
```

## Uwierzytelnianie WebAuthn

Jak działa uwierzytelnianie WebAuthn

- **Użytkownik inicjuje uwierzytelnianie:**
  - Użytkownik podaje nazwę użytkownika i klika **Zarejestruj**.
- **Przeglądarka żąda opcji od serwera:**
  - Interfejs użytkownika wysyła żądanie POST, aby uzyskać opcje uwierzytelniania.
- **Przeglądarka tworzy poświadczenia:**
  - Za pomocą `API navigator.credentials.get()` przez `SimpleWebAuthnBrowser.startAuthentication()`.
- **Serwer weryfikuje uwierzytelnianie:**
  - Przeglądarka odsyła poświadczenie do serwera.
  - Serwer weryfikuje podpisany wynik.

### 1. Konfiguracja interfejsu HTML

```
<div class="container">
<h1>WebAuthn Authentication Sample</h1>

  <section id="userdata">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" autocomplete="username webauthn"
autofocus />
  </section>
```

```

<button id="btnAuthBegin"><strong>Authenticate</strong></button>

<p id="Success" class="success"></p>
<p id="Error" class="error"></p>

<details open>
  <summary>Console</summary>
  <textarea id="Debug"></textarea>
</details>
</div>

```

## 2. Pobierz opcje uwierzytelniania

Przed wywołaniem startAuthentication należy wysłać nazwę użytkownika do serwera

```

const resp = await fetch('<#webauthn_authentication_options>', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    username: document.getElementById("username").value,
    user_verification: 'preferred'
  }),
});

```

Serwer odpowiada obiektem JSON zawierającym:

```

{
  "challenge": "base64url-encoded-random-string",
  "allowCredentials": [
    {
      "id": "base64url-credential-id",
      "type": "public-key"
    }
  ],
  "userVerification": "preferred",
  "rpId": "yourdomain.com"
}

```

Jest to PublicKeyCredentialRequestOptions.



### 3. Odbierz odpowiedź uwierzytelniającą

Odpowiedź `asseResp` wygląda następująco (uproszczona):

```
{
  "id": "credentialId",
  "rawId": "base64url-encoded-id",
  "response": {
    "authenticatorData": "...",
    "clientDataJSON": "...",
    "signature": "...",
    "userHandle": "..."
  },
  "type": "public-key",
  "clientExtensionResults": {}
}
```

Odpowiedź ta dowodzi, że użytkownik:

- Posiada klucz prywatny przechowywany na urządzeniu uwierzytelniającym
- Podpisano wyzwanie serwera
- Był fizycznie obecny (jeśli wymagane)

### 4. Wyślij podpisaną odpowiedź uwierzytelniającą do serwera

Po interakcji użytkownika z jego uwierzytelniaczem (za pomocą `startAuthentication()`), w języku JavaScript otrzymywany jest obiekt odpowiedzi.

```
const verificationResp = await fetch('/webauthn/authenticate/verify', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(asseResp), // this is the signed response
});
```

### 5. Pobierz odpowiedź serwera

Serwer odpowie wynikiem w następującej postaci:

```
{ "verified": true }
```

Lub jeśli coś poszło nie tak:

```
{ "verified": false, "error": "Invalid signature" }
```

Obsługuje się to w kodzie frontendowym w następujący sposób:

```
const result = await verificationResp.json();

if (result.verified) {
  document.getElementById('Success').textContent = 'User authenticated!';
} else {
  document.getElementById('Error').textContent = 'Authentication failed!';
}
```

# Amazon AWS | SQS

## Czym jest Amazon SQS?

Amazon Simple Queue Service (SQS) to w pełni zarządzana usługa kolejkowania komunikatów, która umożliwia rozdzielenie i skalowanie mikroservisów, systemów rozproszonych i aplikacji bezserwerowych. SQS eliminuje złożoność i koszty związane z zarządzaniem i obsługą oprogramowania pośredniczącego zorientowanego na komunikaty, umożliwiając programistom skupienie się na wartościowej pracy. Korzystając z SQS, można wysyłać, przechowywać i odbierać komunikaty między komponentami oprogramowania w dowolnym wolumenie, bez utraty komunikatów i bez konieczności dostępności innych usług.

## Korzyści

### Eliminacja nakładu administracyjnego

W przypadku SQS nie ma kosztów początkowych, nie ma potrzeby nabywania, instalowania i konfigurowania oprogramowania do przesyłania wiadomości ani czasochłonnego budowania i utrzymywania infrastruktury pomocniczej.

### Niezawodne dostarczanie wiadomości

SQS pozwala na rozdzielenie komponentów aplikacji, dzięki czemu działają i mogą ulegać awarii niezależnie, co zwiększa ogólną odporność systemu na błędy.

### Zabezpieczaj poufne dane

Usługi Amazon SQS można używać do wymiany poufnych danych między aplikacjami, stosując szyfrowanie po stronie serwera (SSE) do szyfrowania każdej treści wiadomości.

### Elastyczne i efektywne kosztowo skalowanie

SQS skaluje się elastycznie wraz z aplikacją, dzięki czemu nie trzeba martwić się o planowanie wydajności i wstępne przydzielanie zasobów.

## Przepływ pracy

Poniższy scenariusz opisuje cykl życia wiadomości Amazon SQS w kolejce, od jej utworzenia do usunięcia.



## Pierwsze kroki z Amazon SQS

Przed rozpoczęciem wykonaj kroki opisane w sekcji Konfigurowanie Amazon SQS.

### Krok 1: Utwórz kolejkę

1. Zaloguj się do konsoli Amazon SQS.
2. Wybierz opcję Utwórz nową kolejkę.
3. Na stronie tworzenia nowej kolejki należy upewnić się, że wybrano właściwy region, a następnie wpisać nazwę kolejki.
4. Standard jest wybrany domyślnie. Wybierz FIFO.
5. Aby utworzyć kolejkę z domyślnymi parametrami, należy wybrać Quick-Create Queue.

Nowa kolejka została utworzona i wybrana na liście kolejek.

### Krok 2: Wyślij wiadomość

Po utworzeniu kolejki można wysłać do niej wiadomość. Poniższy przykład pokazuje wysyłanie wiadomości do istniejącej kolejki.

1. Z listy kolejek należy wybrać utworzoną kolejkę.
2. W obszarze Queue Actions wybierz Send a Message.
3. Wiadomość jest wysyłana, a wyświetlane jest okno dialogowe Wyślij wiadomość do QueueName z atrybutami wysłanej wiadomości.

### Krok 3: Odbierz i usuń wiadomość

Po umieszczeniu wiadomości w kolejce można ją pobrać (odebrać z kolejki). Przy pobieraniu wiadomości z kolejki nie można określić, którą wiadomość chce się pobrać. Zamiast tego określa się maksymalną liczbę wiadomości (do 10), które chce się pobrać.

## Krok 4: Usunąć kolejkę

Jeśli kolejka Amazon SQS nie jest używana (i nie przewiduje się jej używania w najbliższej przyszłości), zalecaną praktyką jest jej usunięcie z Amazon SQS.

## Klient SQS

```
// TsgcHTTPAWS_SQS_Client is the component used for connect to Amazon SQS.
// Client connects using HTTPs protocol and authenticates using Access Key provided by Amazon.

// Before you attempt to connect to SQS service, you must set some data in AWSOptions property.

// Region: your endpoint region, example: us-east-1.
// AccessKey: access key provided by Amazon.
// SecretKey: secret key provided by Amazon.

// The following methods are supported by SQS client:

// AddPermission
// Adds a permission to a queue for a specific principal. This allows sharing access to the queue.

// ChangeMessageVisibility
// Changes the visibility timeout of a specified message in a queue to a new value. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.

// ChangeMessageVisibilityBatch
// Changes the visibility timeout of multiple messages. This is a batch version of ChangeMessageVisibility.
// The result of the action on each message is reported individually in the response. You can send up to 10 ChangeMessageVisibilityBatch requests per second.

// CreateQueue
// Creates a new standard or FIFO queue. You can pass one or more attributes in the request.

vURL = SQS->CreateQueue("sqs_queue");
if (vURL != "")
{
    DoLog("#CreateQueue: " + vURL);
}

// DeleteMessage
// Deletes the specified message from the specified queue. To select the message to delete, use the ReceiptHandle of the message.

if (SQS->DeleteMessage("sqs_queue", "...receipt handle goes here...") == true)
{
    DoLog("#DeleteMessage: ok");
}
else
{
    DoLog("#DeleteMessage: error");
}
```

```

// DeleteMessageBatch
// Deletes up to ten messages from the specified queue. This is a batch version of DeleteMessage. The result of the action

// DeleteQueue
// Deletes the queue specified by the queue name, regardless of the queue's contents.

    if (SQS->DeleteQueue(txtQueueName->Text) == true)
    {
        DoLog("#Delete Queue: ok");
    }
    else
    {
        DoLog("#Delete Queue: error");
    }

// GetQueueAttributes
// Gets attributes for the specified queue.

TsgcSQSAttributes *oAttributes = new TsgcSQSAttributes();
try
{
    if (SQS->GetQueueAttributes("sqs_queue", oAttributes) == true)
    {
        for (int i = 0; i < oAttributes->Count; i++)
        {
            DoLog("#Attribute: " + static_cast<TsgcSQSAttribute*>(oAttributes->Item[i])
                ->AttributeName + " " + static_cast<TsgcSQSAttribute*>(oAttributes->Item[i])
                ->AttributeValue);
        }
    }
    else
    {
        DoLog("#GetQueueAttributes: error");
    }
}
__finally
{
    oAttributes->Free();
}

// GetQueueUrl
// Returns the URL of an existing Amazon SQS queue.

// ListDeadLetterSourceQueues
// Returns a list of your queues that have the RedrivePolicy queue attribute configured with a dead-letter queue.

// ListQueueTags
// List all cost allocation tags added to the specified Amazon SQS queue.

// PurgeQueue
// Deletes the messages in a queue specified by the QueueName parameter.

    if (SQS->PurgeQueue("sqs_queue") == true)
    {
        DoLog("#PurgeQueue: ok");
    }
    else
    {
        DoLog("#PurgeQueue: error");
    }

// ReceiveMessage
// Retrieves one or more messages (up to 10), from the specified queue.

TsgcSQSReceiveMessageResponses *oResponses = new TsgcSQSReceiveMessageResponses();
try
{

```

```

    if (SQS->ReceiveMessage("sqs_test", oResponses) == true)
    {
        for (int i = 0; i < oResponses->Count; i++)
        {
            DoLog("#ReceiveMessage: " + static_cast<TsgcSQSReceiveMessageResponse*>(oResponses->Item[i])->Body);
            FReceiptHandle = static_cast<TsgcSQSReceiveMessageResponse*>(oResponses->Item[i])->ReceiptHandle;
        }
    }
}
__finally
{
    oResponses->Free();
}

// RemovePermission
// Revokes any permissions in the queue policy that matches the specified Label parameter.

// SendMessage
// Delivers a message to the specified queue.

if (SQS->SendMessage("sqs_queue", "My First Message") == true)
{
    DoLog("#SendMessage: ok");
}
else
{
    DoLog("#SendMessage: error");
}

// SendMessageBatch
// Delivers up to ten messages to the specified queue. This is a batch version of SendMessage.

// SetQueueAttributes
// Sets the value of one or more queue attributes. When you change a queue's attributes, the change can take up to 60 seconds
// for most of the attributes to propagate throughout the Amazon SQS system.

TsgcSQSAttributes *oAttributes = new TsgcSQSAttributes();
try
{
    oAttributes->AddSQSAttribute(sqsatVisibilityTimeout, "45");
    if (SQS->SetQueueAttributes("sqs_queue", oAttributes) == true)
    {
        DoLog("#SetQueueAttributes: ok");
    }
    else
    {
        DoLog("#SetQueueAttributes: error");
    }
}
__finally
{
    oAttributes->Free();
}

// TagQueue
// Add cost allocation tags to the specified Amazon SQS queue.

// UntagQueue
// Remove cost allocation tags from the specified Amazon SQS queue.

```

## **Zdarzenia**

### **OnSQSBeforeRequest**

To zdarzenie jest wywoływane przed wysłaniem przez komponent sqs żądania HTTP. Można uzyskać dostęp do parametru URL, a jeśli parametr Handled zostanie ustawiony na True, komponent nie wyśle żądania HTTP.

### **OnSQSError**

Jeśli podczas wykonywania żądania przez komponent wystąpi błąd, zdarzenie zostanie wywołane z kodem błędu i opisem błędu.

### **OnSQSResponse**

Zdarzenie jest wywoływane po żądaniu HTTP zawierającym nieprzetworzoną odpowiedź serwera.



## Google Cloud | Google OAuth2 Keys

Aby korzystać ze składników Google Cloud dla sgcWebSockets i uwierzytelniać się przy użyciu OAuth2, należy najpierw uzyskać klucz OAuth2 z Google Cloud.

Poniżej znajdują się kroki umożliwiające uzyskanie kluczy Google OAuth2 oraz sposób ich konfiguracji w naszej przykładowej aplikacji PubSub.

Najpierw **zaloguj się** do swojego **konta Google Cloud** i użyj istniejącego projektu lub utwórz nowy.

Następnie należy przejść do menu **Credentials** i nacisnąć przycisk **CREATE CREDENTIALS**, a następnie wybrać opcję **OAuth Client ID**.



Wybierz typ aplikacji i nadaj jej opisową nazwę.



Jeśli operacja zakończy się powodzeniem, zostaną zwrócone identyfikator klienta (Client ID) i tajny klucz klienta (Client Secret).



Nie należy udostępniać danych OAuth2 nikomu!

Teraz skopiuj do przykładu PubSub sgcWebSockets i dodaj Project Id (NIE nazwę projektu)



W ten sposób należy go skonfigurować w przykładzie PubSub dla sgcWebSocket.



Następnie można spróbować na przykład utworzyć nowy temat. Przy pierwszym uruchomieniu wymagana jest autoryzacja połączenia OAuth2, dlatego zostanie otwarta nowa przeglądarka internetowa z prośbą o zezwolenie na dostęp do konta przy użyciu poświadczeń OAuth2 dostarczonych przez Google.



Zezwól na połączenie, a po jego nawiązaniu możesz rozpocząć pracę z tym API.



## Google Cloud | Konta usług Google

Aby korzystać z komponentów sgcWebSockets Google Cloud i uwierzytelniać się przy użyciu kont usługi, należy najpierw uzyskać certyfikat klucza prywatnego

z Google Cloud.

Poniżej przedstawiono kroki umożliwiające uzyskanie certyfikatu klucza prywatnego Google oraz sposób jego konfiguracji w przykładowej aplikacji PubSub.

Najpierw **zaloguj się** do swojego **konta Google Cloud** i użyj istniejącego projektu lub utwórz nowy.



Wybierz **CREATE SERVICE ACCOUNT** i zostanie wyświetlona nowa strona, na której należy ustawić nazwę i opis konta usługi



Następnie wybierz co najmniej jedną rolę. W tym przykładzie wybrano PubSub Admin, aby umożliwić klientowi publikowanie i subskrybowanie tematów, ale można wybrać inną rolę z mniejszymi uprawnieniami.



Naciśnij CONTINUE, a następnie możesz przyznać dostęp innym użytkownikom



Po zakończeniu naciśnij DONE, a zostanie wyświetlony nowy rekord.



Następnym krokiem jest utworzenie nowego klucza — w kolumnie akcji należy wybrać opcję Utwórz klucz. Aby pobrać konfigurację w formacie JSON, należy wybrać JSON; zostanie wówczas utworzony nowy klucz.



Na koniec wystarczy wypełnić dane dostarczone przez Google w kliencie PubSub sgcWebSockets. Można użyć **LoadSettingsFromFile** do załadowania pliku JSON z konfiguracją.



# Google Cloud | Pub/Sub

## Co to jest Google Cloud Pub/Sub?

Pub/Sub zapewnia elastyczność i niezawodność korporacyjnego oprogramowania pośredniczącego zorientowanego na wiadomości w chmurze. Jednocześnie Pub/Sub to skalowalny, trwały system pozyskiwania i dostarczania zdarzeń, stanowiący podstawę nowoczesnych potoków analityki strumieniowej. Zapewniając asynchroniczne przesyłanie wiadomości między wieloma nadawcami i odbiorcami, które oddziela nadawców od odbiorców, umożliwia bezpieczną i wysoce dostępną komunikację między niezależnie napisanymi aplikacjami. Pub/Sub zapewnia niskolatencyjne, trwałe przesyłanie wiadomości, które pomaga deweloperom szybko integrować systemy hostowane na platformie Google Cloud Platform i poza nią.

## Funkcje

|                                                                                                                                                                                                 |                                                                                                                                                                                                                          |                                                                                                                                            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Dostarczanie co najmniej raz</b> Synchroniczna, wielostrefowa replikacja wiadomości i śledzenie potwierdzeń dla każdej wiadomości zapewnia dostarczanie co najmniej raz w dowolnej skali.    | <b>Otwarte</b> Otwarte API i biblioteki klienckie w siedmiu językach obsługują wdrożenia wielochmurowe i hybrydowe.                                                                                                      | <b>Przetwarzanie dokładnie raz</b> Cloud Dataflow obsługuje niezawodne, ekspresyjne przetwarzanie strumieni Cloud Pub/Sub dokładnie raz.   |
| <b>Domyślnie globalny</b> Publikuj z dowolnego miejsca na świecie i konsumuj z dowolnego miejsca, ze stałym opóźnieniem. Replikacja nie jest konieczna.                                         | <b>Brak konfiguracji wstępnej, pełna automatyzacja</b> Cloud Pub/Sub nie stosuje fragmentów ani partycji. Wystarczy ustawić przydział, publikować i odbierać wiadomości.                                                 | <b>Zgodność i bezpieczeństwo</b> Cloud Pub/Sub to usługa zgodna z HIPAA, oferująca szczegółową kontrolę dostępu i kompleksowe szyfrowanie. |
| <b>Zintegrowany</b> Skorzystaj z integracji z wieloma usługami, takimi jak Cloud Storage, zdarzenia aktualizacji Gmail oraz Cloud Functions do bezserwerowych obliczeń sterowanych zdarzeniami. | <b>Przewijanie i odtwarzanie</b> Przewijanie zaległości do dowolnego momentu w czasie lub migawki, umożliwiające ponowne przetwarzanie wiadomości. Szybkie przewijanie do przodu w celu odrzucenia nieaktualnych danych. |                                                                                                                                            |

## Relacje wydawca-subskrybent

Aplikacja wydawcy tworzy i wysyła wiadomości do tematu. Aplikacje subskrybentów tworzą subskrypcję tematu w celu odbierania wiadomości. Komunikacja może być typu jeden-do-wielu (fan-out), wiele-do-jednego (fan-in) i wiele-do-wielu.



## Typowe przypadki użycia

- **Równoważenie obciążeń w klastrach sieciowych.** Na przykład duża kolejka zadań może być efektywnie rozdzielana między wielu pracowników, takich jak instancje Google Compute Engine.
- **Implementowanie asynchronicznych przepływów pracy.** Na przykład aplikacja do przetwarzania zleceń może umieszczać zlecenie w temacie, skąd może być ono przetworzone przez jednego lub więcej pracowników.
- **Dystrybucja powiadomień o zdarzeniach.** Na przykład usługa akceptująca rejestracje użytkowników może wysyłać powiadomienia za każdym razem, gdy zarejestruje się nowy użytkownik, a usługi downstream mogą subskrybować w celu odbierania powiadomień o tym zdarzeniu.
- **Odświeżanie rozproszonych pamięci podręcznych.** Na przykład aplikacja może publikować zdarzenia unieważnienia w celu aktualizacji identyfikatorów zmienionych obiektów.
- **Rejestrowanie w wielu systemach.** Na przykład instancja Google Compute Engine może zapisywać logi do systemu monitorowania, do bazy danych na potrzeby późniejszego wyszukiwania itd.
- **Strumieniowanie danych z różnych procesów lub urządzeń.** Na przykład czujnik w miejscu zamieszkania może strumieniować dane do serwerów zaplecza hostowanych w chmurze.
- **Poprawa niezawodności.** Na przykład usługa Compute Engine działająca w jednej strefie może działać w dodatkowych strefach, subskrybując wspólny temat, co pozwala na odtwarzanie po awarii w strefie lub regionie.

## Autoryzacja

Komponent klienta Google Pub/Sub może logować się do serwerów Google przy użyciu następujących metod:

- **gcaOAuth2:** protokół OAuth2
- **gcaJWT:** tokeny JWT.

## OAuth2

Logowanie odbywa się za pośrednictwem przeglądarki internetowej, w której użytkownik loguje się na własne konto i autoryzuje żądania PubSub.

- **GoogleCloudOptions.OAuth2.ClientId:** jest to ClientID dostarczony przez Google do uwierzytelniania przez protokół OAuth2.

- **GoogleCloudOptions.OAuth2.ClientSecret:** ciąg Client Secret dostarczony przez Google w celu uwierzytelniania przez protokół OAuth2.
- **GoogleCloudOptions.OAuth2.Scope:** zakres OAuth2; zazwyczaj nie ma potrzeby modyfikowania wartości domyślnej, chyba że wymagany jest szerszy dostęp niż domyślny.
- **GoogleCloudOptions.OAuth2.LocalIP:** protokół OAuth2 wymaga lokalnego serwera nasłuchującego na odpowiedź z serwera uwierzytelniania. Jest to adres IP lub nazwa DNS. Domyślnie wynosi 127.0.0.1.
- **GoogleCloudOptions.OAuth2.LocalPort:** lokalny port nasłuchiwanie serwera.
- **GoogleCloudOptions.OAuth2.RedirectURL:** jeśli wymagany jest adres URL przekierowania inny niż LocalPort + LocalIP, można go ustawić w tej właściwości (przykład: http://127.0.0.1:8080/oauth2).

## Konta usługowe

Logowanie odbywa się przez podpisywanie żądań przy użyciu klucza prywatnego dostarczonego przez Google. Ta metoda jest zalecana dla zautomatyzowanych usług lub aplikacji bez interakcji użytkownika.

- **GoogleCloudOptions.JWT.ClientEmail:** nazwa adresu e-mail klienta podana podczas tworzenia nowego konta usługi. Węzeł „client\_email” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKeyId:** identyfikator klucza prywatnego dostarczony przez Google. Węzeł "private\_key\_id" w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKey:** certyfikat klucza prywatnego dostarczony przez Google. Węzeł "private\_key" w pliku konfiguracyjnym JSON.

## Opcje TLS

**TLSOptions** centralizuje konfigurację bezpiecznego kanału używanego do komunikacji z Google Cloud Pub/Sub.

- **TLSOptions.IOHandler:** należy wybrać implementację TLS odpowiednią dla danego wdrożenia (OpenSSL, SChannel itp.).
- **TLSOptions.Version:** ogranicza negocjację do określonej wersji TLS, gdy wymaga tego infrastruktura.
- **TLSOptions.VerifyCertificate:** kontroluje walidację certyfikatu, umożliwiając stosowanie bardziej restrykcyjnych zasad bezpieczeństwa.
- **TLSOptions.OpenSSL\_Options.LibPath:** ustaw katalog zawierający pliki binarne OpenSSL dostarczone z aplikacją.
- **TLSOptions.Schannel\_Options.UseLegacyCredentials:** aktywuje obsługę starszych poświadczeń systemu Windows, gdy wymaga tego SChannel.

## Przykład:

```
TsgcHTTPGoogleCloud_PubSub_Client *oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client(NULL);
oPubSub->TLSOptions->IOHandler = iohOpenSSL;
oPubSub->TLSOptions->Version = tls1_3;
oPubSub->TLSOptions->VerifyCertificate = true;
oPubSub->TLSOptions->OpenSSL_Options->LibPath = oslpDefaultFolder;
```

Po utworzeniu nowego konta usługi można pobrać plik JSON zawierający wszystkie konfiguracje. Plik ten może być przetworzony przez komponent PubSub. Wystarczy wywołać metodę **LoadSettingsFromFile** i przekazać nazwę pliku JSON jako argument.

## Najczęstsze zastosowania

- Konfiguracja
  - [Klucze Google OAuth2](#)
  - [Konta usługowe](#)

## Klient Google Pub/Sub

### OAuth2

Aby korzystać z Google Pub/Sub API, komponent Pub/Sub `sgcWebSockets` używa OAuth2 jako domyślnego uwierzytelniania; dlatego najpierw należy ustawić swój **ClientId** i **ClientSecret** z konta Google.

```
oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->Authorization = gcaOAuth2;
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
```

### Konta usługowe

Konta usług wymagają zbudowania tokenu JWT i przekazania go jako tokenu autoryzacji.

```
TsgcHTTPGoogleCloud_PubSub_Client *oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client(this);
oPubSub->GoogleCloudOptions->Authorization = gcaJWT;
oPubSub->GoogleCloudOptions->JWT->ClientEmail = "...google email...";
oPubSub->GoogleCloudOptions->JWT->PrivateKeyId = "...private key id...";
oPubSub->GoogleCloudOptions->JWT->PrivateKey->Lines->Text = "...private key certificate...";
```

Jest to wymagane w celu uzyskania tokenu autoryzacyjnego z Google, który będzie używany we wszystkich wywołaniach Rest API.

Wszystkie metody zwracają odpowiedź, która może być pomyślna lub zawierać błąd.



Projects.Snapshots

|                 |                                 |                                                                                                                                                                                                                                                                                  |                                                                 |
|-----------------|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| Create-Snapshot | project, snapshot, subscription | Tworzy migawkę z żądanej subskrypcji. Migawki są używane w operacjach subscriptions.seek, które umożliwiają masowe zarządzanie potwierdzeniami wiadomości. Oznacza to, że można ustawić stan potwierdzenia wiadomości w istniejącej subskrypcji na stan uchwycony przez migawkę. | CreateSnapshot('pubsub-270909', 'snapshot-1', 'subscription-1') |
| Delete-Snapshot | projekt, migawka                | Usuwa istniejącą migawkę.                                                                                                                                                                                                                                                        | DeleteSnapshot('pubsub-270909', 'snapshot-1')                   |
| ListSnapshots   | projekt                         | Wyświetla istniejące migawki                                                                                                                                                                                                                                                     | ListSnapshots('pubsub-270909')                                  |

Projects.Subscriptions

|                          |                              |                                                                                                                                                     |                                                                  |
|--------------------------|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Acknowledge-Subscription |                              |                                                                                                                                                     |                                                                  |
| Create-Subscription      | project, subscription, topic | Tworzy subskrypcję dla danego tematu. Jeśli subskrypcja już istnieje, zwraca ALREADY_EXISTS. Jeśli odpowiedni temat nie istnieje, zwraca NOT_FOUND. | CreateSubscription('pubsub-270909', 'subscription-1', 'topic-1') |
| Delete-Subscription      | projekt, subskrypcja         | Usuwa istniejącą subskrypcję. Wszystkie wiadomości zachowane w subskrypcji są natychmiast usuwane.                                                  | DeleteSubscription('pubsub-270909', 'subscription-1')            |
| Get-Subscription         | projekt, subskrypcja         | Pobiera szczegóły konfiguracji subskrypcji.                                                                                                         | GetSubscription('pubsub-270909', 'subscription-1')               |
| List-Subscriptions       | projekt                      | Wyświetla pasujące subskrypcje.                                                                                                                     | ListSubscriptions('pubsub-270909', 'subscription-1')             |

|                                                        |                                                          |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------|----------------------------------------------------------|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| scrip-<br>tions                                        |                                                          |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Modi-<br>fyAck-<br>Deadli-<br>neSub-<br>scrip-<br>tion | project,<br>subscrip-<br>tion, Ac-<br>kIds               |  | <p>Modyfikuje termin potwierdzenia dla konkretnej wiadomości. Metoda jest przydatna, gdy subskrybent potrzebuje więcej czasu na przetworzenie wiadomości lub gdy chce udostępnić ją ponownie do dostarczenia po przerwaniu przetwarzania. Należy pamiętać, że operacja ta nie zmienia właściwości ackDeadlineSeconds obowiązującej dla kolejnych wiadomości na poziomie subskrypcji.</p> <p>Modyfikuje konfigurację PushConfig dla wskazanej subskrypcji. Operacja ta może służyć do zmiany subskrypcji push na subskrypcję pull (co sygnalizuje pusta wartość PushConfig) lub odwrotnie, a także do zmiany adresu URL punktu końcowego i innych atrybutów subskrypcji push. Wiadomości będą się gromadzić do dostarczenia w sposób ciągły przez cały czas trwania wywołania, niezależnie od zmian w PushConfig.</p> |
| Modi-<br>fyPush-<br>Config-<br>Sub-<br>scrip-<br>tion  | projekt,<br>subskrypcja                                  |  | <p>Pobiera wiadomości z serwera. Serwer może zwrócić UNAVAILABLE, jeśli dla danej subskrypcji oczekuje zbyt wiele jednoczesnych żądań pobierania.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Pull                                                   | projekt,<br>subskrypcja                                  |  | <p><code>pull('pubsub-270909', 'subscription-1')</code></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Seek                                                   | project,<br>subscrip-<br>tion, ti-<br>meUTC,<br>snapshot |  | <p>Przenosi istniejącą subskrypcję do określonego punktu w czasie lub do danej migawki, w zależności od tego, co zostało podane w żądaniu. Migawki są używane w operacjach subscriptions.seek, które umożliwiają zbiorcze zarządzanie potwierdzeniami wiadomości. Oznacza to, że można ustawić stan potwierdzenia wiadomości w istniejącej subskrypcji na stan zarejestrowany przez migawkę. Należy pamiętać, że zarówno subskrypcja, jak i migawka</p>                                                                                                                                                                                                                                                                                                                                                              |

muszą dotyczyć tego samego tematu.

## Projects.Topics

|             |                         |                                                                                                                                                                                                                           |                                                                 |
|-------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| CreateTopic | project, topic          | Tworzy dany temat o podanej nazwie                                                                                                                                                                                        | CreateTopic('pubsub-270909', 'topic-1')                         |
| DeleteTopic | project, topic          | Usuwa temat o podanej nazwie. Zwraca NOT_FOUND, jeśli temat nie istnieje. Po usunięciu tematu można utworzyć nowy temat o tej samej nazwie; będzie to zupełnie nowy temat bez żadnej starej konfiguracji ani subskrypcji. | DeleteTopic('pubsub-270909', 'topic-1')                         |
| GetTopic    | project, topic          | Pobiera konfigurację tematu.                                                                                                                                                                                              | GetTopic('pubsub-270909', 'topic-1')                            |
| ListTopics  | projekt                 | Wyświetla pasujące tematy.                                                                                                                                                                                                | ListTopics('pubsub-270909')                                     |
| Opublikuj   | project, topic, message | Dodaje jedną lub więcej wiadomości do tematu. Zwraca NOT_FOUND, jeśli temat nie istnieje.                                                                                                                                 | Publish('pubsub-270909', 'topic-1', 'My First PubSub Message.') |

## Projects.Topics.Subscriptions

|                        |                |                                                      |                                                    |
|------------------------|----------------|------------------------------------------------------|----------------------------------------------------|
| ListTopicSubscriptions | project, topic | Wyświetla nazwy subskrypcji dotyczących tego tematu. | ListTopicSubscriptions('pubsub-270909', 'topic-1') |
|------------------------|----------------|------------------------------------------------------|----------------------------------------------------|

## Najczęstsze metody

Poniżej znajdują się najczęściej używane metody w Google Cloud Pub/Sub API.

## Jak utworzyć nowy temat

Utwórz nowy temat dla projektu o id: pubsub-270909 i nazwie tematu topic-1.

```
oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
oPubSub->CreateTopic("pubsub-270909", "topic-1");
```

## Odpowiedź serwera

```
{
  "name": "projects/pubsub-270909/topics/topic-1"
}
```

## Opublikuj wiadomość

Opublikowanie nowej wiadomości w nowo utworzonym temacie.

```
oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
oPubSub->Publish("pubsub-270909", "topic-1", "My First Message from sgcWebSockets.");
```

## Odpowiedź serwera

```
{
  "messageIds": [
    "1050732082561505"
  ]
}
```

## Publikacja wiadomości z atrybutami

```
TsgcHTTPGoogleCloud_PubSub_Client *oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
oAttributes = new TStringList();
try
{
  oAttributes->CommaText = "origin=gcloud-sample,username=gcp";
  oPubSub->Publish("pubsub-270909", "topic-1", "My First Message from sgcWebSockets.", oAttributes, "username"));
}
__finally
{
  oAttributes->Free();
}
```

## Jak utworzyć nową subskrypcję

Utwórz nową subskrypcję dla projektu o identyfikatorze pubsub-270909, z nazwą subskrypcji subscription-1 i topic-1

```
oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
```

```
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
oPubSub->CreateSubscription("pubsub-270909", "subscription-1", "topic-1");
```

## Odpowiedź serwera

```
{
  "name": "projects/pubsub-270909/subscriptions/subscription-1",
  "topic": "projects/pubsub-270909/topics/topic-1",
  "pushConfig": {},
  "ackDeadlineSeconds": 10,
  "messageRetentionDuration": "604800s",
  "expirationPolicy": {
    "ttl": "2678400s"
  }
}
```

## Jak odczytywać wiadomości z subskrypcji

Odczytaj wiadomości z poprzednio utworzonej subskrypcji.

```
oPubSub = new TsgcHTTPGoogleCloud_PubSub_Client();
oPubSub->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oPubSub->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
oPubSub->pubsub->Pull("pubsub-270909", "subscription-1");
```

## Odpowiedź serwera

```
{
  "receivedMessages": [
    {
      "ackId": "PjA-RVNEUAYWLF1GSFE3GQhoUQ5PXiM_NSAoRREFC08CKF15MEorQVh0Dj4N",
      "message": {
        "data": "TXkgRmlyc3QgTWVzc2FnZSBmcm9tIHNNY1dlY1NvY2tldHMu",
        "messageId": "1050732082561505",
        "publishTime": "2020-03-14T15:25:31.505Z"
      }
    }
  ]
}
```

Wiadomość jest odbierana zakodowana w Base64, dlatego przed odczytaniem treści należy ją najpierw zdekodować.

```
sgcBase_Helpers->DecodeBase64("TXkgRmlyc3QgTWVzc2FnZSBmcm9tIHNNY1dlY1NvY2tldHMu=");
```

# Google Cloud | Calendar

Google Calendar API umożliwia integrację aplikacji z Google Calendar, tworząc nowe możliwości angażowania użytkowników. Calendar API pozwala wyświetlać, tworzyć i modyfikować zdarzenia w kalendarzu, a także pracować z wieloma innymi obiektami związanymi z kalendarzem, takimi jak kalendarze i kontrola dostępu.

## Zasoby API

Google Calendar korzysta z następujących zasobów:

- **Zdarzenie:** Zdarzenie w kalendarzu zawierające informacje takie jak tytuł, godziny rozpoczęcia i zakończenia oraz uczestnicy. Zdarzenia mogą być jednorazowe lub cykliczne. Zdarzenie jest reprezentowane przez zasób Event. Kolekcja Events dla danego kalendarza zawiera wszystkie zasoby zdarzeń tego kalendarza.
- **Calendar:** Kalendarz to zbiór wydarzeń. Każdy kalendarz ma powiązane metadane, takie jak opis kalendarza lub domyślna strefa czasowa. Metadane pojedynczego kalendarza są reprezentowane przez zasób Calendar. Kolekcja Calendars zawiera zasoby Calendar dla wszystkich istniejących kalendarzy.
- **CalendarList:** Lista wszystkich kalendarzy na liście kalendarzy użytkownika w interfejsie Calendar. Metadane pojedynczego kalendarza widocznego na liście kalendarzy są reprezentowane przez zasób CalendarListEntry. Metadane te obejmują właściwości specyficzne dla użytkownika, takie jak kolor lub powiadomienia o nowych zdarzeniach. Kolekcja CalendarList zawiera wszystkie zasoby CalendarListEntry dla danego użytkownika. Dalsze wyjaśnienie różnicy między kolekcjami Calendars i CalendarList można znaleźć w sekcji Calendar and Calendar List.
- **Setting:** Preferencja użytkownika z interfejsu Calendar, na przykład strefa czasowa użytkownika. Pojedyncza preferencja użytkownika jest reprezentowana przez zasób Setting. Kolekcja Settings zawiera wszystkie zasoby Setting dla danego użytkownika.
- **ACL:** Reguła kontroli dostępu przyznająca użytkownikowi (lub grupie użytkowników) określony poziom dostępu do kalendarza. Pojedyncza reguła kontroli dostępu jest reprezentowana przez zasób ACL. Kolekcja ACL dla danego kalendarza zawiera wszystkie zasoby ACL przyznające dostęp do tego kalendarza.
- **Color:** Kolor prezentowany w interfejsie użytkownika kalendarza. Zasób Colors reprezentuje zestaw wszystkich kolorów dostępnych w interfejsie kalendarza, podzielony na dwie grupy: kolory dostępne dla wydarzeń i kolory dostępne dla kalendarzy.
- **Wolny/zajęty:** Czas, w którym kalendarz ma zaplanowane zdarzenia, jest uznawany za „zajęty”; czas bez zdarzeń jest uznawany za „wolny”. Zasób Freebusy umożliwia zapytanie o zbiór zajętych przedziałów czasu dla danego kalendarza lub zestawu kalendarzy.

## Główne funkcje

- W pełni funkcjonalne Google Calendar Client API V3.
- Wszystkie metody obsługiwane przez API mogą być wywoływane przy użyciu API klienta.
- Żądania klientów z użyciem protokołu HTTP/2 (wyłącznie w wersji Enterprise Edition).
- Automatyczna obsługa częściowych odpowiedzi przy użyciu PageNextToken.
- Łatwy dostęp do właściwości danych kalendarza i zdarzeń.
- Metody uwierzytelniania:
  - OAuth2: wymaga interakcji użytkownika.
  - Konta usług (wymagają delegowania uprawnień na poziomie domeny): dla usług Windows, demonów...

## Konfiguracja

Komponent klienta Google Calendar posiada następujące właściwości:

## OAuth2

- **GoogleCloudOptions.OAuth2.ClientId:** jest to ClientID dostarczony przez Google do uwierzytelniania przez protokół OAuth2.
- **GoogleCloudOptions.OAuth2.ClientSecret:** ciąg Client Secret dostarczony przez Google w celu uwierzytelniania przez protokół OAuth2.
- **GoogleCloudOptions.OAuth2.Scope:** zakres OAuth2; zazwyczaj nie ma potrzeby modyfikowania wartości domyślnej, chyba że wymagany jest szerszy dostęp niż domyślny.
- **GoogleCloudOptions.OAuth2.LocalIP:** protokół OAuth2 wymaga lokalnego serwera nasłuchującego na odpowiedź z serwera uwierzytelniania. Jest to adres IP lub nazwa DNS. Domyślnie wynosi 127.0.0.1.
- **GoogleCloudOptions.OAuth2.LocalPort:** lokalny port nasłuchiwanie serwera.
- **GoogleCloudOptions.OAuth2.RedirectURL:** jeśli wymagany jest adres URL przekierowania inny niż LocalPort + LocalIP, można go ustawić w tej właściwości (przykład: <http://127.0.0.1:8080/oauth2>).

Zakresy klienta API można modyfikować za pomocą właściwości Scopes — wystarczy wybrać zakresy obsługiwane przez klienta.

## JWT

Logowanie odbywa się przez podpisywanie żądań przy użyciu klucza prywatnego dostarczonego przez Google; metoda ta jest zalecana dla zautomatyzowanych usług lub aplikacji bez interakcji użytkownika. Wymaga skonfigurowania konta usługi z [delegowaniem uprawnień na poziomie domeny](#).

- **GoogleCloudOptions.JWT.ClientEmail:** nazwa adresu e-mail klienta podana podczas tworzenia nowego konta usługi. Węzeł „client\_email” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKeyId:** identyfikator klucza prywatnego dostarczony przez Google. Węzeł „private\_key\_id” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKey:** certyfikat klucza prywatnego dostarczony przez Google. Węzeł „private\_key” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.Subject:** to adres e-mail konta roboczego powiązanego z kontem usługi przy użyciu delegowania uprawnień w całej domenie.

## **Opcje TLS**

Należy użyć właściwości **TLSOptions** do dostosowania bezpiecznego połączenia nawiązanego z serwerami Google.

- **TLSOptions.IOHandler:** wybiera stos TLS (OpenSSL, SChannel...).
- **TLSOptions.Version:** wymusza określoną wersję protokołu TLS (na przykład `tls1_2` lub `tls1_3`).
- **TLSOptions.VerifyCertificate:** włącza lub wyłącza walidację certyfikatu serwera.
- **TLSOptions.OpenSSL\_Options.LibPath:** wskazuje folder zawierający biblioteki OpenSSL wdrożone wraz z aplikacją.
- **TLSOptions.SChannel\_Options.UseLegacyCredentials:** włącza starszy przepływ poświadczeń wymagany przez niektóre wersje systemu Windows podczas korzystania z SChannel.

Poniższe fragmenty kodu pokazują, jak konfigurować opcje TLS:

```
TsgcHTTPGoogleCloud_Calendar_Client *oCalendar = new TsgcHTTPGoogleCloud_Calendar_Client(NULL);
oCalendar->TLSOptions->IOHandler = iohOpenSSL;
oCalendar->TLSOptions->Version = tls1_3;
oCalendar->TLSOptions->VerifyCertificate = true;
oCalendar->TLSOptions->OpenSSL_Options->LibPath = oslpDefaultFolder;
```

## Najczęstsze zastosowania

- **Konfiguracja**
  - [Token odświeżania Google Calendar](#)
  - [Konto usługi Google Calendar](#)
- **Synchronizacja**
  - [Synchronizacja kalendarzy Google Calendar](#)
  - [Synchronizacja zdarzeń Google Calendar](#)

## Synchronizacja kalendarzy

Komponent TsgcHTTPGoogleCloud\_Calendar\_Client umożliwia synchronizację kalendarzy za pomocą bezpośrednich wywołań Google API lub wygodnych metod Calendars do synchronizacji kalendarzy.

|                      |                                                                    |                                                                               |
|----------------------|--------------------------------------------------------------------|-------------------------------------------------------------------------------|
| NewCalendar          | <b>aSummary:</b> tytuł kalendarza.                                 | Tworzy nowy kalendarz                                                         |
| DeleteCalendar       | <b>aId:</b> identyfikator kalendarza.                              | Usuwa istniejący kalendarz.                                                   |
| UpdateCalendar       | <b>aResource:</b> obiekt z danymi kalendarza.                      | Aktualizuje istniejący Kalendarz.                                             |
| LoadCalendars        |                                                                    | Ładuje wszystkie kalendarze i wypełnia właściwość Calendars pobranymi danymi. |
| LoadCalendarsChanged | <b>aSyncToken:</b> ostatni token użyty do aktualizacji kalendarza. | Wczytuje wszystkie zmiany w kalendarzach z ustawionego tokena.                |

Klient kalendarza posiada właściwość o nazwie **Calendars**, za pośrednictwem której można uzyskać dostęp do danych kalendarza po wywołaniu dowolnej z poprzednich metod. Właściwość ta jest synchronizowana automatycznie.



## Synchronizacja zdarzeń

Komponent `TsgcHTTPGoogleCloud_Calendar_Client` umożliwia synchronizację zdarzeń za pomocą bezpośrednich wywołań Google API lub za pomocą naszych uproszczonych metod zdarzeń.

|                   |                                                                                                                     |                                                             |
|-------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| NewEvent          | <b>aCalendarId:</b> identyfikator kalendarza.<br><b>aResource:</b> obiekt z danymi zdarzenia.                       | Tworzy nowe zdarzenie.                                      |
| DeleteEvent       | <b>aCalendarId:</b> identyfikator kalendarza.<br><b>aId:</b> identyfikator zdarzenia.                               | Usuwa istniejące zdarzenie.                                 |
| UpdateEvent       | <b>aCalendarId:</b> identyfikator kalendarza.<br><b>aResource:</b> obiekt z danymi zdarzenia.                       | Aktualizuje istniejące zdarzenie.                           |
| LoadEvents        | <b>aCalendarId:</b> identyfikator kalendarza.                                                                       | Wczytuje wszystkie zdarzenia kalendarza.                    |
| LoadEventsChanged | <b>aCalendarId:</b> identyfikator kalendarza.<br><b>aSyncToken:</b> ostatni token użyty do aktualizacji kalendarza. | Ładuje wszystkie zdarzenia kalendarza z ustawionego tokenu. |

Dostęp do danych zdarzeń można uzyskać za pomocą właściwości **Calendars**. Należy wybrać dowolny z istniejących kalendarzy z listy i uzyskać dostęp do jego właściwości **Events**.

## Wywołania Google Calendar API

|            |                                                                        |
|------------|------------------------------------------------------------------------|
| ACL_Delete | Usuwa regułę kontroli dostępu.                                         |
| ACL_Get    | Zwraca regułę kontroli dostępu.                                        |
| ACL_Insert | Tworzy regułę kontroli dostępu.                                        |
| ACL_List   | Zwraca reguły z listy kontroli dostępu dla kalendarza.                 |
| ACL_Patch  | Aktualizuje regułę kontroli dostępu. Metoda obsługuje semantykę patch. |

|            |                                      |
|------------|--------------------------------------|
| ACL_Update | Aktualizuje regułę kontroli dostępu. |
| ACL_Watch  | Obserwuj zmiany zasobów ACL.         |

|                     |                                                                                                         |
|---------------------|---------------------------------------------------------------------------------------------------------|
| CalendarList_Delete | Usuwa kalendarz z listy kalendarzy użytkownika.                                                         |
| CalendarList_Get    | Zwraca kalendarz z listy kalendarzy użytkownika.                                                        |
| CalendarList_Insert | Wstawia istniejący kalendarz na listę kalendarzy użytkownika.                                           |
| CalendarList_List   | Zwraca kalendarze z listy kalendarzy użytkownika.                                                       |
| CalendarList_Patch  | Aktualizuje istniejący kalendarz na liście kalendarzy użytkownika. Ta metoda obsługuje semantykę patch. |
| CalendarList_Update | Aktualizuje istniejący kalendarz na liście kalendarzy użytkownika.                                      |
| CalendarList_Watch  | Obserwuj zmiany w zasobach CalendarList.                                                                |

|                 |                                                                                                                          |
|-----------------|--------------------------------------------------------------------------------------------------------------------------|
| Calendar_Clear  | Czyści główny kalendarz. Operacja ta usuwa wszystkie zdarzenia powiązane z głównym kalendarzem konta.                    |
| Calendar_Delete | Usuwa pomocniczy kalendarz. Do wyczyszczenia wszystkich zdarzeń w kalendarzach podstawowych należy użyć calendars.clear. |
| Calendar_Get    | Zwraca metadane kalendarza.                                                                                              |
| Calendar_Insert | Tworzy pomocniczy kalendarz.                                                                                             |
| Calendar_Patch  | Aktualizuje metadane kalendarza. Ta metoda obsługuje semantykę patch.                                                    |
| Calendar_Update | Aktualizuje metadane kalendarza.                                                                                         |

|              |                                                 |
|--------------|-------------------------------------------------|
| Channel_Stop | Zatrzymaj obserwowanie zasobów przez ten kanał. |
|--------------|-------------------------------------------------|

|           |                                                    |
|-----------|----------------------------------------------------|
| Color_Get | Zwraca definicje kolorów dla kalendarzy i zdarzeń. |
|-----------|----------------------------------------------------|

|                 |                                                                                                                                                                                                                                                                                           |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Event_Delete    | Usuwa zdarzenie.                                                                                                                                                                                                                                                                          |
| Event_Get       | Zwraca zdarzenie.                                                                                                                                                                                                                                                                         |
| Event_Import    | Importuje zdarzenie. Operacja ta służy do dodawania prywatnej kopii istniejącego zdarzenia do kalendarza.                                                                                                                                                                                 |
| Event_Insert    | Tworzy zdarzenie.                                                                                                                                                                                                                                                                         |
| Event_Instances | Zwraca instancje określonego zdarzenia cyklicznego.                                                                                                                                                                                                                                       |
| Event_List      | Zwraca zdarzenia z określonego kalendarza.                                                                                                                                                                                                                                                |
| Event_Move      | Przenosi zdarzenie do innego kalendarza, tj. zmienia organizatora zdarzenia.                                                                                                                                                                                                              |
| Event_Patch     | Aktualizuje zdarzenie. Ta metoda obsługuje semantykę łatkowania. Podane wartości pól zastępują istniejące wartości. Pola, które nie zostały określone w żądaniu, pozostają bez zmian. Pola tablicowe, jeśli zostały podane, nadpisują istniejące tablice, odrzucając poprzednie elementy. |
| Event_QuickAdd  | Tworzy zdarzenie na podstawie prostego ciągu tekstowego.                                                                                                                                                                                                                                  |
| Event_Update    | Aktualizuje zdarzenie.                                                                                                                                                                                                                                                                    |
| Event_Watch     | Obserwowanie zmian zasobów Events.                                                                                                                                                                                                                                                        |

Zwraca informacje o dostępności (wolny/zajęty) dla zestawu kalendarzy.

|                |                                                                            |
|----------------|----------------------------------------------------------------------------|
| Settings_Get   | Zwraca pojedyncze ustawienie użytkownika.                                  |
| Settings_List  | Zwraca wszystkie ustawienia użytkownika dla uwierzytelnionego użytkownika. |
| Settings_Watch | Obserwuj zmiany zasobów Settings.                                          |

## Przełącz konto

Aby przełączyć się z jednego konta na inne, należy najpierw wywołać metodę **Clear**, aby wyczyścić bieżącą sesję kalendarza i skonfigurować nową sesję.

# Google Calendar | Wczytywanie kalendarzy

Pobieranie wszystkich kalendarzy z konta jest bardzo proste. Wystarczy wykonać następujące kroki:

1. Należy wywołać metodę **LoadCalendars**.
2. Jeśli metoda zwraca True, można uzyskać dostęp do właściwości **Calendars** i iterować po liście, aby uzyskać dostęp do wszystkich kalendarzy.

```
TsgcHTTPGoogleCloud_Calendar_Client oGoogleCalendar = new TsgcHTTPGoogleCloud_Calendar_Client(NULL);
// ... configure OAuth2 options
oGoogleCalendar->GoogleCloudOptions->OAuth2->ClientId = "google ClientId";
oGoogleCalendar->GoogleCloudOptions->OAuth2->ClientSecret = "google ClientSecret";
// ... request calendars
if (oGoogleCalendar->LoadCalendars)
{
    // ... get calendars data
    for (int i = 0; i < oGoogleCalendar->Calendars->Count; i++)
    {
        vCalendarTitle = oGoogleCalendar->Calendars->Calendar[i]->Summary;
    }
}
else
{
    throw Exception("Error Calendar Sync");
}
```

# Google Calendar | Synchronizacja zdarzeń

Proces pobierania wszystkich zdarzeń z kalendarza jest bardzo prosty. Wystarczy wykonać kolejne kroki:

1. Wywołaj metodę **LoadEvents** i przekaż **CalendarId** jako parametr.
2. Jeśli metoda zwróci True, można uzyskać dostęp do właściwości **Calendars.Events** i iterować po liście, aby uzyskać dostęp do wszystkich zdarzeń kalendarza.

```
TsgcHTTPGoogleCloud_Calendar_Client oGoogleCalendar = new TsgcHTTPGoogleCloud_Calendar_Client(NULL);
// ... configure OAuth2 options
oGoogleCalendar->GoogleCloudOptions->OAuth2->ClientId = "google ClientId";
oGoogleCalendar->GoogleCloudOptions->OAuth2->ClientSecret = "google ClientSecret";
// ... request calendars first;
oGoogleCalendar->LoadCalendars;
// ... request events from first calendar
oCalendar = TsgcGoogleCalendarItem(oGoogleCalendar->Calendars->Calendar[0]);
if (oGoogleCalendar->LoadEvents(oCalendar->ID))
{
    // ... get events data
    for (int i = 0; i < oCalendar->Events->Count; i++)
    {
        vEventTitle = oCalendar->Events[i]->Summary;
    }
}
else
{
    throw Exception("Error Event Sync");
}
```

# Google Calendar | RefreshToken

Google Calendar API używa OAuth2 do uwierzytelniania na serwerach Google. sgcWebSockets posiada komponent obsługujący cały proces uwierzytelniania, ale jeśli aplikacja zostanie zamknięta i zostanie podjęta próba ponownego połączenia, dostępne są dwie opcje:

1. Uwierzytelnij się ponownie za pomocą swoich Google APIs
2. Użyj tokenu odświeżania (jeśli jest nadal ważny), aby uniknąć procesu uwierzytelniania.

## Używanie RefreshToken

Przy pierwszym uwierzytelnieniu użyj zdarzenia OnAuthToken, aby zapisać **RefreshToken**, jeśli istnieje. Można go zapisać na przykład w pliku INI:

```
void OnGoogleCalendarAuthToken(TObject *Sender, string TokenType, string Token, string Data)
{
    TsgcJSON *oJSON = new TsgcJSON();
    try
    {
        oJSON->Read(Data);
        if (oJSON->Node["refresh_token"] != NULL)
        {
            TINIFile *oINI = new TINIFile(ChangeFileExt(Application->ExeName, ".ini"));
            try
            {
                oINI->WriteString("OAUTH2", "Token", oJSON->Node["refresh_token"]->Value);
            }
            __finally
            {
                oINI->Free();
            }
        }
    }
    __finally
    {
        oJSON->Free();
    }
}
```

Następnie, po ponownym uruchomieniu aplikacji, jeśli istnieje RefreshToken, należy wywołać metodę RefreshToken i przekazać token jako argument (najpierw należy ustawić klucze API Google Calendar). W przypadku powodzenia nastąpi zalogowanie do serwerów Google bez konieczności ponownego uwierzytelniania.

```
GoogleCalendar->RefreshToken("your refresh token here");
```

## Google Calendar | Konto usługi

Klient Google Calendar może działać jako usługa bez interakcji użytkownika, co jest przydatne przy uruchamianiu usługi Windows, demonów itp.

Google Cloud wymaga utworzenia **konta usługi** (zamiast poświadczeń OAuth2) w celu uruchomienia tego typu projektu, a Google Calendar API wymaga, aby konto usługi korzystało z **delegowania w całej domenie** w celu uzyskania wymaganych uprawnień dostępu do kalendarzy.

Więcej informacji na temat tworzenia [kont usługi Google](#) można znaleźć w dokumentacji.

Po skonfigurowaniu konta Google Cloud z kontem usługi i powiązaniu go z kontem e-mail w obszarze roboczym przy użyciu delegowania w całej domenie można skonfigurować klienta Google Calendar do pracy z tym kontem, wykonując następujące kroki:

- Ustaw właściwość **GoogleCloudOptions.Authentication** na wartość **gcaJWT**.
- Zaimportuj plik JSON wygenerowany na koncie Google Cloud za pomocą metody **LoadSettingsFromFile**. Plik ten zawiera klucz prywatny do szyfrowania JWT oraz inne właściwości wymagane przez klienta.
- Po zaimportowaniu pliku JSON następujące właściwości są wypełniane automatycznie:
  - **ClientEmail**: nazwa konta usługi
  - **PrivateKeyId**: identyfikator pliku klucza prywatnego
  - **PrivateKey**: plik klucza prywatnego
- Na koniec należy ustawić we właściwości **GoogleCloudOptions.JWT.Subject** adres e-mail konta Workspace powiązanego z kontem usługi.

Po skonfigurowaniu klienta można rozpocząć wysyłanie żądań do Google Calendar API bez interakcji użytkownika.



# Google Cloud FCM

Firestore Cloud Messaging (FCM) to wieloplatformowe rozwiązanie do przesyłania wiadomości umożliwiające niezawodne wysyłanie wiadomości bez żadnych kosztów.

Korzystając z FCM, można powiadamiać aplikację kliencką o dostępności nowych wiadomości e-mail lub innych danych do synchronizacji. Możliwe jest wysyłanie wiadomości powiadomień w celu ponownego angażowania i zatrzymywania użytkowników. W przypadkach takich jak natychmiastowe przesyłanie wiadomości, wiadomość może przenosić ładunek do 4096 bajtów do aplikacji klienckiej.

Komponent obsługuje HTTP API v1.

## Autoryzacja

Komponent klienta Google FCM może logować się do serwerów Google przy użyciu następujących metod:

- **gcaOAuth2:** protokół OAuth2
- **gcaJWT:** tokeny JWT.

## OAuth2

Logowanie odbywa się za pośrednictwem przeglądarki internetowej, w której użytkownik loguje się na własnym koncie i autoryzuje żądania FCM.

- **GoogleCloudOptions.OAuth2.ClientId:** jest to ClientID dostarczony przez Google do uwierzytelniania przez protokół OAuth2.
- **GoogleCloudOptions.OAuth2.ClientSecret:** ciąg Client Secret dostarczony przez Google w celu uwierzytelniania przez protokół OAuth2.
- **GoogleCloudOptions.OAuth2.Scope:** zakres OAuth2; zazwyczaj nie ma potrzeby modyfikowania wartości domyślnej, chyba że wymagany jest szerszy dostęp niż domyślny.
- **GoogleCloudOptions.OAuth2.LocalIP:** protokół OAuth2 wymaga lokalnego serwera nasłuchującego na odpowiedź z serwera uwierzytelniania. Jest to adres IP lub nazwa DNS. Domyślnie wynosi 127.0.0.1.
- **GoogleCloudOptions.OAuth2.LocalPort:** lokalny port nasłuchiwanie serwera.
- **GoogleCloudOptions.OAuth2.RedirectURL:** jeśli wymagany jest adres URL przekierowania inny niż LocalPort + LocalIP, można go ustawić w tej właściwości (przykład: `http://127.0.0.1:8080/oauth2`).

## Konta usługowe

Logowanie odbywa się przez podpisywanie żądań przy użyciu klucza prywatnego dostarczonego przez Google. Ta metoda jest zalecana dla zautomatyzowanych usług lub aplikacji bez interakcji użytkownika.

- **GoogleCloudOptions.JWT.ClientEmail:** nazwa adresu e-mail klienta podana podczas tworzenia nowego konta usługi. Węzeł „client\_email” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.Subject:** adres e-mail klienta podany podczas tworzenia nowego konta usługi. Węzeł „client\_email” w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKeyId:** identyfikator klucza prywatnego dostarczony przez Google. Węzeł "private\_key\_id" w pliku konfiguracyjnym JSON.
- **GoogleCloudOptions.JWT.PrivateKey:** certyfikat klucza prywatnego dostarczony przez Google. Węzeł "private\_key" w pliku konfiguracyjnym JSON.

## Opcje TLS

Połączenia Firebase Cloud Messaging można dostroić za pomocą właściwości **TLSOptions**.

- **TLSOptions.IOHandler:** wybór stosu TLS, który ma być używany (OpenSSL, SChannel...).
- **TLSOptions.Version:** żądanie określonej wersji protokołu TLS wymaganej przez projekt.
- **TLSOptions.VerifyCertificate:** wymusza walidację certyfikatu serwera podczas łączenia się z Google.
- **TLSOptions.OpenSSL\_Options.LibPath:** określa lokalizację, w której wdrożono biblioteki środowiska uruchomieniowego OpenSSL.
- **TLSOptions.SChannel\_Options.UseLegacyCredentials:** należy włączyć, gdy Windows SChannel wymaga starszego negocjowania poświadczeń.

Przykładowa konfiguracja:

```
TsgcHTTPGoogleCloud_FCM_Client *oFCM = new TsgcHTTPGoogleCloud_FCM_Client(NULL);
oFCM->TLSOptions->IOHandler = iohOpenSSL;
oFCM->TLSOptions->Version = tls1_3;
oFCM->TLSOptions->VerifyCertificate = true;
oFCM->TLSOptions->OpenSSL_Options->LibPath = oslpDefaultFolder;
```

Po utworzeniu nowego konta usługi można pobrać plik JSON ze wszystkimi konfiguracjami. Plik ten może zostać przetworzony przez komponent FCM — wystarczy wywołać metodę **LoadSettingsFromFile** i przekazać nazwę pliku JSON jako argument.

## Najczęstsze zastosowania

- Konfiguracja
  - [Klucze Google OAuth2](#)
  - [Konta usługowe](#)

## Klient Google FCM

### OAuth2

Aby pracować z Google FCM API, komponent FCM `sgcWebSockets` używa OAuth2 jako domyślnej metody uwierzytelniania, dlatego najpierw należy ustawić **ClientId** i **ClientSecret** ze swojego konta Google.

```
oFCM = new TsgcHTTPGoogleCloud_FCM_Client();
oFCM->GoogleCloudOptions->Authorization = gcaOAuth2;
oFCM->GoogleCloudOptions->OAuth2->ClientId = "... your google client id...";
oFCM->GoogleCloudOptions->OAuth2->ClientSecret = "... your google client secret...";
```

## Konta usługowe

Konta usług wymagają zbudowania tokenu JWT i przekazania go jako tokenu autoryzacji.

```
TsgcHTTPGoogleCloud_FCM_Client *oFCM = new TsgcHTTPGoogleCloud_FCM_Client(this);
oFCM->GoogleCloudOptions->Authorization = gcaJWT;
oFCM->GoogleCloudOptions->JWT->ClientEmail = "...google email...";
oFCM->GoogleCloudOptions->JWT->Subject = "...google email...";
oFCM->GoogleCloudOptions->JWT->PrivateKeyId = "...private key id...";
oFCM->GoogleCloudOptions->JWT->PrivateKey->Lines->Text = "...private key certificate...";
```

Jest to wymagane w celu uzyskania tokenu autoryzacyjnego z Google, który będzie używany we wszystkich wywołaniach Rest API.

Wszystkie metody zwracają odpowiedź, która może być pomyślna lub zawierać błąd.

## Wyślij wiadomość

Należy użyć metody `SendMessage` do wysłania wiadomości powiadomienia. Funkcja ma 2 argumenty:

1. Project: to identyfikator projektu konta Google Cloud.
2. Payload: jest wiadomością tekstową w formacie JSON. Przykład:

```
{
  "message": {
    "topic": "news",
    "notification": {
      "title": "Breaking News",
      "body": "New news story available."
    },
    "data": {
      "story_id": "story_12345"
    }
  }
}
```

}  
}

# TsgcWebView2

TsgcWebView2 to wizualny komponent VCL opakowujący Microsoft Edge WebView2 (Chromium). Upuść go na formularz, aby osadzić nowoczesną, w pełni funkcjonalną przeglądarkę internetową w aplikacji Delphi. Obsługuje Delphi 7 do Delphi 13.

## Wymagania

- Microsoft Edge WebView2 Runtime (dołączony do systemu Windows 10/11 lub dostępny do pobrania od firmy Microsoft)
- WebView2Loader.dll (dołączony, należy umieścić obok pliku wykonywalnego)
- Tylko dla systemu Windows

## Szybki start

Upuść komponent TsgcWebView2 na formularzu i przejdź do adresu URL:

```
sgcWebView21 := TsgcWebView2.Create(Self);
sgcWebView21.Parent := Self;
sgcWebView21.Align := alClient;
sgcWebView21.DefaultURL := 'https://www.example.com';
```

## Funkcje

- **Nawigacja**
  - [Nawiguj](#), [NavigateToString](#), [GoBack](#), [GoForward](#), [Reload](#), [Stop](#), [nawigacja POST](#)
- **JavaScript**
  - [ExecuteScript \(async\)](#), [ExecuteScriptSync](#), [AddInitScript](#)
- **Cookie Management**
  - [GetCookies](#), [AddOrUpdateCookie](#), [DeleteCookie](#), [DeleteAllCookies](#)
- **Kontrola pobierania**
  - [OnDownloadStarting](#), [OnDownloadProgress](#), [OnDownloadCompleted](#)
- **Ustawienia i konfiguracja**
  - [ScriptEnabled](#), [DevToolsEnabled](#), [ContextMenuEnabled](#) i inne
- **Zaawansowane funkcje**
  - [Drukowanie](#), [dźwięk/wyciszenie](#), [certyfikaty](#), [favicon](#), [wirtualny host](#), [zrzut ekranu](#)

## Właściwości

- **DefaultURL**: string — URL, do którego następuje nawigacja po inicjalizacji.
- **AutoInitialize**: Boolean — gdy True, WebView2 jest tworzony automatycznie podczas alokowania uchwytu okna. Domyślnie: True.
- **ZoomFactor**: Double — poziom powiększenia (1.0 = 100%).
- **UserDataFolder**: string — ścieżka do folderu przechowywania plików cookie, pamięci podręcznej i uprawnień.
- **BrowserExecutableFolder**: string — ścieżka do dystrybucji środowiska uruchomieniowego WebView2 o stałej wersji.
- **Settings**: TsgcWebView2Settings — ustawienia przeglądarki w czasie projektowania (ScriptEnabled, DevToolsEnabled itp.).
- **URL**: string (tylko do odczytu) — bieżący adres URL strony.
- **DocumentTitle**: string (tylko do odczytu) — aktualny tytuł strony.
- **CanGoBack**: Boolean (tylko do odczytu) — informuje, czy możliwe jest cofnięcie nawigacji.
- **CanGoForward**: Boolean (tylko do odczytu) — określa, czy możliwa jest nawigacja do przodu.
- **Initialized**: Boolean (tylko do odczytu) — czy WebView2 jest gotowy.
- **IsMuted**: Boolean — wycisz/odcisz dźwięk.

- **IsDocumentPlayingAudio**: Boolean (tylko do odczytu) — określa, czy strona odtwarza dźwięk.
- **FaviconURI**: string (tylko do odczytu) — URI ikony strony favicon bieżącej strony.
- **StatusBarText**: string (tylko do odczytu) — bieżący tekst paska stanu.
- **CookieManager**: TsgcWebView2CookieManager (tylko do odczytu) — obiekt zarządzania plikami cookie.
- **WebView**: ICoreWebView2 (tylko do odczytu) — bezpośredni dostęp do interfejsu COM.
- **Controller**: ICoreWebView2Controller (tylko do odczytu) — bezpośredni dostęp do interfejsu COM.
- **Środowisko**: ICoreWebView2Environment (tylko do odczytu) — bezpośredni dostęp do interfejsu COM.

## Metody

- **Navigate**(aURL: string) — przejdź do adresu URL.
- **NavigateToString**(aHTML: string) — załaduj treść HTML bezpośrednio.
- **GoBack** / **GoForward** — nawigacja po historii.
- **Reload** / **Stop** — przeładuj lub zatrzymaj ładowanie.
- **ExecuteScript**(aScript: string) — wykonuje skrypt JavaScript asynchronicznie (wynik w OnScriptExecuted).
- **ExecuteScriptSync**(aScript: string): string — synchroniczne wykonanie kodu JavaScript, zwraca wynik w formacie JSON.
- **AddInitScript**(aScript: string) — dodaje skrypt JavaScript wykonywany przy każdym wczytaniu strony.
- **RemoveInitScript**(aScriptId: string) — usuwa wcześniej dodany skrypt inicjalizacyjny.
- **NavigateWithPostData**(aURI, aMethod, aPostData, aHeaders: string) — nawigacja z niestandardową metodą HTTP, treścią i nagłówkami.
- **PrintToPdf**(aFilePath: string) — zapisuje stronę jako plik PDF.
- **ShowPrintUI** — wyświetla natywne okno dialogowe drukowania.
- **CapturePreviewToFile**(aFilePath: string; aFormat: Integer) — zrzut ekranu (0=PNG, 1=JPEG).
- **OpenDevToolsWindow** — otwórz narzędzia deweloperskie Edge.
- **OpenTaskManagerWindow** — otwórz menedżer zadań Edge.
- **SetVirtualHostNameToFolderMapping**(aHostName, aFolderPath: string; aAccessKind: Integer) — mapuje nazwę hosta na folder lokalny.
- **ClearVirtualHostNameToFolderMapping**(aHostName: string) — usuwa mapowanie.
- **PostWebMessageAsString**(aMessage: string) — wyślij wiadomość do zawartości webowej.
- **PostWebMessageAsJson**(aJson: string) — wysła JSON do treści strony internetowej.
- **GetProfileName**: string — pobiera nazwę profilu przeglądarki.
- **ClearBrowsingData**(aKinds: Cardinal) — czyści określone typy danych przeglądania.
- **ClearAllBrowsingData** — wyczyść wszystkie dane przeglądania.
- **PostSharedBufferToScript**(aSharedBuffer: IUnknown; aAccess: Integer; aAdditionalDataAsJson: string) — udostępnia bufor pamięci treściami webowym.
- **InitializeWebView** / **FinalizeWebView** — ręczne sterowanie inicjalizacją.

## Zdarzenia

- **OnInitialized** — wywoływane, gdy WebView2 jest gotowy.
- **OnNavigationStarting**(aURI; aIsUserInitiated, aIsRedirected: Boolean; var aCancel: Boolean) — przed nawigacją.
- **OnNavigationCompleted**(aIsSuccess: Boolean; aWebErrorStatus: Integer) — po zakończeniu nawigacji.
- **OnSourceChanged**(aIsNewDocument: Boolean) — zmieniony adres URL.
- **OnDocumentTitleChanged**(aTitle: string) — tytuł strony uległ zmianie.
- **OnWebMessageReceived**(aSource, aWebMessageAsJson, aWebMessageAsString: string) — wiadomość z zawartości webowej.
- **OnNewWindowRequested**(aURI: string; aIsUserInitiated: Boolean; var aHandled: Boolean) — żądanie wyskakującego okna lub nowej karty.
- **OnScriptExecuted**(aErrorCode: HRESULT; aResultAsJson: string) — asynchroniczny wynik JS.
- **OnContentLoading**(aIsErrorPage: Boolean) — rozpoczęto ładowanie zawartości.
- **OnHistoryChanged** — historia nawigacji uległa zmianie.
- **OnProcessFailed**(aKind: Integer) — proces przeglądarki uległ awarii.
- **OnPermissionRequested**(aURI: string; aKind: Integer; aIsUserInitiated: Boolean; var aState: Integer) — żądanie uprawnień.
- **OnWindowCloseRequested** — wywołano window.close().
- **OnDownloadStarting**(aURI, aResultFilePath: string; var aCancel, aHandled: Boolean; var aFilePath: string) — rozpoczęto pobieranie.
- **OnDownloadProgress**(aBytesReceived, aTotalBytes: Int64) — postęp pobierania.
- **OnDownloadCompleted**(aFilePath: string; aState: Integer) — pobieranie zakończone.
- **OnContextMenuRequested**(aMenuItems: string; aContextKind: Integer; aLocation: TPoint; var aHandled: Boolean) — menu kontekstowe (prawy przycisk myszy).
- **OnFaviconChanged**(aFaviconURI: string) — ikona strony uległa zmianie.
- **OnClientCertificateRequested**(aHost: string; aPort: Integer; var aHandled: Boolean) — wymagany certyfikat klienta.

- **OnServerCertificateError**(aRequestURI: string; aErrorStatus: Integer; var aAction: Integer) — błąd certyfikatu.
- **OnStatusBarTextChanged**(aText: string) — zmieniono tekst paska stanu.
- **OnDOMContentLoaded** — DOM gotowy.
- **OnBasicAuthRequested**(aURI: string; var aUserName, aPassword: string; var aHandled: Boolean) — uwierzytelnianie podstawowe HTTP.
- **OnCapturePreviewCompleted**(aErrorCode: HRESULT) — zrzut ekranu zakończony.

# TsgcWebView2 | Nawigacja

TsgcWebView2 udostępnia kilka metod nawigacji do adresów URL, ładowania zawartości HTML i kontrolowania historii przeglądarki.

## Podstawowa nawigacja

Należy użyć metody **Navigate**, aby załadować adres URL. Komponent wywołuje `OnNavigationStarting` przed żądaniem i `OnNavigationCompleted` po jego zakończeniu.

```
// Navigate to a URL
sgcWebView21.Navigate('https://www.example.com');
```

Właściwość **DefaultURL** można również ustawić w czasie projektowania lub przed inicjalizacją. Komponent automatycznie przechodzi do tego adresu URL po gotowości WebView2.

```
// Set the default URL before initialization
sgcWebView21.DefaultURL := 'https://www.example.com';
```

## Zawartość HTML

Użyj metody **NavigateToString**, aby załadować zawartość HTML bezpośrednio bez serwera lub pliku.

```
// Load HTML content directly
sgcWebView21.NavigateToString(
  '<html><body><h1>Hello from Delphi</h1>' +
  '<p>This content was loaded with NavigateToString.</p>' +
  '</body></html>');
```

## Nawigacja historyczna

Użyj metod **GoBack** i **GoForward**, aby nawigować po historii przeglądarki. Przed wywołaniem tych metod należy sprawdzić właściwości **CanGoBack** i **CanGoForward**.

```
// Navigate back
if sgcWebView21.CanGoBack then
  sgcWebView21.GoBack;

// Navigate forward
if sgcWebView21.CanGoForward then
  sgcWebView21.GoForward;
```

Należy użyć **Reload**, aby odświeżyć bieżącą stronę, oraz **Stop**, aby anulować oczekującą nawigację.

```
// Reload the current page
sgcWebView21.Reload;

// Stop loading
sgcWebView21.Stop;
```



## Nawigacja POST

Należy użyć metody **NavigateWithPostData** w celu wysłania żądania HTTP z niestandardową metodą, treścią i nagłówkami. Jest to przydatne do przesyłania danych formularza lub wywoływania REST API bezpośrednio w przeglądarce.

```
// POST form data with custom headers
sgcWebView21.NavigateWithPostData(
    'https://api.example.com/login',
    'POST',
    'username=admin&password=secret',
    'Content-Type: application/x-www-form-urlencoded');

// POST JSON data
sgcWebView21.NavigateWithPostData(
    'https://api.example.com/data',
    'POST',
    '{"name":"John","age":30}',
    'Content-Type: application/json');
```

## Zdarzenia nawigacji

Należy użyć **OnNavigationStarting**, aby sprawdzić lub anulować nawigację przed jej rozpoczęciem. Ustawienie **aCancel** na True blokuje żądanie.

```
procedure TFormMain.sgcWebView21NavigationStarting(Sender: TObject;
    const aURI: string; aIsUserInitiated, aIsRedirected: Boolean;
    var aCancel: Boolean);
begin
    // Block navigation to unwanted domains
    if Pos('ads.example.com', aURI) > 0 then
        aCancel := True;
end;
```

Użyj zdarzenia **OnNavigationCompleted**, aby sprawdzić, czy nawigacja zakończyła się sukcesem czy niepowodzeniem.

```
procedure TFormMain.sgcWebView21NavigationCompleted(Sender: TObject;
    aIsSuccess: Boolean; aWebErrorStatus: Integer);
begin
    if aIsSuccess then
        StatusBar1.SimpleText := 'Page loaded: ' + sgcWebView21.URL
    else
        StatusBar1.SimpleText := 'Navigation failed, error: ' +
            IntToStr(aWebErrorStatus);
end;
```

## Obsługa nowego okna

Gdy strona otwiera wyskakujące okno lub link z parametrem `target="_blank"`, wywoływane jest zdarzenie **OnNewWindowRequested**. Należy ustawić **aHandled** na True, aby zapobiec domyślnemu wyskakującemu oknu i nawigować w tym samym oknie.

```
procedure TFormMain.sgcWebView21NewWindowRequested(Sender: TObject;
    const aURI: string; aIsUserInitiated: Boolean;
    var aHandled: Boolean);
begin
```

```
// Navigate in the same window instead of opening a popup
aHandled := True;
sgcWebView21.Navigate(aURI);
end;
```

# TsgcWebView2 | JavaScript

TsgcWebView2 udostępnia metody umożliwiające wykonywanie kodu JavaScript w kontekście przeglądarki, pobieranie wyników oraz ustanawianie dwukierunkowej komunikacji między aplikacją Delphi a treścią internetową.

## Wykonywanie asynchroniczne

Użyj **ExecuteScript**, aby asynchronicznie wykonać JavaScript. Wynik jest zwracany w zdarzeniu **OnScriptExecuted** jako ciąg JSON.

```
// Execute JavaScript asynchronously
sgcWebView21.ExecuteScript('document.title');

procedure TFormMain.sgcWebView21ScriptExecuted(Sender: TObject;
  aErrorCode: HRESULT; const aResultAsJson: string);
begin
  if aErrorCode = S_OK then
    ShowMessage('Result: ' + aResultAsJson)
  else
    ShowMessage('Script error: ' + IntToStr(aErrorCode));
end;
```

Można wykonać dowolne prawidłowe wyrażenie JavaScript. Zwracana wartość jest zawsze serializowana jako JSON.

```
// Get the number of links on the page
sgcWebView21.ExecuteScript('document.querySelectorAll("a").length');

// Modify page content
sgcWebView21.ExecuteScript(
  'document.body.style.backgroundColor = "lightyellow"');
```

## Wykonanie synchroniczne

Należy użyć metody **ExecuteScriptSync** do blokującego wykonania, które zwraca wynik bezpośrednio. Jest to prostsze rozwiązanie, gdy wartość jest potrzebna natychmiast, jednak blokuje wątek wywołujący do czasu zakończenia skryptu.

```
var
  vResult: string;
begin
  // Get the page title synchronously
  vResult := sgcWebView21.ExecuteScriptSync('document.title');
  ShowMessage('Title: ' + vResult);

  // Get form field value
  vResult := sgcWebView21.ExecuteScriptSync(
    'document.getElementById("email").value');
  ShowMessage('Email: ' + vResult);
end;
```

## Skrypty inicjalizacyjne

Należy użyć metody **AddInitScript**, aby zarejestrować JavaScript uruchamiany automatycznie przy każdym ładowaniu strony, przed innymi skryptami. Jest to przydatne do wstrzykiwania polyfillów, nadpisywania API przeglądarki lub konfigurowania procedur obsługi wiadomości.

```
// Add a script that runs on every page load
sgcWebView21.AddInitScript(
    'window.addEventListener("DOMContentLoaded", function() {' +
    '  console.log("Page loaded at: " + new Date().toISOString());' +
    '});});

// Override window.alert to send messages to Delphi
sgcWebView21.AddInitScript(
    'window.alert = function(msg) {' +
    '  window.chrome.webview.postMessage(msg);' +
    '});');
```

Użyj metody **RemoveInitScript**, aby wyrejestrować wcześniej dodany skrypt inicjalizacyjny na podstawie jego identyfikatora.

## Komunikacja webowa

Web messaging zapewnia dwukierunkową komunikację między aplikacją Delphi a kodem JavaScript działającym w przeglądarce. Należy użyć **PostWebMessageAsString** lub **PostWebMessageAsJson**, aby wysyłać dane z Delphi do JavaScript, oraz obsługiwać **OnWebMessageReceived**, aby odbierać dane z JavaScript.

### Wysyłanie wiadomości z Delphi do JavaScript:

```
// Send a plain string message
sgcWebView21.PostWebMessageAsString('Hello from Delphi!');

// Send a JSON message
sgcWebView21.PostWebMessageAsJson('{"action":"refresh","id":42}');
```

### Odbieranie wiadomości w JavaScript:

```
// Add an init script to listen for messages from Delphi
sgcWebView21.AddInitScript(
    'window.chrome.webview.addEventListener("message", function(e) {' +
    '  console.log("Received from Delphi: " + e.data);' +
    '});});
```

### Odbieranie wiadomości z JavaScript w Delphi:

```
procedure TFormMain.sgcWebView21WebMessageReceived(Sender: TObject;
    const aSource, aWebMessageAsJson, aWebMessageAsString: string);
begin
    // aWebMessageAsString contains the plain text message
    // aWebMessageAsJson contains the JSON-serialized message
    Memo1.Lines.Add('Message from web: ' + aWebMessageAsString);
end;
```

Z poziomu JavaScript wysyłaj wiadomości do Delphi za pomocą:

```
// JavaScript code inside the web page:  
// window.chrome.webview.postMessage('Hello from JavaScript!');  
// window.chrome.webview.postMessage({action: 'save', data: [1,2,3]});
```

# TsgcWebView2 | Zarządzanie plikami cookie

TsgcWebView2 udostępnia właściwość **CookieManager** do odczytu, tworzenia, aktualizowania i usuwania plików cookie w kontekście przeglądarki.

## Pobieranie ciasteczek

Metody **CookieManager.GetCookies** należy użyć do pobrania ciasteczek dla danego URI. Metoda zwraca listę obiektów ciasteczek, po której można iterować.

```
procedure TFormMain.ButtonGetCookiesClick(Sender: TObject);
var
  vCookies: TsgcWebView2CookieList;
  i: Integer;
begin
  vCookies := sgcWebView21.CookieManager.GetCookies(
    'https://www.example.com');
  try
    for i := 0 to vCookies.Count - 1 do
      Memo1.Lines.Add(
        vCookies[i].Name + ' = ' + vCookies[i].Value);
  finally
    vCookies.Free;
  end;
end;
```

## Dodawanie plików Cookie

Użyj **CookieManager.AddOrUpdateCookie**, aby utworzyć nowe ciasteczko lub zaktualizować istniejące. Należy co najmniej podać nazwę, wartość, domenę i ścieżkę.

```
// Add a session cookie
sgcWebView21.CookieManager.AddOrUpdateCookie(
  'session_id',           // name
  'abc123def456',        // value
  '.example.com',        // domain
  '/');                  // path

// Add a cookie with an expiration date
sgcWebView21.CookieManager.AddOrUpdateCookie(
  'preferences',          // name
  'theme=dark',           // value
  '.example.com',         // domain
  '/',                    // path
  Now + 30);              // expires in 30 days
```

## Usuwanie plików cookie

Należy użyć **DeleteCookie** do usunięcia konkretnego pliku cookie według nazwy i URI. Należy użyć **DeleteAllCookies** do wyczyszczenia wszystkich plików cookie z przeglądarki.

```
// Delete a specific cookie
sgcWebView21.CookieManager.DeleteCookie(
  'session_id',
  'https://www.example.com');
```

```
// Delete cookies matching a domain and path
sgcWebView21.CookieManager.DeleteCookiesWithDomainAndPath(
    'session_id',
    '.example.com',
    '/');
```

```
// Delete all cookies
sgcWebView21.CookieManager.DeleteAllCookies;
```

# TsgcWebView2 | Kontrola pobierania

TsgcWebView2 udostępnia zdarzenia umożliwiające przechwytywanie, monitorowanie i sterowanie pobieraniem plików inicjowanym przez przeglądarkę.

## Zdarzenia pobierania

Zdarzenie **OnDownloadStarting** jest wywoływane po rozpoczęciu pobierania. Można anulować pobieranie, zmienić docelową ścieżkę pliku lub pozwolić, aby przebiegło zgodnie z domyślnym zachowaniem.

```
procedure TFormMain.sgcWebView2DownloadStarting(Sender: TObject;
  const aURI, aResultFilePath: string;
  var aCancel, aHandled: Boolean; var aFilePath: string);
begin
  // Log the download
  Memo1.Lines.Add('Download starting: ' + aURI);
  Memo1.Lines.Add('Default path: ' + aResultFilePath);

  // Cancel downloads of .exe files
  if Pos('.exe', LowerCase(aURI)) > 0 then
  begin
    aCancel := True;
    ShowMessage('Executable downloads are blocked.');
```

## Postęp pobierania

Zdarzenie **OnDownloadProgress** jest wywoływane okresowo podczas pobierania, raportując odebrane bajty i łączną oczekiwaną liczbę bajtów.

```
procedure TFormMain.sgcWebView2DownloadProgress(Sender: TObject;
  aBytesReceived, aTotalBytes: Int64);
begin
  if aTotalBytes > 0 then
    ProgressBar1.Position :=
      Round((aBytesReceived / aTotalBytes) * 100)
  else
    ProgressBar1.Position := 0;

  LabelStatus.Caption := Format('Downloaded %d of %d bytes',
    [aBytesReceived, aTotalBytes]);
end;
```

## Pobieranie zakończone

Zdarzenie **OnDownloadCompleted** jest wywoływane po zakończeniu pobierania. Należy sprawdzić wartość parametru **aState**, aby ustalić, czy pobieranie zakończyło się sukcesem, zostało anulowane czy zakończyło się błędem.

```
procedure TFormMain.sgcWebView2DownloadCompleted(Sender: TObject;
  const aFilePath: string; aState: Integer);
begin
  case aState of
    0: // Completed
      ShowMessage('Download complete: ' + aFilePath);
    1: // Cancelled
```



```
        ShowMessage('Download was cancelled.');
```

2: // Failed

```
        ShowMessage('Download failed.');
```

end;

end;

## Niestandardowa ścieżka pobierania

Należy ustawić parametr **aFilePath** w OnDownloadStarting, aby przekierować pobieranie do niestandardowej lokalizacji. Należy ustawić **aHandled** na True, aby pominąć domyślne okno dialogowe zapisywania.

```
procedure TFormMain.sgWebView21DownloadStarting(Sender: TObject;
  const aURI, aResultFilePath: string;
  var aCancel, aHandled: Boolean; var aFilePath: string);
begin
  // Redirect all downloads to a custom folder
  aFilePath := 'C:\Downloads\' + ExtractFileName(aResultFilePath);
  aHandled := True; // suppress the default save dialog
end;
```

# TsgcWebView2 | Ustawienia

TsgcWebView2 udostępnia właściwość **Settings** typu TsgcWebView2Settings, która steruje zachowaniem przeglądarki. Ustawienia te można konfigurować w czasie projektowania lub zmieniać w czasie wykonywania po inicjalizacji.

## TsgcWebView2Settings Właściwości

- **ScriptEnabled**: Boolean — włącza lub wyłącza wykonywanie JavaScript. Domyślnie: True.
- **WebMessageEnabled**: Boolean — włącz lub wyłącz komunikację sieciową (postMessage). Domyślnie: True.
- **DefaultScriptDialogsEnabled**: Boolean — włącza lub wyłącza domyślne okna dialogowe JavaScript (alert, confirm, prompt). Domyślnie: True.
- **StatusBarEnabled**: Boolean — pokazuje lub ukrywa pasek stanu przeglądarki. Domyślnie: True.
- **DevToolsEnabled**: Boolean — zezwala lub blokuje dostęp do narzędzi Edge DevTools. Domyślnie: True.
- **ContextMenuEnabled**: wartość logiczna — włącza lub wyłącza menu kontekstowe prawego przycisku myszy przeglądarki. Domyślnie: True.
- **ZoomControlEnabled**: Boolean — zezwól lub zablokuj powiększanie inicjowane przez użytkownika (Ctrl+scroll, Ctrl+plus/minus). Domyślnie: True.
- **BuiltInErrorPageEnabled**: Boolean — pokazuje lub ukrywa wbudowaną stronę błędów dla błędów nawigacji. Domyślnie: True.

## Konfiguracja podczas projektowania

Ustaw właściwości Settings w Inspektorze obiektów. Są one stosowane automatycznie po zainicjalizowaniu środowiska WebView2.

```
// These values can also be set in the Object Inspector
sgcWebView21.Settings.ScriptEnabled := True;
sgcWebView21.Settings.DevToolsEnabled := False;
sgcWebView21.Settings.ContextMenuEnabled := False;
sgcWebView21.Settings.ZoomControlEnabled := False;
```

## Konfiguracja środowiska uruchomieniowego

Ustawienia można zmieniać w czasie działania aplikacji po inicjalizacji komponentu. Zmiany są natychmiastowo stosowane.

```
procedure TFormMain.sgcWebView21Initialized(Sender: TObject);
begin
    // Disable right-click menu and DevTools at runtime
    sgcWebView21.Settings.ContextMenuEnabled := False;
    sgcWebView21.Settings.DevToolsEnabled := False;

    // Disable JavaScript dialogs
    sgcWebView21.Settings.DefaultScriptDialogsEnabled := False;
end;
```

## UserDataFolder

Właściwość **UserDataFolder** określa katalog, w którym przeglądarka przechowuje pliki cookie, pamięć podręczną, uprawnienia i inne dane profilu. Każdy unikalny folder tworzy izolowany profil przeglądarki.

```
// Use a custom data folder for isolated profiles
sgcWebView21.UserDataFolder := 'C:\AppData\MyApp\Profile1';
```

Należy ustawić tę właściwość przed inicjalizacją (przed utworzeniem środowiska WebView2 przez komponent). Jeśli nie zostanie ustawiona, używany jest domyślny folder w katalogu tymczasowym aplikacji.

## BrowserExecutableFolder

Właściwość **BrowserExecutableFolder** wskazuje na dystrybucję środowiska uruchomieniowego WebView2 w stałej wersji. Użyj tej właściwości, aby dostarczyć określoną wersję przeglądarki wraz z aplikacją zamiast polegać na środowisku uruchomieniowym zainstalowanym w systemie.

```
// Use a fixed-version runtime bundled with the application
sgcWebView21.BrowserExecutableFolder :=
    ExtractFilePath(ParamStr(0)) + 'WebView2Runtime';
```

Ustaw tę właściwość przed inicjalizacją. Jeśli pole pozostanie puste, komponent korzysta z zainstalowanego w systemie środowiska uruchomieniowego Evergreen WebView2.

# TsgcWebView2 | Funkcje zaawansowane

TsgcWebView2 zapewnia dostęp do zaawansowanych możliwości przeglądarki, w tym drukowania, zrzutów ekranu, kontroli dźwięku, obsługi certyfikatów i innych funkcji.

## Obsługa drukowania

Użyj metody **PrintToPdf**, aby zapisać bieżącą stronę jako plik PDF, lub **ShowPrintUI**, aby wyświetlić natywne okno dialogowe drukowania.

```
// Save the current page as PDF
sgcWebView21.PrintToPdf('C:\Output\page.pdf');
```

```
// Show the native print dialog
sgcWebView21.ShowPrintUI;
```

## Przechwytywanie zrzutów ekranu

Należy użyć **CapturePreviewToFile**, aby wykonać zrzut ekranu bieżącej strony. Parametr format określa typ obrazu: 0 dla PNG, 1 dla JPEG. Zdarzenie **OnCapturePreviewCompleted** jest wywoływane po zakończeniu przechwytywania.

```
// Capture as PNG
sgcWebView21.CapturePreviewToFile('C:\Output\screenshot.png', 0);
```

```
// Capture as JPEG
sgcWebView21.CapturePreviewToFile('C:\Output\screenshot.jpg', 1);
```

```
procedure TFormMain.sgcWebView21CapturePreviewCompleted(
    Sender: TObject; aErrorCode: HRESULT);
begin
    if aErrorCode = S_OK then
        ShowMessage('Screenshot saved.')
    else
        ShowMessage('Screenshot failed: ' + IntToStr(aErrorCode));
end;
```

## Audio / Sterowanie wyciszeniem

Właściwość **IsMuted** służy do wyciszania lub przywracania dźwięku. Właściwość **IsDocumentPlayingAudio** pozwala sprawdzić, czy strona aktualnie odtwarza dźwięk.

```
// Toggle mute
sgcWebView21.IsMuted := not sgcWebView21.IsMuted;
```

```
// Check if audio is playing
if sgcWebView21.IsDocumentPlayingAudio then
    LabelStatus.Caption := 'Audio is playing';
```

## Obsługa certyfikatów

Należy użyć zdarzenia **OnClientCertificateRequested**, aby odpowiedzieć na żądanie certyfikatu klienta przez serwer. Zdarzenie **OnServerCertificateError** służy do obsługi błędów certyfikatu TLS (na przykład certyfikatów z własnym podpisem w środowisku deweloperskim).

```
procedure TFormMain.sgcWebView21ClientCertificateRequested(
    Sender: TObject; const aHost: string; aPort: Integer;
    var aHandled: Boolean);
begin
    // Handle client certificate selection
    aHandled := True;
end;

procedure TFormMain.sgcWebView21ServerCertificateError(
    Sender: TObject; const aRequestURI: string;
    aErrorStatus: Integer; var aAction: Integer);
begin
    // Accept self-signed certificates in development
    // aAction: 0 = deny, 1 = allow
    aAction := 1;
end;
```

## Mapowanie wirtualnych hostów

Należy użyć metody **SetVirtualHostNameToFolderMapping**, aby zmapować nazwę hosta na folder lokalny. Umożliwia to odwoływanie się do lokalnych plików przez zawartość sieci Web przy użyciu wirtualnego adresu URL zamiast ścieżek file://.

```
// Map "app.local" to a local folder
sgcWebView21.SetVirtualHostNameToFolderMapping(
    'app.local',
    'C:\MyApp\WebContent',
    0); // 0 = deny remote access

// Now navigate to local content using the virtual host
sgcWebView21.Navigate('https://app.local/index.html');

// Remove the mapping
sgcWebView21.ClearVirtualHostNameToFolderMapping('app.local');
```

## Zarządzanie profilem

Użyj **GetProfileName**, aby pobrać nazwę bieżącego profilu przeglądarki. Użyj **ClearBrowsingData** lub **ClearAllBrowsingData**, aby usunąć dane z pamięci podręcznej, pliki cookie i inne artefakty przeglądania.

```
// Get the profile name
ShowMessage('Profile: ' + sgcWebView21.GetProfileName);

// Clear all browsing data
sgcWebView21.ClearAllBrowsingData;

// Clear specific browsing data types
sgcWebView21.ClearBrowsingData($0001); // cache only
```

## Uwierzytelnianie podstawowe

Zdarzenie **OnBasicAuthRequested** jest wywoływane, gdy serwer żąda uwierzytelnienia HTTP Basic. Należy podać dane uwierzytelniające i ustawić właściwość **aHandled** na True.

```
procedure TFormMain.sgcWebView21BasicAuthRequested(Sender: TObject;  
  const aURI: string; var aUserName, aPassword: string;  
  var aHandled: Boolean);  
begin  
  aUserName := 'admin';  
  aPassword := 'secret';  
  aHandled := True;  
end;
```

## Menu kontekstowe

Zdarzenie **OnContextMenuRequested** jest wywoływane, gdy użytkownik kliknie prawym przyciskiem myszy w przeglądarce. Należy ustawić właściwość **aHandled** na True, aby pominąć domyślne menu i wyświetlić własne.

```
procedure TFormMain.sgcWebView21ContextMenuRequested(Sender: TObject;  
  const aMenuItems: string; aContextKind: Integer;  
  aLocation: TPoint; var aHandled: Boolean);  
begin  
  // Suppress the default context menu  
  aHandled := True;  
  
  // Show a custom popup menu at the click location  
  PopupMenu1.Popup(aLocation.X, aLocation.Y);  
end;
```

## Favicon

Właściwość **FaviconURI** zwraca identyfikator URI ikony favicon bieżącej strony. Zdarzenie **OnFaviconChanged** jest wywoływane po zmianie ikony favicon.

```
procedure TFormMain.sgcWebView21FaviconChanged(Sender: TObject;  
  const aFaviconURI: string);  
begin  
  LabelFavicon.Caption := 'Favicon: ' + aFaviconURI;  
end;
```

## Pasek stanu

Właściwość **StatusBarText** zwraca bieżący tekst paska stanu (zazwyczaj adres URL najechanego łącza). Zdarzenie **OnStatusBarTextChanged** jest wywoływane po zmianie tekstu.

```
procedure TFormMain.sgcWebView21StatusBarTextChanged(Sender: TObject;  
  const aText: string);  
begin  
  StatusBar1.SimpleText := aText;  
end;
```

## Menedżer zadań

Należy użyć metody **OpenTaskManagerWindow**, aby otworzyć menedżer zadań przeglądarki Edge, który wyświetla użycie pamięci i procesora dla każdego procesu przeglądarki.

```
sgcWebView21.OpenTaskManagerWindow;
```

## Wspólny bufor

Należy użyć metody **PostSharedBufferToScript**, aby udostępnić bufor pamięci między aplikacją Delphi a treścią internetową w celu wydajnego transferu danych.

```
// Share a buffer with web content
sgcWebView21.PostSharedBufferToScript(
    vSharedBuffer, // IUnknown shared buffer object
    0,             // 0 = read-only, 1 = read-write
    '{"type":"image","width":640,"height":480}');
```

## Bezpośredni dostęp COM

W zaawansowanych scenariuszach nieobsługiwanych przez API składnika należy użyć właściwości **WebView**, **Controller** i **Environment**, aby uzyskać bezpośredni dostęp do podstawowych interfejsów COM WebView2.

```
var
    vWebView: ICoreWebView2;
begin
    vWebView := sgcWebView21.WebView;
    if Assigned(vWebView) then
    begin
        // Call any ICoreWebView2 method directly
    end;
end;
```

# TsgcUDPClient

TsgcUDPClient implementuje klienta UDP opartego na bibliotece Indy.

## Wprowadzenie

TsgcUDPClient implementuje klienta UDP opartego na bibliotece Indy.

UDP to protokół bezpołączeniowy, który nie gwarantuje dostarczenia wysłanych wiadomości do odbiorcy, jednak w przeciwieństwie do TCP jest znacznie szybszy.

1. Umieść komponent **TsgcUDPClient** na formularzu
2. Ustaw właściwości **Host** i **Port** (domyślnie 80), aby połączyć się z dostępnym serwerem UDP.

```
oClient = new TsgcUDPClient();  
oClient->Host = "127.0.0.1";  
oClient->Port = 80;
```

3. Można połączyć się przez serwer proxy HTTP. Należy zdefiniować właściwości proxy:

**Host:** nazwa hosta serwera proxy.

**Port:** numer portu serwera proxy.

**Username:** nazwa użytkownika do uwierzytelnienia; pozostaw puste w przypadku dostępu anonimowego.

**Password:** hasło do uwierzytelnienia; pozostaw puste w przypadku dostępu anonimowego.

4. Opcjonalnie można obsłużyć zdarzenia
5. Wywołaj metodę **WriteData**, aby wysłać dowolną wiadomość do serwera UDP.

## Dokumentacja

- Właściwości — 9 właściwości dostępnych w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 3 zdarzenia do obsłużenia.



# TsgcUDPServer

TsgcUDPServer implementuje serwer UDP oparty na bibliotece Indy.

## Wprowadzenie

TsgcUDPServer implementuje serwer UDP oparty na bibliotece Indy.

UDP to protokół bezpołączeniowy, który nie gwarantuje dostarczenia wysłanych wiadomości do celu, jednak w przeciwieństwie do protokołu TCP jest znacznie szybszy.

1. Upuść komponent **TsgcUDPServer** na formularz

2. Ustaw nasłuchujący **Port**.

```
oClient = new TsgcUDPServer();  
oClient->Port = 80;
```

3. Aby **uruchomić** serwer, ustaw właściwość **Active = true**.

## Dokumentacja

- Właściwości — 11 właściwości dostępnych w komponencie.
- Metody — 7 metod publicznych.
- Zdarzenia — 6 zdarzeń do obsługi.

# STUN

STUN (Session Traversal Utilities for NAT) to protokół IETF stosowany do obsługi audio i wideo w czasie rzeczywistym w sieciach IP. STUN to protokół klient-serwer, a serwer STUN zazwyczaj działa zarówno na UDP, jak i TCP, nasłuchując na porcie 3478.

Głównym celem protokołu STUN jest umożliwienie urządzeniu działającemu za NAT odkrycia swojego publicznego adresu IP oraz określenia rodzaju NAT, za którym się znajduje.

STUN zapewnia mechanizm komunikacji między peerami za NAT. Peery wysyłają żądanie do serwera STUN, aby poznać swój publiczny adres IP i port. Żądania powiązania wysyłane przez klienta do serwera służą do ustalania powiązań IP i portów przydzielanych przez NAT. Klient STUN wysyła żądanie Binding do serwera STUN, serwer sprawdza źródłowy adres IP i port używany przez klienta, a następnie zwraca te informacje klientowi.



Serwer STUN wysyła zasadniczo dwa typy odpowiedzi: pomyślną lub z błędem. Każda odpowiedź zawiera listę atrybutów z informacjami o adresie IP powiązania, kodzie błędu, przyczynie błędu itp.

## Komponenty

- **TsgcSTUNClient:** komponent klienta implementujący protokół STUN, umożliwiający wysyłanie żądań powiązania do serwerów STUN.
- **TsgcSTUNServer:** komponent serwera implementujący protokół STUN.

# TsgcSTUNClient

Komponent TsgcSTUNClient.

## Wprowadzenie

**TsgcSTUNClient** to klient implementujący protokół STUN, który umożliwia wysyłanie żądań wiązania do serwerów STUN.

Komponent umożliwia używanie **UDP** i **TCP** jako transportu. Przy użyciu UDP jako transportu implementuje **mechanizm retransmisji** umożliwiający ponowne wysyłanie żądań, jeśli odpowiedź nie nadejdzie po krótkim czasie.

## Podstawowe użycie

Zazwyczaj serwery STUN działają na porcie UDP 3478 i nie wymagają uwierzytelnienia. Aby wysłać żądanie wiązania STUN, należy wypełnić właściwości serwera, aby klient wiedział, z czym się połączyć, oraz obsłużyć zdarzenia, w których komponent otrzyma odpowiedź od serwera.

## Konfiguracja serwera

- Host: adres IP lub nazwa DNS serwera, przykład: stun.sgcwebsockets.com
- Port: port nasłuchiwania serwera, przykład: 3478

Należy wywołać metodę **SendRequest**, aby wysłać żądanie do serwera STUN.

## Obsługa zdarzeń

- Jeśli serwer zwróci pomyślną odpowiedź, wywoływane jest zdarzenie **OnSTUNResponseSuccess** i można uzyskać dostęp do informacji o powiązaniu, odczytując obiekt **aBinding**.
- Jeśli serwer zwróci błąd, zostanie wywołane zdarzenie **OnSTUNResponseError**, a dostęp do kodu błędu i przyczyny można uzyskać, odczytując obiekt **aError**.

```
TsgcSTUNClient oSTUN = new TsgcSTUNClient(this);
oSTUN->Host = "stun.sgcwebsockets.com";
oSTUN->Port = 3478;
oSTUN->SendRequest();

private void OnSTUNResponseSuccess(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcSTUN_Message *aMessage, const TsgcSTUN_ResponseBinding *aBinding)
{
    DoLog("Remote IP: " + aBinding->RemoteIP + ". Remote Port: " + IntToStr(aBinding->RemotePort));
}
```

```
private void OnSTUNResponseError(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcSTUN_Message *aMessage, const TsgcSTUN_ResponseError *aError)
{
    DoLog("Error: " + IntToStr(aError->Code) + " " + aError->Reason);
}
```

## Dokumentacja

- Właściwości — 9 właściwości dostępnych w komponencie.
- Metody — 3 metody publiczne.
- Zdarzenia — 4 zdarzenia do obsługi.

## Przewodniki

- **Powiązania**
  - [Retransmisje UDP](#)
  - [Poświadczenia długoterminowe](#)
- [Atrybuty](#)

## Klient STUN | Retransmisje UDP

Gdy protokół STUN działa przez UDP, istnieje możliwość, że **wiadomość STUN** zostanie **odrzucona** przez sieć. Niezawodność transakcji żądanie/odpowiedź STUN jest zapewniana przez ponowne wysyłanie wiadomości żądania przez samą aplikację kliencką.

Klient powinien retransmitować wiadomość żądania STUN, zaczynając od interwału RTO („Retransmission TimeOut”), podwajając go po każdej retransmisji. RTO jest szacunkiem czasu podróży w obie strony.

Domyślnie klient STUN sgcWebSockets jest skonfigurowany z RTO wynoszącym 500 ms i maksymalną liczbą ponownych prób równą 7.

Na przykład, zakładając RTO wynoszące 500 ms, żądania byłyby wysyłane w chwilach 0 ms, 500 ms, 1500 ms, 3500 ms, 7500 ms, 15500 ms i 31500 ms. Jeśli klient nie otrzyma odpowiedzi po 39500 ms, uzna transakcję za przeterminowaną.

```
TsgcSTUNClient oSTUN = new TsgcSTUNClient(this);
oSTUN->Host = "stun.sgcwebsockets.com";
oSTUN->Port = 3478;
oSTUN->RetransmissionOptions->Enabled = true;
oSTUN->RetransmissionOptions->RTO = 500;
oSTUN->RetransmissionOptions->MaxRetries = 7;
oSTUN->SendRequest();
```

## Klient STUN | Długoterminowe poświadczenia

Mechanizm długoterminowych poświadczeń opiera się na długoterminowych poświadczeniach w postaci nazwy użytkownika i hasła, które są współdzielone między klientem a serwerem. Poświadczenia są uznawane za długoterminowe, ponieważ zakłada się, że są przypisane do użytkownika i pozostają aktywne do momentu, gdy użytkownik nie jest już subskrybentem systemu lub do momentu ich zmiany.

Długoterminowe poświadczenia w kliencie sgcWebSockets STUN można skonfigurować przy użyciu następującego kodu.

```
TsgcSTUNClient oSTUN = new TsgcSTUNClient(this);
oSTUN->Host = "stun.sgcwebsockets.com";
oSTUN->Port = 3478;
oSTUN->STUNOptions->Authentication->Credentials = stauLongTermCredential;
oSTUN->STUNOptions->Authentication->Username = "user_name";
oSTUN->STUNOptions->Authentication->Password = "secret";
oSTUN->SendRequest();
```

Jeśli serwer wymaga długoterminowych poświadczeń, a poświadczenia przesłane przez klienta są nieprawidłowe, klient otrzyma odpowiedź z błędem 401 Unauthorized w zdarzeniu **OnSTUNResponseError**.



# TsgcSTUNServer

Serwer STUN można skonfigurować z uwierzytelnianiem lub bez niego, z możliwością weryfikacji atrybutu Fingerprint, wysyłania alternatywnego serwera i nie tylko.

## Wprowadzenie

**TsgcSTUNServer** to serwer implementujący protokół STUN i umożliwiający przetwarzanie żądań wiązania od klientów STUN.

Serwer STUN można skonfigurować z uwierzytelnianiem lub bez niego, z możliwością weryfikacji atrybutu Fingerprint, wysyłania alternatywnego serwera i nie tylko.

## Podstawowe użycie

Zazwyczaj serwery STUN działają na porcie UDP 3478 i nie wymagają uwierzytelniania, dlatego w celu skonfigurowania serwera STUN należy ustawić port nasłuchiwanie (domyślnie 3478) i uruchomić serwer.

## Konfiguracja serwera

- Port: port nasłuchiwanie serwera, przykład: 3478

Ustaw właściwość **Active = True**, aby uruchomić serwer STUN.

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();  
oSTUN->Port = 3478;  
oSTUN->Active = true;
```

## Dokumentacja

- Właściwości — 8 właściwości dostępnych w komponencie.
- Metody — 2 metody publiczne.
- Zdarzenia — 4 zdarzenia do obsłużenia.

## Przewodniki

- Konfiguracje
  - [Poświadczenia długoterminowe](#)
  - [Serwer alternatywny](#)



# Serwer STUN | Długoterminowe dane uwierzytelniające

Zazwyczaj serwery STUN są konfigurowane bez uwierzytelniania, dzięki czemu dowolny klient STUN może wysłać żądanie powiązania i oczekiwać odpowiedzi serwera bez uwierzytelniania.

Serwer STUN sgcWebSockets obsługuje długoterminowe poświadczenia, dzięki czemu można skonfigurować TsgcSTUNServer tak, aby zezwalał wyłącznie na żądania powiązania zawierające informacje o długoterminowych poświadczeniach.

Aby skonfigurować, należy uzyskać dostęp do właściwości STUNOptions.Authorization i ją włączyć.

Następnie należy uzyskać dostęp do właściwości LongTermCredentials i włączyć ją. Domyślnie ten typ autoryzacji jest już skonfigurowany z ciągiem Realm oraz domyślną wartością StaleNonce wynoszącą 10 minut (= 600 sekund).

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();
oSTUN->STUNOptions->Authentication->Enabled = true;
oSTUN->STUNOptions->Authentication->LongTermCredentials->Enabled = true;
oSTUN->STUNOptions->Authentication->LongTermCredentials->Realm = "sgcWebSockets";
oSTUN->STUNOptions->Authentication->LongTermCredentials->StaleNonce = 600;
oSTUN->Port = 3478;
oSTUN->Active = true;

private void OnSTUNRequestAuthorization(TObject *Sender, const TsgcSTUN_Message *aRequest,
    const string aUsername, const string aRealm, ref string Password)
{
    if (aUsername == "my-user")
    {
        Password = "my-password";
    }
}
```

## STUN Server | Alternate Server

Serwer alternatywny reprezentuje alternatywny adres transportowy identyfikujący inny serwer STUN, z którym klient STUN powinien nawiązać połączenie.

Serwer STUN można skonfigurować tak, aby w odpowiedzi na żądanie powiązania wysyłał alternatywny serwer. Aby skonfigurować to zachowanie, należy uzyskać dostęp do właściwości `STUNOptions.BindingAttributes.AlternateServer` i ustawić tam wymagane wartości.

```
TsgcSTUNServer oSTUN = new TsgcSTUNServer();  
oSTUN->Port = 3478;  
oSTUN->STUNOptions->BindingAttributes->AlternateServer->Enabled = true;  
oSTUN->STUNOptions->BindingAttributes->AlternateServer->IPAddress = "80.54.54.1";  
oSTUN->STUNOptions->BindingAttributes->AlternateServer->Port = 3478;  
oSTUN->Active = true;
```

Gdy klient otrzyma atrybut odpowiedzi Alternate Server, spróbuje wysłać żądanie powiązania do nowego serwera.

# TURN

Protokół TURN (Traversal Using Relays around NAT) umożliwia serwerowi przekazywanie pakietów danych między urządzeniami.

Jeśli publiczny adres IP zarówno inicjatora, jak i odbiorcy połączenia nie zostanie wykryty, TURN zapewnia technikę rezerwową do przekazywania połączenia między punktami końcowymi.

Nawiązanie sesji WebRTC to skoordynowane działanie realizowane przy wsparciu wielu serwerów WebRTC. Serwery przechodzenia przez NAT w WebRTC są odpowiedzialne za zapewnienie prawidłowego nawiązania połączenia z mediami. Tymi serwerami są STUN i TURN.

## Jak nawiązywane są sesje WebRTC

### Bezpośrednio

Jeśli oba urządzenia znajdują się w sieci lokalnej, nie są wymagane żadne specjalne działania, aby je ze sobą połączyć. Jeśli jedno urządzenie zna lokalny adres IP drugiego, mogą komunikować się bezpośrednio.

### Bezpośrednio z publicznym adresem IP

Bezpośrednie połączenie WebRTC przy użyciu publicznego adresu IP uzyskanego przez protokół [STUN](#).

### Trasowanie przez serwer TURN

Gdy elementy sieci równorzędnej znajdują się za NAT i zaporami sieciowymi, bezpośrednie połączenie nie jest możliwe, dlatego do trasowania danych między węzłami wymagany jest serwer TURN.

## Komponenty

- **TsgcTURNClient:** komponent klienta implementujący protokół TURN, umożliwiający alokowanie zasobów, tworzenie uprawnień, wysyłanie wskazań itp. do serwera TURN.

- **TsgcTURNServer:** komponent serwera implementujący protokół TURN.

# TsgcTURNClient

Komponent TsgcTURNClient.

## Wprowadzenie

**TsgcTURNClient** to klient implementujący protokół TURN i umożliwiający wysyłanie żądań alokacji do serwerów TURN. Klient dziedziczy po kliencie STUN, więc wszystkie metody obsługiwane przez [klienta STUN](#) są już obsługiwane przez klienta TURN.

## Podstawowe użycie

Serwery TURN zazwyczaj działają na porcie UDP 3478 i nie wymagają uwierzytelniania. Aby wysłać żądanie TURN, należy wypełnić właściwości serwera, informując klienta o miejscu połączenia, oraz obsłużyć zdarzenia, za pośrednictwem których komponent odbierze odpowiedź serwera.

## Konfiguracja serwera

- Host: adres IP lub nazwa DNS serwera, przykład: turn.sgcwebsockets.com
- Port: port nasłuchiwania serwera, przykład: 3478

Należy wywołać metodę **Allocate**, aby wysłać żądanie przydzielenia adresu IP i portu do serwera TURN.

## Obsługa zdarzeń

- Jeśli serwer zwróci pomyślną odpowiedź, zostanie wywołane zdarzenie **OnTURNAllocateSuccess** i można uzyskać dostęp do informacji o alokacji, odczytując obiekt **aAllocation**.
- Jeśli serwer zwróci błąd, zostanie wywołane zdarzenie **OnSTUNResponseError**, a dostęp do kodu błędu i przyczyny można uzyskać, odczytując obiekt **aError**.

```
TsgcTURNClient oTURN = new TsgcTURNClient(this);
oTURN->Host = "turn.sgcwebsockets.com";
oTURN->Port = 3478;
oTURN->Allocate();

private void OnTURNAllocate(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcSTUN_Message *aMessage, const TsgcTURN_ResponseAllocation *aAllocation)
{
    DoLog("Relayed IP: " + aAllocation->RelayedIP + ". Relayed Port: " + IntToStr(aAllocation->RelayedPort));
}

private void OnSTUNResponseError(TObject *Sender, const TsgcSTUN_Message *aMessage,
    const TsgcSTUN_ResponseError *aError)
{
}
```

```
DoLog("Error: " + IntToStr(aError->Code) + " " + aError->Reason);  
}
```

## Dokumentacja

- Właściwości — 10 właściwości dostępnych w komponencie.
- Metody — 9 metod publicznych.
- Zdarzenia — 11 zdarzeń do obsługi.

## Przewodniki

- **Alokacja**
  - [Przydziel adres IP](#)
  - [Utwórz uprawnienia](#)
- **Wskazania**
  - [Wyślij wskazanie](#)
- **Kanały**
  - [Kanały klienta TURN](#)

# Klient TURN | Przydzielanie adresu IP

Protokół TURN umożliwia korzystanie z przekazanego adresu IP do wymiany danych między węzłami znajdującymi się za NAT-ami.

Aby utworzyć nowy relayowany adres IP na serwerze TURN, klient musi najpierw wywołać metodę **Allocate**. Metoda ta wysyła żądanie do serwera TURN o utworzenie nowego relayowanego adresu IP. Jeżeli serwer TURN może go utworzyć, klient otrzyma pomyślną odpowiedź i będzie mógł komunikować się z innymi węzłami przez czas zdefiniowany w parametrze lifetime alokacji.

```
TsgcTURNClient oTURN = new TsgcTURNClient(this);
oTURN->Host = "turn.sgcwebsockets.com";
oTURN->Port = 3478;
oTURN->Allocate();

private void OnTURNAllocate(TObject *Sender, const TsgcSocketConnection *aSocket,
const TsgcSTUN_Message *aMessage, const TsgcTURN_ResponseAllocation *aAllocation)
{
    DoLog("Relayed IP: " + aAllocation->RelayedIP + ". Relayed Port: " +
        IntToStr(aAllocation->RelayedPort));
}

private void OnSTUNResponseError(TObject *Sender, const TsgcSTUN_Message *aMessage,
const TsgcSTUN_ResponseError *aError)
{
    DoLog("Error: " + IntToStr(aError->Code) + " " + aError->Reason);
}
```

Czas życia można zaktualizować, aby zapobiec wygaśnięciu, używając metody **Refresh**. Lifetime to liczba sekund do wygaśnięcia. Jeśli wartość wynosi zero, alokacja zostanie usunięta.

```
oTURN->Refresh(600);
```

## Klient TURN | Tworzenie uprawnień

Gdy w serwerze TURN tworzona jest nowa Alokacja, nie może ona przetwarzać żadnych przychodzących pakietów od innych węzłów, jeśli nie posiada uprawnień. Aby umożliwić innym węzłom komunikację z użyciem przekazanego adresu IP, klient TURN musi najpierw utworzyć uprawnienia dla adresów IP, które mają prawo do wymiany danych.

Aby utworzyć nowe uprawnienie, należy wywołać metodę **CreatePermission** i przekazać jako parametr adres IP węzła zdalnego. Jeśli adres IP węzła już istnieje na serwerze TURN, zostanie odświeżony; jeśli nie, zostanie utworzony. Uprawnienia wygasają po 5 minutach, chyba że zostaną odświeżone.

Klient TURN umożliwia wywołanie metody **CreatePermission** tylko wtedy, gdy istnieje aktywna alokacja.

Jeśli uprawnienie zostanie utworzone pomyślnie, wywoływane jest zdarzenie **OnTURNCreatePermission**.

```
oTURN->CreatePermission("80.147.23.157");
```

```
void OnTURNCreatePermission(TObject *Sender, const TsgcSocketConnection *aSocket,  
    const TsgcSTUN_Message *aMessage, const TsgcTURN_ResponseCreatePermission *aCreatePermission)  
{  
    DoLog("#Create Permission: " + aCreatePermission->IPAddresses->Text);  
}
```



## Klient TURN | Wysyłanie wskazania

Protokół TURN obsługuje 2 mechanizmy wysyłania i odbierania danych od węzłów; jednym z nich jest mechanizm Send i Data.

Klient TURN może użyć metody **SendIndication** do wysyłania danych do serwera w celu przekazania ich do uczestnika. Klient TURN musi upewnić się, że istnieje uprawnienie dla adresu IP uczestnika, do którego wysyłana jest wskazówka Send.

Odpowiedzi na metodę SendIndication są odbierane w zdarzeniu **OnTURNDataIndication**.

```
oTURN->SendIndication("80.147.23.157", 5000, "random data");

void OnTURNDataIndication(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcSTUN_Message *aMessage, const TsgcTURN_ResponseDataIndication *aDataIndication)
{
    DoLog("#Data Indication: [" + aDataIndication->PeerIP + ":" + IntToStr(aDataIndication->PeerPort) + "] " +
        sgcGetStringFromBytes(aDataIndication->Data));
}
```

## Klient TURN | Kanały

Kanały zapewniają klientowi i serwerowi TURN możliwość wysyłania danych aplikacji przy użyciu wiadomości ChannelData, które mają mniejszy narzut niż [wskazania Send i Data](#).

Przed wymianą danych za pomocą wiadomości ChannelData między uczestnikami klient TURN musi utworzyć nowy kanał — w tym celu należy wywołać metodę **ChannelBind**, przekazując adres IP i port uczestnika jako parametry.

Jeśli serwer TURN może powiązać nowy kanał, klient TURN otrzyma zdarzenie **OnTURNChannelBind** z pomyślną odpowiedzią.

```
oTURN->ChannelBind("80.147.23.157", 5000);

void OnTURNChannelBind(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcSTUN_Message *aMessage, const TsgcTURN_ResponseChannelBind *aChannelBind)
{
    DoLog("#Channel Bind: " + IntToStr(aChannelBind->Channel));
}
```

Wiązanie kanału jest ważne przez 10 minut, chyba że zostanie odświeżone. Aby odświeżyć kanał, należy ponownie wywołać metodę **ChannelBind**.

Gdy klient TURN odbiera nową wiadomość ChannelMessage, wywoływane jest zdarzenie **OnTURNChannelData**.

```
void OnTURNChannelData(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcTURNChannelData *aChannelData)
{
    DoLog("#Channel Data: [" + IntToStr(aChannelData->ChannelID) + "] " +
        sgcGetStringFromBytes(aChannelData->Data));
}
```

# TsgcTURNServer

Serwer TURN obsługuje uwierzytelnianie długoterminowe, alokację, uprawnienia, dane kanałów i wiele innych funkcji.

## Wprowadzenie

**TsgcTURNServer** to serwer implementujący protokół TURN umożliwiający przetwarzanie żądań od klientów TURN. Komponent dziedziczy po [TsgcSTUNServer](#), więc wszystkie metody i właściwości są dostępne również w TsgcTURNServer.

Serwer TURN obsługuje uwierzytelnianie długoterminowe, alokację, uprawnienia, dane kanałów i wiele innych funkcji.

## Podstawowe użycie

Zazwyczaj serwery TURN działają na porcie UDP 3478 i wymagają poświadczeń długoterminowych; aby skonfigurować serwer TURN, należy ustawić port nasłuchiwania (domyślnie 3478) i uruchomić serwer.

## Konfiguracja serwera

- Port: port nasłuchiwania serwera, przykład: 3478
- Zdefiniuj właściwości długoterminowych poświadczeń w `TURNOptions.Authentication.LongTermCredentials`
- Należy obsłużyć zdarzenie `OnSTUNRequestAuthorization`, aby ustawić hasło, gdy klient TURN wysła żądanie do serwera TURN.

Ustaw właściwość **Active = True**, aby uruchomić serwer TURN.

```
TsgcTURNServer oTURN = new TsgcTURNServer();
oTURN->Port = 3478;
oTURN->TURNOptions->Authentication->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Realm = "esegece.com";
oTURN->Active = true;
void OnSTUNRequestAuthorization(TObject *Sender, const TsgcSTUN_Message *aRequest,
    const string aUsername, const string aRealm, ref string Password)
{
    if ((aUsername == "user") && (aRealm == "esegece.com"))
    {
        Password = "password";
    }
}
```

# Dokumentacja

- Właściwości — 9 właściwości dostępnych w komponencie.
- Methods — 2 metody publiczne.
- Zdarzenia — 11 zdarzeń do obsługi.

# Przewodniki

- **Konfiguracje**
  - [Poświadczenia długoterminowe](#)
- **Alokacje**
  - [Alokacje](#)

# Serwer TURN | Długoterminowe dane uwierzytelniające

Serwery TURN są zazwyczaj konfigurowane z uwierzytelnianiem dla żądań TURN i bez uwierzytelniania dla żądań STUN.

Serwer sgcWebSockets TURN obsługuje długoterminowe poświadczenia, więc można skonfigurować TsgcTURNServer tak, aby zezwalał jedynie na żądania z informacjami o długoterminowych poświadczeniach.

Aby skonfigurować, należy uzyskać dostęp do właściwości TURNOptions.Authorization i ją włączyć.

Następnie należy uzyskać dostęp do właściwości LongTermCredentials i włączyć ją. Domyślnie ten typ autoryzacji jest już skonfigurowany z ciągiem Realm oraz domyślną wartością StaleNonce wynoszącą 10 minut (= 600 sekund).

```
TsgcTURNServer oTURN = new TsgcTURNServer();
oTURN->STUNOptions->Authentication->Enabled = false;
oTURN->TURNOptions->Authentication->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Enabled = true;
oTURN->TURNOptions->Authentication->LongTermCredentials->Realm = "sgcWebSockets";
oTURN->TURNOptions->Authentication->LongTermCredentials->StaleNonce = 600;
oTURN->Port = 3478;
oTURN->Active = true;

private void OnSTUNRequestAuthorization(TObject *Sender, const TsgcSTUN_Message *aRequest,
    const string aUsername, const string aRealm, ref string Password)
{
    if ((aUsername == "my-user") && (aRealm == "sgcWebSockets"))
    {
        Password = "my-password";
    }
}
```

# TURN Server | Alokacje

Wszystkie operacje TURN opierają się na alokacjach, a wszystkie wiadomości TURN są powiązane z alokacją. Alokacja składa się z:

- Adres transportu przekaźnikowego
- 5-krotka: adres IP klienta, port IP klienta, adres IP serwera, port serwera i protokół transportowy.
- Informacje uwierzytelniające.
- Czas do wygaśnięcia dla każdego przekazywanego adresu transportowego.
- Lista uprawnień dla każdego przekazywanego adresu transportowego.
- Lista powiązań kanałów dla każdego przekazywanego adresu transportowego.

Gdy klient TURN wysyła żądanie Allocate, ta wiadomość TURN jest przetwarzana przez serwer, który próbuje utworzyć nowy przekazany adres transportowy. Domyślnie, jeśli dostępny jest jakikolwiek port UDP, zostanie utworzony nowy adres przekazywania, jednak zdarzenie **OnTURNBeforeAllocate** pozwala odrzucić nowe żądanie alokacji.

```
void OnTURNBeforeAllocate(TObject *Sender, const TsgcSocketConnection *aSocket,
    const string aIP, Word aPort, ref bool Reject)
{
    if (your own rules) == false
    {
        Reject = false;
    }
}
```

Jeśli proces jest kontynuowany, serwer tworzy nową alokację i wywoływane jest zdarzenie **OnTURNCreateAllocation**. Zdarzenie to dostarcza informacji o alokacji za pośrednictwem klasy TsgcTURNAllocationItem.

```
void OnTURNCreateAllocation(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcTURNAllocationItem *Allocation)
{
    DoLog("New Allocation: " + Allocation->RelayIP + ":" + IntToStr(Allocation->RelayPort));
}
```

Gdy alokacja wygaśnie lub zostanie usunięta po odebraniu żądania Refresh od klienta z czasem życia równym zero, wywoływane jest zdarzenie **OnTURNDeleteAllocation**.

```
void OnTURNDeleteAllocation(TObject *Sender, const TsgcSocketConnection *aSocket,
    const TsgcTURNAllocationItem *Allocation)
{
    DoLog("Allocation Deleted: " + Allocation->RelayIP + ":" + IntToStr(Allocation->RelayPort));
}
```



# ICE

Protokół Interactive Connectivity Establishment (ICE) służy do przechodzenia przez NAT. ICE wykorzystuje kombinację metod, w tym Session Traversal Utility for NAT (STUN) oraz Traversal Using Relay NAT (TURN). Obecność translatora adresów sieciowych (NAT) stwarza problemy dla implementacji Voice over IP (VoIP) i WebRTC.

## Komponenty

- **TsGcICEClient:** komponent klienta implementujący protokół ICE, umożliwiający pozyskiwanie, wymianę i weryfikację kandydatów.



# TsgcICEClient

ICEOptions.

## Wprowadzenie

**TsgcICEClient** to klient implementujący protokół ICE, który umożliwia wysyłanie żądań alokacji do serwerów TURN. Klient wymaga komponentu [TsgcTURNClient](#) oraz [TsgcWebSocketClient](#).

## Konfiguracja

Klient ICE posiada następujące właściwości

- **ICEOptions.CheckList.MaxCandidates:** maksymalna liczba kandydatów, które może obsłużyć klient; domyślnie 100.
- **ICEOptions.GatherCandidates.STUN:** domyślnie True; pozwala na uzyskanie kandydatów za pomocą protokołu STUN (adres refleksywny).
- **ICEOptions.GatherCandidates.TURN:** domyślnie true; powoduje pobranie kandydatów za pomocą protokołu TURN (adres przekazywany).

Klient ICE wymaga **klienta TURN** do zbierania kandydatów, dlatego przed wywołaniem metody **GatherCandidates** należy połączyć TsgcTURNClient z właściwością TURN komponentu TsgcICEClient. Więcej informacji na temat konfiguracji [klienta TURN](#) można znaleźć w dokumentacji.

```
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();
```

## Dokumentacja

- Właściwości — 3 właściwości dostępne w komponencie.
- Metody — 3 metody publiczne.
- Zdarzenia — 6 zdarzeń do obsługi.

# Przewodniki

- **Kandydaci**
  - [Zbieranie kandydatów](#)
  - [Parowanie kandydatów](#)

# ICE | Zbieranie kandydatów

ICE rozpoczyna zbieranie kandydatów — zazwyczaj uzyskuje lokalne adresy IP, adresy refleksyjne przy użyciu protokołu STUN i adresy przekazywane przy użyciu protokołu TURN.

Aby rozpocząć zbieranie kandydatów, należy wywołać metodę **GatherCandidates**. Spowoduje to uruchomienie wewnętrznego licznika czasu, który najpierw pobiera lokalne adresy IP, następnie łączy się z serwerem STUN w celu uzyskania refleksyjnego adresu IP, a na końcu łączy się z serwerem TURN, aby uzyskać przekierowany adres IP.

Za każdym razem, gdy zostanie uzyskany nowy kandydat, zdarzenie **OnICECandidate** zostanie wywołane asynchronicznie. W przypadku błędu podczas zbierania kandydatów zostanie wyzwolone zdarzenie **OnICECandidateError**.

```
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);
}
```

## ICE | Pary kandydatów

Po uzyskaniu kandydatów (lokalnych i zdalnych) oraz ustawieniu opisów SDP, klient wywołujący ICE może rozpocząć przetwarzanie wszystkich par kandydatów w celu znalezienia tych, które mogą wymieniać dane. Aby uruchomić ten proces, należy wywołać metodę **ProcessCandidates**.

Metoda **ProcessCandidates** ocenia wszystkie pary kandydatów, wysyłając pakiet STUN binding; jeśli pakiet STUN binding jest odebrany jako odpowiedź od drugiego węzła, oznacza to, że połączenie między tymi dwoma węzłami jest możliwe i para jest nominowana.

Gdy parowanie zakończy się **pomyślnie**, zdarzenie **OnICECandidatePairNominated** jest wywoływane asynchronicznie. Jeśli parowanie napotka **błąd** lub **nie powiedzie się po upływie limitu czasu**, wywoływane jest zdarzenie **OnICECandidatePairFailed**.

# TsgcRTCPeerConnection

TsgcRTCPeerConnection to komponent kliencki umożliwiający połączenie węzłów peer za pomocą P2P przez UDP.

## Wprowadzenie

**[\*Komponent jest obecnie w fazie tworzenia]**

TsgcRTCPeerConnection jest komponentem klienta umożliwiającym łączenie partnerów za pomocą P2P przez UDP. Przepływ można podzielić na 4 kroki:

- Sygnalizacja
- Nawiązywanie połączenia
- Zabezpieczanie
- Komunikowanie

W celu realizacji tych kroków klient korzysta z następujących protokołów:

- **WebSocket:** protokół ten jest używany do sygnalizacji; klienci wymieniają Session Description Protocol oraz lokalne, publiczne i przekazywane adresy IP.
- **UDP:** jest to protokół transportowy; klient używa UDP do wysyłania i odbierania wiadomości między peerami.
- **DTLS:** podobnie jak TLS, jest specyfikacją szyfrowania zabezpieczającą wiadomości między uczestnikami, uniemożliwiając osobom trzecim odczyt lub zapis wiadomości.
- **STUN:** protokół do uzyskiwania publicznego adresu IP.
- **TURN:** protokół do przekazywania adresów IP, gdy peery znajdują się za NAT-ami.
- **ICE:** protokół służący do znajdowania adresów IP i portów dostępnych między węzłami.

## Sygnalizacja

Gdy klient uruchamia się, nie wie, z kim będzie się komunikował ani o czym. Sygnalizacja używa protokołu SDP (Session Description Protocol), który zawiera szczegóły takie jak:

- Adresy IP i porty, pod którymi osiągalny jest peer.
- Certyfikat odcisku palca używany do zabezpieczenia komunikacji.
- Nazwa użytkownika i hasło.
- ...

Sygnalizacja wykorzystuje protokół WebSocket do wymiany danych, działa za pośrednictwem subprotokołu i jest zaimplementowana w komponencie TsgcWSPServer\_RTCPeerConnection po stronie serwera.

TsgcRTCPeerConnection tworzy wewnętrznie klienta WebSocket z dołączonym TsgcWSPClient\_RTCPeerConnection.

Aby uzyskać adresy IP i porty, klient korzysta z protokołów STUN/TURN. Wymagany jest zatem serwer STUN/TURN.

#### **Linki:**

- [RTCPeerConnection WebSocket Server](#)
- [RTCPeerConnection WebSocket Client](#)
- [RTCPeerConnection STUN TURN](#)
- [Sygnalizacja RTCPeerConnection](#)

## **Nawiązywanie połączenia**

Gdy obaj partnerzy znają kandydatów i SDP, klient używa innego standardowego protokołu zwanego ICE.

ICE (Interactive Connection Establishment) umożliwia nawiązanie połączenia między 2 peerami. Peerzy mogą znajdować się w tej samej sieci lub za NAT... ICE jest rozwiązaniem pozwalającym nawiązać bezpośrednie połączenie bez centralnego serwera. Jeśli połączenie P2P nie jest możliwe, ICE używa TURN do przekazywania danych przez serwer TURN.

Gdy ICE znajdzie prawidłowego kandydata umożliwiającego połączenie między dwoma węzłami, kolejnym krokiem jest szyfrowanie komunikacji.

#### **Linki:**

- [RTCPeerConnection ICE](#)

## **Zabezpieczanie**

Po nawiązaniu połączenia przez uczestników komunikacja musi być zabezpieczona. Realizuje to DTLS, czyli kryptograficzny protokół służący do zabezpieczania komunikacji przez UDP.

Po pomyślnym przetworzeniu uzgadniania DTLS używany jest inny protokół — SRTP (Secure Real-Time Transport Protocol); obecnie SRTP nie jest zaimplementowany.

## Linki:

- [RTCPeerConnection DTLS](#)

## Komunikowanie

Po nawiązaniu przez obydwu partnery bezpiecznego protokołu komunikacja odbywa się przy użyciu 2 protokołów:

- **RTP:** Real Time Transport Protocol: używany do wymiany mediów zaszyfrowanych przy użyciu SRTP.
- **SCTP:** Stream Control Transmission Protocol, używany do wysyłania i odbierania wiadomości DataChannel zaszyfrowanych przy użyciu dTLS.

Aktualnie protokoły te nie są zaimplementowane, lecz możliwe jest wysyłanie i odbieranie danych za pomocą DTLS przez UDP.

## Linki:

- [RTCPeerConnection Data](#)

## Dokumentacja

- Właściwości — 2 właściwości dostępne w składniku.
- Metody — 3 metody publiczne.
- Zdarzenia — 14 zdarzeń do obsługi.

## RTCPeerConnection | Serwer WebSocket

Klient TsgcRTCPeerConnection wymaga serwera WebSocket do sygnalizacji. Klient korzysta z protokołu WebSocket do wymiany SDP między uczestnikami oraz kandydatów (adresów IP i portów), co umożliwia komunikację między uczestnikami.

Aby skonfigurować **serwer WebSocket**, można użyć dowolnego z WebSocket serwerów dostępnych w bibliotece sgcWebSockets i dołączyć **TsgcWSPServer\_RTCPeerConnection**, który jest podprotokołem używanym przez RTCPeerConnection.

```
TsgcWebSocketServer *oServer = new TsgcWebSocketServer();
TsgcWSPServer_RTCPeerConnection *oProtocol = new TsgcWSPServer_RTCPeerConnection();
oProtocol->Server = oServer;
oServer->Port = 8080;
oServer->Active = true;
```

Za każdym razem, gdy nowy klient WebSocket łączy się z serwerem WebSocket, serwer sprawdza, czy inny partner nasłuchuje na tym samym kanale, i odpowiednio przekazuje dane.



## RTCTConnection | Klient WebSocket

RTCTPeerConnection tworzy wewnętrznie klienta WebSocket z niestandardowym podprotokołem w celu komunikacji z serwerem WebSocket. W właściwości RTCTOptions.WebSocket można znaleźć wartości definiujące połączenie WebSocket.

- **Host:** nazwa DNS lub adres IP serwera, przykład: 127.0.0.1 lub www.esegece.com.
- **Port:** port nasłuchu serwera websocket.
- **TLS:** należy włączyć, jeśli serwer używa bezpiecznego połączenia.
- **Channel:** nazwa kanału używana do wymiany danych między węzłami (oba węzły muszą mieć tę samą nazwę kanału, a maksymalna liczba węzłów wynosi 2).

## RTCPeerConnection | STUN TURN

TsgcRTCPeerConnection używa protokołu STUN/TURN do uzyskania publicznego adresu IP i przekierowanego adresu IP (jeśli jest wymagany). Konieczny jest zatem serwer STUN/TURN. Więcej informacji na temat serwera STUN/TURN można znaleźć pod następującym linkiem: [Serwer TURN](#).

Po uruchomieniu serwera STUN/TURN można skonfigurować właściwości serwera TURN we właściwości RTCOptions.ICE komponentu TsgcRTCPeerConnection.

- **Host:** adres IP lub DNS serwera TURN. Przykład: 127.0.0.1 lub www.esegece.com.
- **Port:** zazwyczaj domyślny port 3478.
- **Username:** nazwa użytkownika, jeśli serwer TURN używa uwierzytelniania długoterminowego (domyślne).
- **Password:** hasło, jeśli serwer TURN używa uwierzytelniania długoterminowego (domyślne).

## RTCPeerConnection | Sygnalizacja

Gdy komponent `TsgcRTCPeerConnection` skonfiguruje właściwość `RTCOptions`, a serwery (WebSocket oraz STUN/TURN) będą uruchomione, klient może rozpocząć proces zbierania kandydatów.

Klient najpierw łączy się z serwerem WebSocket; jeśli połączenie powiedzie się, wysyła lokalne SDP. Następnie próbuje pobrać lokalne i publiczne adresy IP. W celu uzyskania publicznych adresów IP wysyła żądanie powiązania do serwera STUN, aby uzyskać publiczny adres IP oraz przekazany adres IP serwera TURN. Za każdym razem, gdy zostanie pobrany nowy kandydat, informacja ta jest przekazywana do serwera WebSocket, który przekazuje ją do drugiego węzła.

Gdy `RTCPeerConnection` posiada lokalny SDP, zdalny SDP oraz kandydatów, rozpoczyna proces sprawdzania każdej pary kandydatów pod kątem możliwości nawiązania połączenia. Gdy para kandydatów nawiązuje pomyślne połączenie, staje się ona prawidłową parą kandydatów, a proces zabezpieczania kontynuuje przebieg.

# RTCPeerConnection | ICE

ICE (Interactive Connectivity Establishment) to protokół używany do łączenia dwóch węzłów; określa wszystkie możliwe trasy między nimi, a następnie zapewnia ich połączenie. Trasy te są znane również jako pary kandydatów, czyli powiązanie lokalnych i zdalnych adresów transportowych. Adresy te mogą być lokalnym adresem IP, publicznym adresem IP lub adresem transportowym przekazywanym. Każdy węzeł gromadzi wszystkie adresy, których chce użyć, wymienia je z drugim węzłem, a następnie próbuje nawiązać połączenie.

## Zbieranie adresów

Następujące zdarzenia mogą być wywoływane podczas zbierania adresów.

### **OnRTCLocalCandidate**

Zdarzenie jest wywoływane po znalezieniu nowego lokalnego kandydata.

### **OnRTCRemoteCandidate**

Zdarzenie jest wywoływane, gdy serwer WebSocket wysyła do tego peera zdalny kandydat.

### **OnRTCLocalDescription**

Zdarzenie jest wywoływane, gdy TsgcRTCPeerConnection wymaga lokalnego SDP

### **OnRTCRemoteDescription**

Zdarzenie jest wywoływane, gdy serwer WebSocket wysyła zdalne SDP do tego peera.

## Testowanie łączności

Gdy węzeł wysyła żądania powiązania do drugiego węzła w celu sprawdzenia możliwości połączenia, mogą być wywoływane następujące zdarzenia.

### **OnRTCCandidatePairNominated**

Zdarzenie jest wywoływane, gdy oba węzły mogą połączyć się za pomocą tej pary kandydatów.

### **OnRTCCandidatePairFailed**

Gdy węzły nie mogą nawiązać połączenia przy użyciu tej pary kandydatów, wywoływane jest to zdarzenie.

### **OnRTCCConnect**

To zdarzenie jest wywoływane, gdy istnieje prawidłowa para kandydatów i DTLS nie jest włączone.

Jeśli DTLS jest włączone, zdarzenie to jest wywoływane po pomyślnym uzgodnieniu DTLS.

# RTCPeerConnection | DTLS

Po ustaleniu prawidłowej pary kandydatów (obaj węzłowie mogą się połączyć i wymieniać dane), należy zabezpieczyć połączenie. DTLS to protokół kryptograficzny szyfrujący dane w celu zapobieżenia inspekcji lub modyfikacji treści wymienianych informacji.

## DTLS wymaga bibliotek OpenSSL (od wersji OpenSSL 1.1+)

Konfiguracja DTLS znajduje się we właściwości `RTCOptions.DTLSOptions` komponentu `TsgcRTCPeerConnection`. Aby włączyć DTLS, należy ustawić właściwość `RTCOptions.DTLS` na `True`. Poniżej przedstawiono główne właściwości:

**RootCertFile:** ścieżka do pliku certyfikatu głównego.

**CertFile:** ścieżka do pliku certyfikatu.

**KeyFile:** ścieżka do pliku klucza certyfikatu.

**VerifyCertificate:** jeśli certyfikat musi być weryfikowany, należy włączyć tę właściwość.

**VerifyDepth:** właściwość całkowita reprezentująca maksymalną liczbę ogniw dozwolonych podczas weryfikacji

**OpenSSL\_Options:** konfiguracja bibliotek OpenSSL.

**APIVersion:** umożliwia zdefiniowanie, które API OpenSSL będzie używane. Wyłącznie OpenSSL API 1.1+ obsługuje DTLS.

**oslpAPI\_1\_1:** używa interfejsu API 1.1 OpenSSL, wymaga niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**oslpAPI\_3\_0:** korzysta z API OpenSSL 3.0, wymaga naszej niestandardowej biblioteki Indy i umożliwia używanie bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek openssl

**oslpNone:** jest to wartość domyślna; biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanej ścieżce.

**oslpDefaultFolder:** automatycznie ustawia ścieżkę openssl, w której powinny znajdować się biblioteki dla wszystkich osobowości IDE.

**oslpCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości `LibPathCustom`.

**LibPathCustom:** gdy `LibPath = oslpCustomFolder`, należy podać pełną ścieżkę do folderu, w którym znajdują się biblioteki openssl.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**oslsSymLinksDefault:** domyślnie jest włączone, z wyjątkiem platformy OSX64 (po MacOS Monterey próba załadowania biblioteki bez wersji kończy się błędem).

**oslsSymLinksLoadFirst:** Załaduj dowiązania symboliczne przed próbą załadowania bibliotek wersji.

**oslsSymLinksLoad:** Ładuj dowiązania symboliczne po próbie załadowania bibliotek wersji.

**oslsSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

## RTCPeerConnection | Dane

Po nawiązaniu połączenia można używać metody **WriteData**, aby wysłać wiadomość do drugiego partnera. Połączenie korzysta z DTLS przez UDP, a jeśli to możliwe, będzie to połączenie P2P (lub z wykorzystaniem adresu relayowanego, gdy P2P nie jest możliwe).

### Wyślij dane

Należy użyć metody WriteData, aby wysłać dowolne dane do drugiego partnera. Możliwe jest wysyłanie zarówno ciągu znaków, jak i tablicy bajtów.

```
TsgcRTCPeerConnection *oRTCPeerConnection = new TsgcRTCPeerConnection();
...
oRTCPeerConnection->WriteData("Hello from sgcWebSockets!!!");
```

### Odbieranie danych

Za każdym razem gdy TsgcRTCPeerConnection otrzyma jakiegokolwiek dane od drugiego uczestnika, wywoływane jest zdarzenie OnRTCMessage.

```
void OnRTCMessage(TObject *Sender, const TBytes aBytes)
{
    ShowMessage(TEncoding->UTF8->GetString(aBytes));
}
```



# Datasnap

Domyślnie DataSnap używa **TIdHttpWebBrokerBridge** jako serwera do obsługi żądań HTTP. Można jednak rozszerzyć możliwości aplikacji DataSnap, zastępując ten serwer innym, który obsługuje więcej protokołów i oferuje znacznie lepszą wydajność, wykorzystując najnowsze protokoły, takie jak HTTP/2 poprawiający wydajność serwera, IOCP umożliwiający obsługę znacznie większej liczby połączeń i inne.

## Serwery

|                                                      |                                                                                    |                                                                                                                                                                                              |
|------------------------------------------------------|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">TsgcWSHTTPWebBrokerBridgeServer</a>      | Protokół WebSocket<br>Protokół HTTP 1.*<br>Protokół XHR<br>IOCP                    | Oparty na bibliotece Indy, obsługuje protokoły WebSocket i HTTP na tym samym porcie. Można również włączyć IOCP.                                                                             |
| <a href="#">TsgcWSHTTP2WebBrokerBridgeServer</a>     | Protokół WebSocket<br>Protokół HTTP 1.*<br>Protokół HTTP/2<br>Protokół XHR<br>IOCP | Oparty na bibliotece Indy, obsługuje protokoły WebSocket i HTTP/2 na tym samym porcie. Możliwe jest również włączenie IOCP.                                                                  |
| <a href="#">TsgcWSServer_HTTPAPI_WebBrokerBridge</a> | Protokół WebSocket<br>Protokół HTTP 1.*<br>Protokół HTTP/2<br>Protokół XHR<br>IOCP | Oparty na interfejsie API HTTP.SYS firmy Microsoft, obsługuje protokoły WebSocket i HTTP/2 na tym samym porcie. Domyślnie nie używane jest IOCP. Zalecany ze względu na najlepszą wydajność. |

# TsgcWSHTTPWebBrokerBridgeServer

**TsgcWSHTTPWebBrokerBridgeServer** korzysta z **TIdHttpWebBrokerBridge** jako podstawy serwera i jest przydatny, gdy wymagane jest użycie jednego serwera do obsługi połączeń DataSnap, HTTP i WebSocket.

**TsgcWSHTTPWebBrokerBridgeServer** dziedziczy po **TsgcWebSocketHTTPServer**, więc można odwoływać się do tego serwera.

Wykonaj następujące kroki, aby zastąpić **TIdHttpWebBrokerBridge** przez **TsgcWSHTTPWebBrokerBridgeServer**:

1. Utwórz nową instancję **TsgcWSHTTPWebBrokerBridgeServer**.
2. Zastąp wszystkie wywołania **TIdHttpWebBrokerBridge** przez **TsgcWSHTTPWebBrokerBridgeServer**.
3. Aby obsługiwać połączenia WebSocket, należy odwołać się do [TsgcWebSocketHTTPServer](#).

## Konfiguracja

Komponenty **Datasnap** znajdują się **wyłącznie w folderze Source**; nie znajdziesz ich w skompilowanych folderach, ponieważ te obiekty nie są dołączone do pakietu **sgcWebSockets**, dlatego należy je tworzyć w czasie wykonania.

Wystarczy dodać wymagane pliki do projektu lub ustawić ścieżkę do folderu **Source** pakietu **sgcWebSockets**. Wymagane pliki:

- `sgcWebSocket_Server_WebBrokerBridge`

Jeśli projekt korzysta z **IdHTTPWebBrokerBridge**, należy zastąpić go komponentem **sgcIdHTTPWebBrokerBridge** (dotyczy wyłącznie edycji Enterprise).

## Zdarzenia

```
FServer = new TsgcWSHTTPWebBrokerBridgeServer();  
FServer->OnCommandRequest = OnCommandRequestEvent;
```

```

FServer->OnCommandGet = OnCommandGetevent;

void OnCommandRequestEvent(TidContext *AThread, TidHTTPRequestInfo *ARequestInfo,
TidHTTPResponseInfo *AResponseInfo, ref bool aHandled)
{
    if (ARequestInfo->Document == "/test.html")
    {
        aHandled = true;
    }
}

void OnCommandGetevent(TidContext *AThread, TidHTTPRequestInfo *ARequestInfo, TidHTTPResponseInfo
*AResponseInfo)
{
    if (ARequestInfo->Document == "/test.html"
    {
        AResponseInfo->ResponseNo = 200;
        AResponseInfo->ContentText = "hello all";
    }
}

```

## Równoważnik obciążenia

Jeśli serwer znajduje się za [TsgcWebSocketLoadBalancerServer](#), mogą wystąpić problemy z CORS. Aby ich uniknąć, należy użyć poniższego kodu.

```

void __fastcall TWebModule1::WebModuleBeforeDispatch(TObject *Sender, TWebRequest *Request, TWebResponse *Response, bool &Handled)
{
    Response->SetCustomHeader("Access-Control-Allow-Origin", "");
    if (Trim(Request->GetFieldByName("Access-Control-Request-Headers")) != "")
    {
        Response->SetCustomHeader("Access-Control-Allow-Headers", Request->GetFieldByName("Access-Control-Request-Headers"));
        Handled = true;
    }
    if (FServerFunctionInvokerAction != nullptr)
    {
        FServerFunctionInvokerAction->Enabled = AllowServerFunctionInvoker;
    }
}

```

# TsgcWSHTTP2WebBrokerBridgeServer

**TsgcWSHTTP2WebBrokerBridgeServer** używa **TsgcWebSocketHTTPServer** z włączonym **protokołem HTTP/2** jako bazą serwera i jest przydatny, gdy chce się używać jednego serwera dla połączeń Data-Snap, HTTP/2 i WebSocket.

**TsgcWSHTTP2WebBrokerBridgeServer** dziedziczy po **TsgcWebSocketHTTPServer**, dlatego można odwoływać się do tego serwera.

Należy wykonać poniższe kroki, aby zastąpić **TIdHttpWebBrokerBridge** przez **TsgcWSHTTP2WebBrokerBridgeServer** :

1. Utwórz nową instancję **TsgcWSHTTP2WebBrokerBridgeServer**.
2. Zastąp wszystkie wywołania **TIdHttpWebBrokerBridge** przez **TsgcWSHTTP2WebBrokerBridgeServer**.
3. Aby obsługiwać połączenia WebSocket, należy odwołać się do [TsgcWebSocketHTTPServer](#).

## Konfiguracja

Komponenty **Datasnap** znajdują się **wyłącznie w folderze Source**; nie znajdziesz ich w skompilowanych folderach, ponieważ te obiekty nie są dołączone do pakietu **sgcWebSockets**, dlatego należy je tworzyć w czasie wykonania.

Wystarczy dodać wymagane pliki do projektu lub ustawić ścieżkę do folderu **Source** pakietu **sgcWebSockets**. Wymagane pliki:

- **sgcWebSocket\_Server\_Base\_WebBrokerBridge**
- **sgcWebSocket\_Server\_WebBrokerBridge\_HTTP2**

Jeśli projekt korzysta z **IdHTTPWebBrokerBridge**, należy zastąpić go komponentem **sgcIdHTTPWebBrokerBridge** (dotyczy wyłącznie edycji Enterprise).

## Zdarzenia

```
FServer = new TsgcWSHTTP2WebBrokerBridgeServer();
FServer->OnCommandRequest = OnCommandRequestEvent;
FServer->OnCommandGet = OnCommandGetevent;

void OnCommandRequestEvent(TIdContext *AThread, TIdHTTPRequestInfo *ARequestInfo,
TIdHTTPResponseInfo *AResponseInfo, ref bool aHandled)
{
    if (ARequestInfo->Document == "/test.html")
    {
        aHandled = true;
    }
}

void OnCommandGetevent(TIdContext *AThread, TIdHTTPRequestInfo *ARequestInfo, TIdHTTPResponseInfo
*AResponseInfo)
{
    if (ARequestInfo->Document == "/test.html")
    {
        AResponseInfo->ResponseNo = 200;
        AResponseInfo->ContentText = "hello all";
    }
}
```

# TsgcWSServer\_HTTPAPI\_WebBrokerBridge

**TsgcWSServer\_HTTPAPI\_WebBrokerBridge** używa **TsgcWebSocketServer\_HTTPAPI** z włączonym protokołem **HTTP/2** jako bazą serwera i jest przydatny, gdy chce się używać jednego serwera do obsługi połączeń DataSnap, HTTP/2 i WebSocket.

**TsgcWSServer\_HTTPAPI\_WebBrokerBridge** dziedziczy po **TsgcWebSocketServer\_HTTPAPI**, dzięki czemu można odwoływać się do tego serwera.

Należy wykonać poniższe kroki, aby zastąpić **TIdHttpWebBrokerBridge** przez **TsgcWSServer\_HTTPAPI\_WebBrokerBridge**:

1. Utwórz nową instancję klasy **TsgcWSServer\_HTTPAPI\_WebBrokerBridge**.
2. Należy zastąpić wszystkie wywołania klasy **TIdHttpWebBrokerBridge** klasą **TsgcWSServer\_HTTPAPI\_WebBrokerBridge**.
3. Aby obsłużyć połączenia WebSocket, należy odwołać się do [TsgcWSServer\\_HTTPAPI\\_WebBrokerBridge](#).

## Konfiguracja

Komponenty **Datasnap** znajdują się **wyłącznie w folderze Source**; nie znajdziesz ich w skompilowanych folderach, ponieważ te obiekty nie są dołączone do pakietu **sgcWebSockets**, dlatego należy je tworzyć w czasie wykonania.

Wystarczy dodać wymagane pliki do projektu lub ustawić ścieżkę do folderu **Source** pakietu **sgcWebSockets**. Wymagane pliki:

- **sgcWebSocket\_Server\_Base\_WebBrokerBridge**
- **sgcWebSocket\_Server\_HTTPAPI\_WebBrokerBridge**

Jeśli projekt korzysta z **IdHTTPWebBrokerBridge**, należy zastąpić go komponentem **sgcIdHTTPWebBrokerBridge** (dotyczy wyłącznie edycji Enterprise).

## Zdarzenia

```
TsgcWSServer_HTTPAPI_WebBrokerBridge FServer = new TsgcWSServer_HTTPAPI_WebBrokerBridge();
FServer->OnCommandRequest += OnCommandRequestEvent;
FServer->OnMessage += OnWebSocketMessage;

void OnCommandRequestEvent(TsgcWSConnection_HTTPAPI *aConnection,
    const THttpRequest *aRequestInfo, ref THttpResponse *aResponseInfo, ref bool aHandled)
{
    if (aRequestInfo->Document == "/test.html")
    {
        aResponseInfo->ResponseNo = 200;
        aResponseInfo->ContentText = "... body ...";
        aHandled = true;
    }
}

void OnWebSocketMessage(TsgcWSConnection *aConnection, const string aText)
{
    aConnection->WriteData(aText);
}
```

# OpenAPI

OpenAPI 3.0

The **Specyfikacja OpenAPI**, wcześniej znana jako **Specyfikacja Swagger**, to specyfikacja plików interfejsu czytelnego maszynowo, służących do opisywania, tworzenia, konsumowania i wizualizacji usług sieciowych RESTful. Pierwotnie stanowiła część frameworka Swagger, w 2016 roku stała się osobnym projektem, nadzorowanym przez OpenAPI Initiative - projekt współpracy open-source w ramach Linux Foundation. Swagger i inne narzędzia mogą generować kod, dokumentację i przypadki testowe na podstawie pliku interfejsu.

Aplikacje zaimplementowane na podstawie plików interfejsu OpenAPI mogą automatycznie generować dokumentację metod, parametrów i modeli. Pomaga to utrzymać spójność dokumentacji, bibliotek klientów i kodu źródłowego.

## Parser Pascal

**sgcOpenAPI Generator** umożliwia automatyczne generowanie bibliotek klienta API (generowanie SDK) na podstawie specyfikacji OpenAPI. Obsługiwane są następujące specyfikacje OpenAPI:

- **OpenAPI 3.\***
- **Swagger 2.\*** (automatycznie konwertowane z wersji 2.0 do 3.0)
- **Swagger 1.\*** (automatycznie konwertowany z 1.0 do 3.0)

sgcOpenAPI umożliwia **automatyczne** generowanie interfejsu klienta API w **natywnym języku Pascal** na podstawie JSON/YAML OpenAPI lub Swagger. Aktualnie obsługiwane są wersje od Delphi 7 do najnowszej wersji Delphi.

**sgcOpenAPI Generator** umożliwia tworzenie pliku dokumentacji na podstawie specyfikacji OpenAPI / Swagger.

Czytaj więcej o [parserze OpenAPI Pascal](#).

## OpenAPI Client

Wygenerowany interfejs klienta zawiera wszystkie funkcje i metody zdefiniowane w specyfikacji OpenAPI. Tworzone są również stałe i wyliczenia.

Obsługiwane są następujące metody **uwierzytelniania**:

- Uwierzytelnianie podstawowe
- [Kod OAuth2](#) (interaktywny)
- [Poświadczenia OAuth2](#) (nieinteraktywne)
- [JWT](#)

Przeczytaj więcej o [kliencie OpenAPI](#).

## Serwer OpenAPI

Możesz także hostować usługę OpenAPI 3.0 w Delphi za pomocą wbudowanego serwera OpenAPI. Serwer automatycznie kieruje żądania zgodnie z Twoją specyfikacją, udostępnia specyfikację pod /openapi.json oraz Swagger UI pod /docs, a także obsługuje CORS i walidację żądań/treści od razu po wyjęciu z pudełka.

Dwa sposoby zdefiniowania API:

- [Spec-First](#) – wczytaj istniejący plik JSON/YAML OpenAPI 3.0 w czasie działania.
- [Code-First](#) – oznacz klasę Delphi atrybutami RTTI; specyfikacja jest generowana automatycznie (wymaga Delphi XE7+).

Przeczytaj więcej o [serwerze OpenAPI](#).

## API

Skompilowano i obsługiwane są następujące API:

- [Amazon AWS](#)
- [Google Cloud APIs](#)
- [Microsoft Azure](#)
- [Inne API](#)

# OpenAPI | Parser Pascal

Parser **sgcOpenAPI** odczytuje **specyfikację OpenAPI 3.0 w formacie JSON** i automatycznie tworzy **klienta Delphi w natywnym kodzie Pascal**.

**Parser sgcOpenAPI** jest zgodny z następującymi specyfikacjami:

- **OpenAPI 3.\***
- **Swagger 2.\*** (automatycznie konwertowane z wersji 2.0 do 3.0)
- **Swagger 1.\*** (automatycznie konwertowany z 1.0 do 3.0)

Plik specyfikacji musi być w formacie **JSON**; jeśli specyfikacja jest w formacie **YAML**, zostanie automatycznie przekonwertowana. Parser umożliwia importowanie specyfikacji podzielonych na **wiele schematów**: w przypadku podania adresu URL podejmie próbę pobrania specyfikacji, a w przypadku podania nazwy pliku spróbuje załadować zewnętrzne schematy z dysku lokalnego.

Plik wykonywalny parsera OpenAPI obsługuje [parametry wiersza poleceń](#) lub może być używany jako [biblioteka](#).

## Importowanie specyfikacji OpenAPI

Pierwszym krokiem jest zaimportowanie specyfikacji openAPI. Po uzyskaniu specyfikacji openAPI 3.0 w formacie JSON można wygenerować wymagane pliki Delphi przy użyciu naszego **OpenAPI WebService**. Należy wykonać poniższe kroki:

- **Uruchom parser sgcOpenAPI** i wybierz specyfikację do zaimportowania lub adres URL do pobrania.



- **Specyfikacja jest weryfikowana** i jeśli jest zgodna z parserem, można przejść do kolejnych kroków. Jeśli specyfikacja korzysta ze starszej wersji, takiej jak swagger 2.0, zostanie automatycznie przekonwertowana. Jeśli specyfikacja jest podzielona na wiele schematów, parser spróbuje połączyć je w jedną specyfikację. W przypadku jakiegokolwiek problemu podczas konwersji pliku pojawi się komunikat o błędzie.





- Teraz można **dostosować następujące opcje** przed przetwarzaniem dokumentu
  - **ClassName:** główna nazwa klasy; domyślnie zaczyna się od "sgcOpenAPI\_" i dodaje nazwę pliku specyfikacji.
  - **Namespace:** nazwa generowanego pliku pascal; domyślnie jest taka sama jak nazwa pliku specyfikacji z rozszerzeniem ".pas".
  - **Create Classes:** jeśli zaznaczone, tworzone są klasy na podstawie pliku specyfikacji. Przykład: jeśli żądanie wymaga wysłania obiektu JSON zdefiniowanego w specyfikacji, parser utworzy klasę z polami tego obiektu JSON.
  - **Enable Classes:** jeśli zaznaczone, włącza klasy generowane z pliku specyfikacji do użycia z obiektami JSON (wymaga Rad Studio XE7+).
  - **Documentation:** jeśli zaznaczone, parser dodaje komentarze do pól, klas i metod.
  - **Authentication:** wybierz metodę uwierzytelniania, jeśli istnieje.
    - **None:** API nie korzysta z żadnej metody uwierzytelniania.
    - **Basic:** API korzysta z uwierzytelniania BASIC.
    - **Token:** wymagane jest wysłanie tokenu jako nagłówka HTTP. Token ten jest uzyskiwany z zewnętrznej metody.
    - **OAuth2:** żądania będą uwierzytelniane za pomocą OAuth2.
    - **JWT:** żądania będą uwierzytelniane za pomocą JWT.



- Teraz zostanie wyświetlona **siatka z listą sparsowanych metod**. Domyślnie parser używa OperationId jako nazwy tworzonych metod, jednak tutaj można użyć podsumowania jako nazwy metody, jeśli OperationId nie jest zdefiniowane lub jego wartość jest nieprawidłowa.



- Na koniec należy sprawdzić, czy **Base URL** ma prawidłową wartość, oraz dostosować ustawienia: czy wygenerowany plik ma być otwierany automatycznie po utworzeniu i czy kreator ma być zamykany po wygenerowaniu pliku.



- Naciśnij **Zakończ**, aby przeanalizować specyfikację i wygenerować plik Pascal.

## Przykład

Zostanie użyta prosta specyfikacja openAPI używana przez abstractapi.com do pobierania lokalizacji adresu IP.

Przed przetestowaniem przykładu należy założyć bezpłatne konto w serwisie abstractapi.com w celu uzyskania klucza API.

<https://app.abstractapi.com/users/signup>

Należy zainstalować [sgcOpenAPI Parser Setup](#), otworzyć sgcOpenAPI.exe i zaimportować specyfikację OpenAPI znajdującą się w folderze "Demos\abstractapi.com\geolocation". Po zaimportowaniu i zapisaniu w tym samym folderze co demo pod nazwą "geolocation.pas" należy otworzyć projekt demo, skompilować go i uruchomić. Następnie należy wypełnić klucz API uzyskany ze strony Abstractapi i nacisnąć przycisk Geolocation.



## OpenAPI | Dodatkowe właściwości

Słownik (znany również jako mapa, hashmap lub tablica asocjacyjna) jest zbiorem par klucz/wartość. OpenAPI umożliwia definiowanie słowników, w których klucze są łańcuchami znaków. Aby zdefiniować słownik, należy użyć type: object oraz słowa kluczowego additionalProperties do określenia typu wartości w parach klucz/wartość. Na przykład słownik typu string-to-string wygląda następująco:

```
{
  "en": "English",
  "fr": "French"
}
```

Parser OpenAPI korzysta z klas JSON firmy Embarcadero. Podczas konwersji TDictionary do formatu JSON, zamiast tworzyć pary klucz/wartość, tworzone są wszystkie wewnętrzne obiekty TDictionary, przez co wynik jest nieprawidłowy. Ten sam problem dotyczy próby konwersji ciągu JSON na obiekt TDictionary.

### Konwertuj AdditionalProperties do formatu JSON

Parser OpenAPI tworzy klasy AdditionalProperties jako typ TsgcAdditionalProperties; jeśli konieczna jest konwersja klasy do ciągu JSON, należy utworzyć nową klasę dziedziczącą z TsgcAdditionalProperties, dodać wszystkie wymagane pola, a następnie przypisać do właściwości klasy.

**Przykład:** dana klasa ma 2 właściwości, jedna z nich to właściwość dodatkowa, a dane wyjściowe JSON muszą mieć postać

```
{"Property1":"value1","AdditionalProperties":{"key1":"value1", "key2":"value"}}
TMyClass = class(TsgcOpenAPIClass)
private
  FProperty1: string;
  FAdditionalProperties: TsgcAdditionalProperties;
public
  property Property1: string read FProperty1 write FProperty1;
  property AdditionalProperties: TsgcAdditionalProperties read FAdditionalProperties write FAdditionalProperties;
end;
```

Utwórz nową klasę dziedziczącą po TsgcAdditionalProperties i zdefiniuj 2 właściwości

```
TCustomAdditionalProperties = class(TsgcAdditionalProperties)
private
  FKey1: string;
```

```
    FKey2: string;
public
    property Key1: string read FKey1 write FKey1;
    property Key2: string read FKey2 write FKey2;
end;
```

Na koniec przypisz tę klasę do właściwości `AdditionalProperties`:

```
TMyClass *oClass = new TMyClass();
oClass->Property1 = "value1";
TCustomAdditionalProperties *oAdditionalProperties = new TCustomAdditionalProperties();
oAdditionalProperties->Key1 = "value1";
oAdditionalProperties->Key2 = "value2";
oClass->AdditionalProperties = oAdditionalProperties;
```

## Konwertuj JSON na `AdditionalProperties`

Ze względu na ograniczenia klas JSON, generator OpenAPI tworzy dodatkową metodę ładowania właściwości `AdditionalProperties` do właściwości `TsgcAdditionalProperties.Dictionary` po przeanalizowaniu ciągu JSON. Aby uzyskać dostęp do zawartości właściwości `AdditionalProperties`, należy odwołać się do właściwości `TsgcAdditionalProperties.Dictionary`.

# OpenAPI | Wiersz poleceń

**Parser sgcOpenAPI** zawiera interfejs wiersza poleceń, który konwertuje specyfikację OpenAPI na plik klienta Delphi (Pascal). Parser waliduje i normalizuje specyfikację (OpenAPI 3 JSON wraz z dołączonymi schematami) przed wygenerowaniem danych wyjściowych.

## Użycie

### Podstawowa składnia

```
sgcOpenAPI.exe -i "c:\openapi.json" -o "c:\openapi.pas"
```

## Parametry

Wszystkie parametry są nierozróżniające wielkości liter. Formy krótkie i długie są równoważne.

- **-i -input** : [wymagane] Ścieżka do specyfikacji OpenAPI lub adres URL do pobrania. Jeśli dane wejściowe są w formacie YAML lub Swagger 1/2, zostaną automatycznie przekonwertowane do formatu OpenAPI 3 JSON.
- **-o -output** : [obowiązkowy] Wyjściowy plik Pascal do wygenerowania (klient Delphi).
- **-nc -nclasses** : [opcjonalnie] Nie generuje klas Delphi dla schematów (tylko interfejsy/metody klienta).
- **-nj -njson** : [opcjonalnie] Nie włączaj serializacji JSON dla generowanych klas.
- **-nd -ndoc** : [opcjonalnie] Nie generuj komentarzy dokumentacyjnych w danych wyjściowych.
- **-m -method** : [opcjonalnie] Strategia nazewnictwa dla generowanych metod:
  - **0** : Użyj **OperationId** (domyślnie).
  - **1** : Użyj **Summary**.
  - **2** : Użyj ścieżki **Endpoint**.
- **-a -auth** : [opcjonalnie] Tryb uwierzytelniania do włączenia w generowanym kliencie:
  - **0** : Brak.
  - **1** : Basic.
  - **2** : Token.
  - **3** : OAuth2.
  - **4** : JWT.
- **-u -url** : [opcjonalnie] Zastępuje podstawowy URL wykryty w pliku OpenAPI.
- **-l -log** : [opcjonalnie] Wyświetla dane wyjściowe dziennika podczas analizy.
- **-user** : [opcjonalny] Nazwa użytkownika licencji (wymagana tylko wtedy, gdy licencja nie została aktywowana).
- **-password** : [opcjonalny] Hasło licencji (wymagane tylko, gdy licencja nie została aktywowana).
- **-h -help** : Wyświetl pomoc wiersza poleceń.

## Przykłady

### 1) Generowanie klienta Delphi z lokalnego pliku

```
sgcOpenAPI.exe -i "c:\specs\openapi.json" -o "c:\clients\orders.pas"
```

### 2) Wygeneruj klienta na podstawie URL, wyłącz klasy i dokumentację

```
sgcOpenAPI.exe -i "https://api.example.com/openapi.json" -o "c:\clients\api.pas" -nc -nd
```

### 3) Używaj podsumowań jako nazw metod i włącz OAuth2

```
sgcOpenAPI.exe -i "c:\specs\openapi.json" -o "c:\clients\auth.pas" -m 1 -a 3
```

### 4) Nadpisz bazowy adres URL i wyświetl dane wyjściowe dziennika

```
sgcOpenAPI.exe -i "c:\specs\openapi.json" -o "c:\clients\override.pas" -u "https://api.example.com/v2" -l
```

## Uwagi

- Jeśli specyfikacja wejściowa jest nieprawidłowa lub nie jest zbundlowanym plikiem JSON OpenAPI 3, parser próbuje automatycznie przekonwertować ją i zbundlować przed wygenerowaniem kodu.
- W przypadku korzystania z adresów URL parser pobiera specyfikację przed jej walidacją i generowaniem danych wyjściowych.

# OpenAPI | Biblioteka

Parser OpenAPI może być używany jako biblioteka współdzielona (DLL), umożliwiając generowanie klientów Pascal z własnych aplikacji bez wywoływania narzędzia wiersza poleceń. Biblioteka eksportuje mały zestaw funkcji opakowujących te same funkcje parsera dostępne w CLI.

## Funkcje biblioteczne

- **CreateOpenAPIPascalFile** — tworzy plik klienta Pascal na podstawie specyfikacji OpenAPI przy użyciu domyślnych opcji.
- **CreateOpenAPIPascalFileEx** — tworzy plik klienta Pascal z pełną kontrolą nad opcjami parsera.
- **GetOpenAPILastErrors** — zwraca ostatni komunikat błędu wygenerowany przez parser.

## CreateOpenAPIPascalFile

Generuje plik klienta Pascal z użyciem domyślnych opcji parsera (generowanie klas, obsługa JSON, dokumentacja, nazwy metod z **OperationId**, brak uwierzytelniania, brak nadpisania bazowego URL).

```
typedef bool (__stdcall *TCreateOpenAPIPascalFile)(const char* aInputFile, const char* aOutputFile);
```

## Parametry

- **aInputFile:** pełna ścieżka (lub adres URL) specyfikacji OpenAPI do sparsowania. Jeśli plik jest w formacie YAML lub Swagger 1/2, parser znormalizuje go do formatu OpenAPI 3 JSON przed wygenerowaniem danych wyjściowych.
- **aOutputFile:** pełna ścieżka jednostki Pascal do utworzenia.
- 

## Wartość zwracana

Zwraca **True**, jeżeli plik został wygenerowany pomyślnie, w przeciwnym razie **False**. W przypadku niepowodzenia należy wywołać metodę **GetOpenAPILastErrors**, aby pobrać komunikat błędu.

## Przykład

## CreateOpenAPIPascalFileEx

Generuje plik klienta Pascal z jawną kontrolą opcji parsera.

```
typedef bool (__stdcall *TCreateOpenAPIPascalFileEx)(
    const char* aInputFile, const char* aOutputFile,
    bool aGenerateClasses, bool aGenerateJSON, bool aGenerateDocumentation,
    int aMethodName, int aAuthentication, const char* aBaseUrl, bool aEnableLog);
```

## Parametry

- **aInputFile:** pełna ścieżka (lub URL) do specyfikacji OpenAPI przeznaczonej do przetwarzania.
- **aOutputFile:** pełna ścieżka jednostki Pascal do utworzenia.
- **aGenerateClasses:** gdy wartość wynosi **True**, generator tworzy modele Pascal (klasy) zdefiniowane w schemacie. Gdy wartość wynosi **False**, generowany jest wyłącznie interfejs klienta.
- **aGenerateJSON:** gdy **True**, w klasach generowane są pomocniki serializacji JSON. Gdy **False**, obsługa JSON jest pomijana.
- **aGenerateDocumentation:** gdy wartość wynosi **True**, generowany jest plik dokumentacji dla specyfikacji OpenAPI.
- **aMethodName:** steruje sposobem generowania nazw metod:
  - **0** = OperationId (domyślnie).
  - **1** = Summary.
  - **2** = Endpoint.
- **aAuthentication:** dodaje pomocniki uwierzytelniania do klienta:
  - **0** = Brak.
  - **1** = Podstawowe.
  - **2** = Token.
  - **3** = OAuth2.
  - **4** = JWT.
- **aBaseUrl:** nadpisz bazowy URL ze specyfikacji OpenAPI. Przekaż pusty ciąg, aby zachować oryginalną wartość.
- **aEnableLog:** gdy wartość wynosi **True**, parser zapisuje komunikaty o postępie i walidacji pomocne przy rozwiązywaniu problemów.

## Wartość zwracana

Zwraca **True**, jeżeli plik został wygenerowany pomyślnie, w przeciwnym razie **False**. W przypadku niepowodzenia należy wywołać metodę **GetOpenAPILastErrors**, aby pobrać komunikat błędu.

## Przykład

### GetOpenAPILastErrors

Zwraca komunikat opisujący ostatni błąd (walidacji, konwersji lub analizy składni) wygenerowany przez parser. Tę funkcję należy wywołać po tym, jak metody **CreateOpenAPIPascalFile** lub **CreateOpenAPIPascalFileEx** zwróci wartość **False**.



```
typedef const char* (__stdcall *TGetOpenAPILastErrors)();
```

### **Wartość zwracana**

Zwraca liczbę błędów

### **Przykład**

# OpenAPI | API

Parser `sgcOpenAPI` umożliwia użycie niestandardowej biblioteki DLL w celu uzyskania bardziej szczegółowych informacji o analizowanym pliku Pascal. Należy utworzyć nową bibliotekę DLL z metodą **`sgcOpenAPI_CreateSection`** i skopiować ją do folderu, w którym znajduje się parser. Za każdym razem, gdy tworzona jest nowa sekcja pliku, API wywoła metodę `CreateSection`, a zawartość można w razie potrzeby zmodyfikować. Poniżej znajduje się prosty przykład Delphi:

```
enum TsgcOpenAPIPascalSectionType {
    soapistHeaderStart, soapistHeaderEnums,
    soapistHeaderClasses, soapistHeaderMainClass, soapistHeaderClassesStructure,
    soapistHeaderMainInstance, soapistHeaderEnd, soapistImplementationStart,
    soapistImplementationClasses, soapistImplementationEnd
};

extern "C" __declspec(dllexport) void __stdcall sgcOpenAPI_CreateSection(
    int aType, wchar_t* &aValue)
{
    switch ((TsgcOpenAPIPascalSectionType)aType) {
        case soapistHeaderStart: break;
        case soapistHeaderEnums: break;
        case soapistHeaderClasses: break;
        case soapistHeaderMainClass: break;
        case soapistHeaderClassesStructure: break;
        case soapistHeaderMainInstance: break;
        case soapistHeaderEnd: break;
        case soapistImplementationStart: break;
        case soapistImplementationClasses: break;
        case soapistImplementationEnd: break;
    }
}
```

# OpenAPI | Klient

**TsgcOpenAPI\_Client** jest komponentem niewizualnym, który hermetyzuje główne metody i właściwości do wykonywania żądań HTTP na podstawie specyfikacji OpenAPI.

Każdy interfejs OpenAPI utworzony za pomocą parsera sgcOpenAPI posiada **2 metody**

1. **GetOpenAPIClient:** jest to funkcja singleton zwracająca instancję klasy głównej; jeśli nie istnieje, jest tworzona automatycznie.
2. **FreeOpenAPIClient:** zwalnia główną klasę, jeśli została utworzona.

## Przykład

Użyj usługi Abstractapi, aby pobrać lokalizację adresu IP.

```
GetOpenAPIClient->Retrieve_the_location_of_an_IP_address("your api", "80.258.15.2");
```

## Uwierzytelnianie

- **Basic:** używa nazwy użytkownika i hasła jako nagłówka HTTP do uwierzytelniania.
  - **UserName:** nazwa użytkownika.
  - **Password:** tajne hasło.
- **OAuth2:** uwierzytelnianie przy użyciu OAuth2 obsługuje 2 typy autoryzacji:
  - **auth2Code:** Stosowany do uwierzytelniania i autoryzacji w większości typów aplikacji, w tym w aplikacjach jednostronicowych, aplikacjach webowych i aplikacjach natywnych. Przepływ umożliwia aplikacjom bezpieczne pozyskiwanie tokenów access\_token, które mogą być używane do uzyskiwania dostępu do chronionych zasobów, a także tokenów odświeżania do pozyskiwania kolejnych tokenów access\_token oraz tokenów ID dla zalogowanego użytkownika.
  - **auth2ClientCredentials:** Ten typ przyznania dostępu jest powszechnie stosowany w interakcjach serwer-serwer, które muszą działać w tle bez bezpośredniej interakcji z użytkownikiem. Aplikacje tego typu są często określane jako procesy demoniczne lub konta usług.
  - Więcej informacji o [OAuth2](#).
- **JWT:** uwierzytelnianie przy użyciu JWT. Więcej informacji na temat [JWT](#).
- **Token:** można przekazać token Bearer lub dowolną inną niestandardową wartość tokena.

## TLSOptions

Umożliwia konfigurację sposobu łączenia z bezpiecznymi serwerami SSL/TLS przy użyciu protokołu HTTP/1

**ALPNProtocols:** lista protokołów ALPN, które zostaną wysłane do serwera.

**RootCertFile:** ścieżka do pliku certyfikatu głównego.

**CertFile:** ścieżka do pliku certyfikatu.

**KeyFile:** ścieżka do pliku klucza certyfikatu.

**Password:** jeśli certyfikat jest zabezpieczony hasłem, należy je tutaj podać.

**VerifyCertificate:** jeśli certyfikat musi być weryfikowany, należy włączyć tę właściwość.

**VerifyDepth:** właściwość całkowitoliczbowa określająca maksymalną liczbę ogniw dozwolonych podczas weryfikacji certyfikatu X.509.

**Version:** domyślnie używa TLS 1.0; jeśli serwer wymaga wyższej wersji TLS, można ją tutaj wybrać.

**IOHandler:** wybierz bibliotekę, która będzie używana do połączenia z wykorzystaniem TLS.

**iohOpenSSL:** korzysta z biblioteki OpenSSL i jest domyślną opcją dla komponentów Indy. Wymaga wdrożenia bibliotek OpenSSL dla win32/win64.

**iohSChannel:** używa kanału Secure Channel — protokołu bezpieczeństwa zaimplementowanego przez Microsoft dla systemu Windows; nie wymaga wdrażania bibliotek openssl. Działa wyłącznie w systemie Windows 32/64 bity.

**OpenSSL\_Options:** konfiguracja bibliotek OpenSSL.

**APIVersion:** pozwala określić, które API OpenSSL będzie używane.

**oslAPI\_1\_0:** używa API 1.0 OpenSSL; jest to najnowsza wersja obsługiwana przez Indy

**oslAPI\_1\_1:** używa interfejsu API 1.1 OpenSSL, wymaga niestandardowej biblioteki Indy i umożliwia korzystanie z bibliotek OpenSSL 1.1.1 (z obsługą TLS 1.3).

**oslAPI\_3\_0:** korzysta z API OpenSSL 3.0, wymaga naszej niestandardowej biblioteki Indy i umożliwia używanie bibliotek OpenSSL 3.0.0 (z obsługą TLS 1.3).

**LibPath:** tutaj można skonfigurować lokalizację bibliotek openssl

**oslpNone:** jest to wartość domyślna; biblioteki OpenSSL powinny znajdować się w tym samym folderze co plik binarny lub w znanej ścieżce.

**oslpDefaultFolder:** automatycznie ustawia ścieżkę openssl, w której powinny znajdować się biblioteki dla wszystkich osobowości IDE.

**oslpCustomFolder:** jeśli ta opcja jest wybrana, należy zdefiniować pełną ścieżkę we właściwości LibPathCustom.

**LibPathCustom:** gdy LibPath = oslpCustomFolder, należy podać pełną ścieżkę do folderu, w którym znajdują się biblioteki openssl.

**UnixSymLinks:** włącz lub wyłącz ładowanie dowiązań symbolicznych w systemach Unix (domyślnie włączone, z wyjątkiem OSX64):

**oslsSymLinksDefault:** domyślnie jest włączone, z wyjątkiem platformy OSX64 (po MacOS Monterey próba załadowania biblioteki bez wersji kończy się błędem).

**oslsSymLinksLoadFirst:** Załaduj dowiązania symboliczne przed próbą załadowania bibliotek wersji.

**oslsSymLinksLoad:** Ładuj dowiązania symboliczne po próbie załadowania bibliotek wersji.

**oslsSymLinksDontLoad:** nie ładuj dowiązań symbolicznych.

**SChannel\_Options:** umożliwia korzystanie z certyfikatu z magazynu certyfikatów systemu Windows.

**CertHash:** to skrót certyfikatu. Skrót certyfikatu można znaleźć, uruchamiając polecenie dir w programie PowerShell.

**CipherList:** w tym miejscu można określić, które szyfry będą używane (oddzielone znakiem ":" ). Przykład: CALG\_AES\_256:CALG\_AES\_128

**CertStoreName:** nazwa magazynu, w którym przechowywany jest certyfikat. Należy wybrać jedną z poniższych opcji:

**scsnMY** (wartość domyślna)

**scsnCA**

**scsnRoot**

**scsnTrust**

**CertStorePath:** ścieżka magazynu, w którym przechowywany jest certyfikat. Wybierz jedną z poniższych opcji:

**scspStoreCurrentUser** (domyślne)

**scspStoreLocalMachine**

## Opcje proxy

Ta właściwość służy do konfiguracji połączeń przez serwer proxy.

**Włączone:** ustaw wartość true, aby włączyć połączenia przez serwer proxy.

**Host:** Adres serwera proxy

**Port:** port serwera proxy

**UserName/Password:** Uwierzytelnianie do połączenia z serwerem proxy, wyłącznie jeśli jest wymagane.

**ProxyType:** obsługiwane są następujące typy proxy:

- HTTP
- Socks4
- Socks4A
- Socks5

## Dziennik

Jeśli właściwość Log jest włączona, komunikaty gniazda są zapisywane do określonego pliku dziennika, co jest przydatne podczas debugowania.

**Log:** włącz, jeśli chcesz zapisywać żądania HTTP do pliku tekstowego.

**LogFileName:** pełna ścieżka do pliku.

## Właściwości

Inne właściwości, które można wykorzystać do dostosowania klienta OpenAPI:

**EncodeBodyAsUTF8:** jeśli włączone, treść JSON jest kodowana jako UTF8 (domyślnie false).

## Zdarzenia

Poniżej znajduje się lista zdarzeń, które można obsługiwać podczas korzystania z klienta OpenAPI.

### OnBeforeRequest

Zdarzenie jest wywoływane przed wysłaniem żądania HTTP. Umożliwia dostosowanie nazw parametrów, nagłówków, zabezpieczeń itp. Poniżej znajduje się przykład pokazujący, jak zastąpić nazwy niektórych parametrów.

```
void __fastcall OnBeforeRequestEvent(TObject *Sender, const TsgcOpenAPIRequest *aRequest)
{
    for (int i = 0; i < aRequest->Parameters->Count; i++)
    {
        TsgcOpenAPIParameter *oParameter = aRequest->Parameters->Items[i];
        if (oParameter->_Name == "meta-modified-from")
            oParameter->_Name = "eventDateTime-from";
        if (oParameter->_Name == "meta-modified-to")
            oParameter->_Name = "eventDateTime-to";
    }
}
```

## **OnUpload**

To zdarzenie jest wywoływane po przesłaniu pliku; można je wykorzystać do śledzenia postępu przesyłania.

## **OnDownload**

Zdarzenie to jest wywoływane po pobraniu pliku; można go używać do śledzenia postępu pobierania.

## **OnSSLVerifyPeer**

Jeśli weryfikacja certyfikatu jest włączona, w tym zdarzeniu można zweryfikować certyfikat serwera i zdecydować, czy go zaakceptować.

## **OnSSLGetHandler**

To zdarzenie jest wywoływane przed utworzeniem procedury obsługi SSL. Można tu utworzyć własną procedurę obsługi SSL (musi dziedziczyć po TIdServerIOHandlerSSLBase lub TIdIOHandlerSSLBase) i ustawić wymagane właściwości.

## **OnSSLAfterCreateHandler**

Jeśli nie utworzono niestandardowego obiektu SSL, zostanie on domyślnie utworzony przy użyciu procedury obsługi OpenSSL. Można uzyskać dostęp do właściwości procedury obsługi SSL i modyfikować je w razie potrzeby.





# OpenAPI | Serwer

Serwer **sgcOpenAPI** hostuje usługę HTTP, której kontrakt jest opisany specyfikacją OpenAPI 3.0. Trasy są budowane automatycznie z sekcji **paths** specyfikacji, interfejs Swagger UI jest udostępniany od razu, a przychodzące żądania mogą być walidowane względem schematów JSON-Schema zadeklarowanych przez każdą operację. Do hostowania serwera dostępne są dwa komponenty:

- **TsgcHTTPServer\_OpenAPI**: samodzielny serwer HTTP. Autonomiczny, osadza serwer HTTP Indy, nie wymaga **sgcWebSockets**. Zalecany, gdy potrzebujesz jedynie udostępnić usługę OpenAPI.
- **TsgcWSAPIServer\_OpenAPI**: zgodny z **sgcWebSockets**. Dołącza się do istniejącego **TsgcWebSocketHTTPServer**, dzięki czemu ten sam serwer obsługuje zarówno ruch WebSocket, jak i HTTP/OpenAPI.

Oba komponenty współdzielą ten sam silnik, opcje i zdarzenia.

## TsgcHTTPServer\_OpenAPI

Samodzielny serwer zdefiniowany w module **sgcHTTPServer\_OpenAPI**. Skonfiguruj port nasłuchiwania, wczytaj specyfikację OpenAPI i aktywuj serwer.

## TsgcWSAPIServer\_OpenAPI

Zgodny z **sgcWebSockets**, zdefiniowany w module **sgcWebSocket\_Server\_API\_OpenAPI**. Przypisz istniejący **TsgcWebSocketHTTPServer** do właściwości **Server**, aby pojedyncza instancja serwera mogła rozdzielać zarówno ramki WebSocket, jak i żądania HTTP OpenAPI. Wewnętrznie deleguje do tego samego silnika, którego używa komponent samodzielny, więc powierzchnia API jest identyczna.

## Wbudowane punkty końcowe

Serwer automatycznie udostępnia następujące punkty końcowe (każdy można wyłączyć poprzez **OpenAPIOptions.Endpoint**):

**/openapi.json**: zwraca wczytaną specyfikację OpenAPI, sterowane przez **OpenAPIOptions.Endpoint.ServeSpec**.

**/docs**: wbudowany interfejs Swagger UI do przeglądania i testowania API, sterowany przez **OpenAPIOptions.Endpoint.ServeSwaggerUI**.

**OPTIONS \***: odpowiedzi preflight CORS, wysyłane, gdy **OpenAPIOptions.CORS.Enabled** ma wartość **True**.

## OpenAPIOptions

Główna właściwość konfiguracyjna, typu **TsgcOpenAPIServer\_Options**. Grupuje trzy podobiektety:

**Endpoint**: wbudowane punkty końcowe i wczytywanie specyfikacji.

**BasePath**: opcjonalny prefiks URL dodawany przed każdą trasą zadeklarowaną w specyfikacji.

**SpecFile**: jeśli przypisany, plik jest wczytywany automatycznie, gdy serwer staje się **Active**.

**ServeSpec:** jeśli True (domyślnie), wczytana specyfikacja jest udostępniana pod / **openapi.json**.

**ServeSwaggerUI:** jeśli True (domyślnie), Swagger UI jest udostępniany pod /**docs**.

**Validation:** flagi walidacji żądania, wszystkie domyślnie False.

**ValidateRequest:** główny przełącznik, który włącza walidację dla całego żądania.

**ValidateRequestBody:** waliduje treść żądania względem JSON-Schema zadeklarowanej w specyfikacji.

**ValidateQueryParams:** waliduje parametry zapytania względem ich zadeklarowanego schematu.

**ValidatePathParams:** waliduje parametry ścieżki względem ich zadeklarowanego schematu.

**ValidateRequired:** odrzuca żądanie, gdy brakuje parametru zadeklarowanego jako wymagany.

**CORS:** obsługa cross-origin.

**Enabled:** jeśli True, nagłówki CORS są dodawane do każdej odpowiedzi, a żądania preflight **OPTIONS** są obsługiwane automatycznie.

**AllowOrigins:** wartość **Access-Control-Allow-Origin** (domyślnie \*).

**AllowHeaders:** wartość **Access-Control-Allow-Headers** (domyślnie **Content-Type, Authorization**).

**AllowMethods:** wartość **Access-Control-Allow-Methods** (domyślnie **GET, POST, PUT, DELETE, PATCH, OPTIONS**).

## Zdarzenia

Poniżej znajduje się lista zdarzeń, które możesz obsługiwać podczas korzystania z serwera OpenAPI.

**OnBeforeRequest:** wywoływane przed rozdzieleniem żądania do **OnRequest**. Ustaw parametr **Accept** na False, aby odrzucić żądanie odpowiedzią 403.

**OnRequest:** główny handler. Sprawdź **aOperationId** i użyj **aContext**, aby odczytać żądanie i zapisać odpowiedź. Po wytworzeniu odpowiedzi ustaw **Handled** na True.

**OnAfterRequest:** wywoływane po powrocie handlera użytkownika. Przydatne do logowania lub metryk.

**OnAuthenticate:** wywoływane przed rozdzieleniem, gdy wymagane jest uwierzytelnienie. Ustaw **Authenticated** na False, aby odrzucić odpowiedzią 401.

**OnValidationError:** wywoływane, gdy jedna lub więcej reguł walidacji nie powiedzie się. Sprawdź **aErrors** i ustaw **Continue** na False, aby odrzucić żądanie odpowiedzią 400.

**OnException:** wywoływane, gdy nieobsłużony wyjątek propaguje z handlera użytkownika. Dostosuj **aResponseCode**, jeśli domyślny kod 500 ma zostać zastąpiony.

## Spec-First a Code-First

Serwer obsługuje dwa podejścia do tworzenia:

- **Spec-First**: plik JSON/YAML OpenAPI 3.0 jest źródłem prawdy; serwer odkrywa trasy, parsując go.
- **Code-First**: klasa Delphi oznaczona atrybutami RTTI jest źródłem prawdy; specyfikacja OpenAPI jest generowana podczas uruchamiania. Wymaga Delphi XE7 lub nowszego.

# OpenAPI | Serwer | Spec-First

W podejściu **Spec-First** specyfikacja OpenAPI 3.0 (JSON lub YAML) jest źródłem prawdy: opisuje każdą ścieżkę, operację, parametr i schemat. Gdy serwer startuje, parsuje specyfikację i automatycznie buduje tablicę tras; musisz jedynie dostarczyć logikę biznesową dla każdej operacji.

Ten przepływ pracy jest zalecany, gdy kontrakt API już istnieje, gdy jest współdzielony z konsumentami lub gdy jest utrzymywany poza bazą kodu Delphi.

## Wczytywanie specyfikacji

Użyj **LoadFromFile**, aby wczytać specyfikację z dysku, lub **LoadFromString**, aby wczytać ją z ciągu znaków w pamięci (na przykład specyfikacji pobranej z bazy danych lub wygenerowanej w czasie działania).

Po wczytaniu tablica tras jest budowana z sekcji **paths** specyfikacji. Segmenty szablonowe, takie jak **/pets/{petId}**, są rozpoznawane automatycznie, a przechwycone wartości są udostępniane przez **aContext.PathParamAsXxx**.

## Obsługa operacji

Serwer rozdziela każde przychodzące żądanie do zdarzenia **OnRequest**, przekazując **operationId** zadeklarowany w specyfikacji. Użyj prostego łańcucha **if/else** (lub wyszukiwania w słowniku), aby skierować operację do właściwego handlera.

## Odczyt danych żądania

Obiekt **TsgcOpenAPIServerContext** udostępnia typizowane akcesory do każdej części żądania:

## Zapisywanie odpowiedzi

Obiekt kontekstu udostępnia także pomocnicze metody do zapisu odpowiedzi. Najczęściej używane to **RespondJSON** i **RespondError**; aby uzyskać pełną kontrolę nad odpowiedzią, ustaw bezpośrednio **aContext.Response.Code** oraz **aContext.Response.ContentText**.

## Walidacja

Gdy włączone są flagi walidacji w **OpenAPIOptions.Validation**, serwer waliduje każde żądanie względem JSON-Schema zadeklarowanych w specyfikacji przed wywołaniem **OnRequest**:

**ValidateRequestBody:** treść żądania jest sprawdzana względem schematu odwoływanego przez **requestBody** operacji.

**ValidateQueryParams:** parametry ciągu zapytania są sprawdzane względem ich zadeklarowanych schematów.

**ValidatePathParams:** parametry ścieżki są sprawdzane względem ich zadeklarowanych schematów.

**ValidateRequired:** żądanie jest odrzucane, gdy brakuje parametru zadeklarowanego jako wymagany.

Gdy walidacja nie powiedzie się, wywoływane jest zdarzenie **OnValidationError** z listą błędów. Ustaw **Continue** na False, aby odrzucić żądanie odpowiedzią 400, lub pozostaw True, aby decyzję podjął Twój handler **OnRequest**.

## Demo

Kompletny przykład spec-first jest dostarczany w **Demos/30.Server/02.SpecFirst/**.

# OpenAPI | Serwer | Code-First

W podejściu **Code-First** klasa Delphi jest źródłem prawdy: metody, parametry i atrybuty RTTI opisują kontrakt API, a specyfikacja OpenAPI 3.0 jest z nich generowana podczas uruchamiania. Po wygenerowaniu specyfikacja jest wczytywana do serwera za pomocą **LoadFromString**, a tablica tras jest budowana tak, jakby pochodziła z pliku.

Code-First wymaga **Delphi XE7 lub nowszego**, ponieważ opiera się na nowym RTTI do inspekcji atrybutów. Funkcjonalność znajduje się w module **sgcHTTP\_OpenAPI\_Server\_CodeFirst** i jest chroniona dyrektywą kompilatora **{\$IFDEF DXE7}**.

## Definiowanie usługi

Oznacz klasę Delphi atrybutami, aby opisać API. Ciała metod są zaślepkami: właściwa logika trafia do zdarzenia **OnRequest** serwera; metody istnieją tylko po to, aby skaner RTTI mógł odczytać ich sygnatury.

## Dostępne atrybuty

Skaner rozpoznaje następujące atrybuty. Atrybuty na poziomie klasy opisują usługę jako całość; atrybuty na poziomie metody opisują pojedynczą operację; atrybuty na poziomie parametru opisują, skąd pochodzi wartość.

**sgcServiceContract(name, description, version):** na poziomie klasy. Wypełnia blok **info** OpenAPI.

**sgcRoute(path):** na poziomie klasy ścieżka bazowa, lub na poziomie metody trasa względem ścieżki bazowej klasy.

**sgcHttpGet, sgcHttpPost, sgcHttpPut, sgcHttpDelete, sgcHttpPatch, sgcHttpHead, sgcHttpOptions:** na poziomie metody. Deklaruje czasownik HTTP operacji.

**sgcSummary(text):** na poziomie metody. Krótkie podsumowanie wyświetlane w specyfikacji i w Swagger UI.

**sgcDescription(text):** na poziomie metody. Długi opis dla specyfikacji.

**sgcTag(name):** na poziomie metody. Grupuje operacje pod tym samym tagiem w Swagger UI.

**sgcResponse(code, description):** na poziomie metody. Deklaruje odpowiedź wraz z jej kodem statusu HTTP i opisem. Może być powtórzony, aby zadeklarować wiele odpowiedzi.

**sgcFromPath, sgcFromQuery, sgcFromBody, sgcFromHeader:** na poziomie parametru. Deklarują, skąd parametr jest odczytywany.

**sgcRequired:** na poziomie parametru. Oznacza parametr jako wymagany; brakujące parametry są zgłaszane przez walidację, gdy włączone jest **ValidateRequired**.

## Generowanie i wczytywanie specyfikacji

Użyj **TsgcOpenAPICodeFirstScanner**, aby wygenerować specyfikację OpenAPI 3.0 z oznaczonej klasy, a następnie wczytaj wynik do serwera.

## Obsługa operacji

Skaner wyprowadza **operationId** z nazwy metody: metoda **ListTasks** staje się operacją **ListTasks**. Rozdzielaj wywołania w **OnRequest**, używając tego samego operationId.

## Demo

Kompletny przykład code-first jest dostarczany w **Demos/30.Server/01.CodeFirst/**.

# OpenAPI | Amazon AWS

Klient **sgcOpenAPI Amazon AWS** (TsgcOpenAPI\_Amazon\_Client) posiada własny klient OpenAPI dziedziczący po [TsgcOpenAPI\\_Client](#).

Ten komponent posiada właściwość o nazwie **AmazonOptions**, która zawiera wszystkie wymagane konfiguracje służące do połączenia z serwerami Amazon AWS.

## AmazonOptions

W sekcji AmazonOptions można zdefiniować wymagane właściwości **AccessKey** i **SecretKey** (które należy wcześniej wygenerować z konta Amazon), służące do uwierzytelniania względem serwerów Amazon AWS.

Klucz dostępu zapewnia programowy dostęp do zasobów. Oznacza to, że klucz dostępu należy chronić równie starannie jak dane logowania do konta głównego AWS.

Zaleca się wykonanie następujących czynności:

1. **Utwórz użytkownika IAM**, a następnie **zdefiniuj uprawnienia tego użytkownika** możliwie jak najwięcej.
2. **Utwórz klucz dostępu** dla tego użytkownika IAM.

Po uzyskaniu poświadczeń należy ustawić je w następujących właściwościach:

- AmazonOptions.AccessKey
- AmazonOptions.SecretKey

Właściwość AmazonOptions.JSON pozwala określić, czy odpowiedzi mają być w formacie JSON czy XML.

Role IAM, użytkownicy w AWS IAM Identity Center (następcy AWS Single Sign-On) oraz użytkownicy federowani posiadają tymczasowe poświadczenia bezpieczeństwa. Tymczasowe poświadczenia bezpieczeństwa wygasają po określonym czasie lub gdy użytkownik kończy sesję. Token dla tymczasowych poświadczeń można ustawić we właściwości:



- `AmazonOptions.SessionToken`

## Najczęstsze zastosowania

- **Konfiguracja**
  - [Dane uwierzytelniające Amazon AWS](#)
- **APIs**
  - [Amazon AWS S3](#)

## sgcOpenAPI AWS SDK

Poniżej znajduje się lista aktualnie dostępnych interfejsów API.

- Access Analyzer
- Alexa For Business
- Amazon API Gateway
- Amazon AppConfig
- Amazon Appflow
- Amazon AppIntegrations Service
- Amazon AppStream
- Amazon Athena
- Amazon Augmented AI Runtime
- Amazon Chime
- Amazon Chime SDK Identity
- Amazon Chime SDK Messaging
- Amazon CloudDirectory
- Amazon CloudFront
- Amazon CloudHSM
- Amazon CloudSearch
- Amazon CloudSearch Domain
- Amazon CloudWatch
- Amazon CloudWatch Application Insights
- Amazon CloudWatch Events
- Amazon CloudWatch Logs
- Amazon CodeGuru Profiler
- Amazon CodeGuru Reviewer
- Amazon Cognito Identity
- Amazon Cognito Identity Provider
- Amazon Cognito Sync
- Amazon Comprehend
- Amazon Connect Contact Lens
- Amazon Connect Customer Profiles
- Amazon Connect Participant Service
- Amazon Connect Service
- Amazon Data Lifecycle Manager
- Amazon Detective
- Amazon DevOps Guru

- Amazon DocumentDB ze zgodnością z MongoDB
- Amazon DynamoDB
- Amazon DynamoDB Accelerator (DAX)
- Amazon DynamoDB Streams
- Amazon EC2 Container Registry
- Amazon EC2 Container Service
- Amazon Elastic Inference
- Amazon Elastic Block Store
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Registry Public
- Amazon Elastic File System
- Amazon Elastic Kubernetes Service
- Amazon Elastic Transcoder
- Amazon ElastiCache
- Amazon Elasticsearch Service
- Amazon EMR
- Amazon EMR Containers
- Amazon EventBridge
- Amazon Forecast Query Service
- Amazon Forecast Service
- Amazon Fraud Detector
- Amazon FSx
- Amazon GameLift
- Amazon Glacier
- Amazon GuardDuty
- Amazon HealthLake
- Amazon Honeycode
- Amazon Import/Export Snowball
- Amazon Inspector
- Amazon Interactive Video Service
- Amazon Kinesis
- Amazon Kinesis Analytics
- Amazon Kinesis Firehose
- Amazon Kinesis Video Signaling Channels
- Amazon Kinesis Video Streams
- Amazon Kinesis Video Streams Archived Media
- Amazon Kinesis Video Streams Media
- Amazon Lex Model Building Service
- Amazon Lex Model Building V2
- Amazon Lex Runtime Service
- Amazon Lex Runtime V2
- Amazon Lightsail
- Usługa lokalizacji Amazon
- Amazon Lookout for Equipment
- Amazon Lookout for Metrics
- Amazon Lookout for Vision
- Amazon Machine Learning
- Amazon Macie
- Amazon Macie 2
- Amazon Managed Blockchain
- Amazon Mechanical Turk
- Amazon MemoryDB
- Amazon Mobile Analytics
- Amazon Neptune

- Amazon OpenSearch Service
- Amazon Personalize
- Amazon Personalize Events
- Amazon Personalize Runtime
- Amazon Pinpoint
- Amazon Pinpoint Email Service
- Amazon Pinpoint SMS and Voice Service
- Amazon Polly
- Amazon Prometheus Service
- Amazon QLDB
- Amazon QLDB Session
- Amazon QuickSight
- Amazon Redshift
- Amazon Rekognition
- Amazon Relational Database Service
- Amazon Route 53
- Amazon Route 53 Domains
- Amazon Route 53 Resolver
- Amazon S3 on Outposts
- Amazon SageMaker Edge Manager
- Amazon SageMaker Feature Store Runtime
- Amazon SageMaker Runtime
- Amazon SageMaker Service
- Amazon Simple Email Service
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- Amazon Simple Storage Service
- Amazon Simple Systems Manager (SSM)
- Amazon Simple Workflow Service
- Amazon SimpleDB
- Amazon Textract
- Amazon Timestream Query
- Amazon Timestream Write
- Amazon Transcribe Service
- Amazon Translate
- Amazon WorkDocs
- Amazon WorkLink
- Amazon WorkMail
- Amazon WorkMail Message Flow
- Amazon WorkSpaces
- AmazonApiGatewayManagementApi
- AmazonApiGatewayV2
- AmazonMQ
- AmazonMWAA
- AmazonNimbleStudio
- AmplifyBackend
- Application Auto Scaling
- Application Migration Service
- Automatyczne skalowanie
- AWS Amplify
- AWS App Mesh
- AWS App Runner
- AWS Application Cost Profiler
- AWS Application Discovery Service

- AWS AppSync
- AWS Audit Manager
- AWS Auto Scaling Plans
- AWS Backup
- AWS Batch
- AWS Budgets
- AWS Certificate Manager
- AWS Certificate Manager Private Certificate Authority
- AWS Cloud Map
- AWS Cloud9
- AWS CloudFormation
- AWS CloudHSM V2
- AWS CloudTrail
- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- AWS CodeStar
- AWS CodeStar connections
- AWS CodeStar Notifications
- AWS Comprehend Medical
- AWS Compute Optimizer
- Konfiguracja AWS
- AWS Cost and Usage Report Service
- AWS Cost Explorer Service
- AWS Data Exchange
- AWS Data Pipeline
- AWS Database Migration Service
- AWS DataSync
- AWS Device Farm
- AWS Direct Connect
- AWS Directory Service
- AWS EC2 Instance Connect
- AWS Elastic Beanstalk
- AWS Elemental MediaConvert
- AWS Elemental MediaLive
- AWS Elemental MediaPackage
- AWS Elemental MediaPackage VOD
- AWS Elemental MediaStore
- AWS Elemental MediaStore Data Plane
- AWS Fault Injection Simulator
- AWS Global Accelerator
- AWS Glue
- AWS Glue DataBrew
- AWS Greengrass
- AWS Ground Station
- AWS Health APIs and Notifications
- AWS Identity and Access Management
- AWS Import/Export
- AWS IoT
- AWS IoT 1-Click Devices Service
- AWS IoT 1-Click Projects Service
- AWS IoT Analytics
- AWS IoT Core Device Advisor

- AWS IoT Data Plane
- AWS IoT Events
- AWS IoT Events Data
- AWS IoT Fleet Hub
- AWS IoT Greengrass V2
- AWS IoT Jobs Data Plane
- AWS IoT Secure Tunneling
- AWS IoT SiteWise
- AWS IoT Things Graph
- AWS IoT Wireless
- AWS Key Management Service
- AWS Lake Formation
- AWS Lambda
- AWS License Manager
- AWS Marketplace Catalog Service
- AWS Marketplace Commerce Analytics
- AWS Marketplace Entitlement Service
- AWS MediaConnect
- AWS MediaTailor
- AWS Migration Hub
- AWS Migration Hub Config
- AWS Mobile
- AWS Network Firewall
- AWS Network Manager
- AWS OpsWorks
- AWS OpsWorks CM
- AWS Organizations
- AWS Outposts
- AWS Performance Insights
- AWS Price List Service
- AWS Proton
- AWS RDS DataService
- AWS Resource Access Manager
- AWS Resource Groups
- AWS Resource Groups Tagging API
- AWS RoboMaker
- AWS Route53 Recovery Control Config
- AWS Route53 Recovery Readiness
- AWS S3 Control
- AWS Savings Plans
- AWS Secrets Manager
- AWS Security Token Service
- AWS SecurityHub
- AWS Server Migration Service
- AWS Service Catalog
- AWS Service Catalog App Registry
- AWS Shield
- AWS Signer
- AWS Single Sign-On
- AWS Single Sign-On Admin
- AWS Snow Device Management
- AWS SSO Identity Store
- AWS SSO OIDC
- AWS Step Functions

- AWS Storage Gateway
- Wsparcie AWS
- AWS Systems Manager Incident Manager
- AWS Systems Manager Incident Manager Contacts
- AWS Transfer Family
- AWS WAF
- AWS WAF Regional
- AWS WAFV2
- AWS Well-Architected Tool
- AWS X-Ray
- AWSKendraFrontendService
- AWSMarketplace Metering
- AWSServerlessApplicationRepository
- Braket
- CodeArtifact
- EC2 Image Builder
- Elastic Load Balancing
- FinSpace Public API
- FinSpace User Environment Management service
- Usługa zarządzania zaporą
- Zarządzane przesyłanie strumieniowe dla Kafka
- Managed Streaming for Kafka Connect
- Redshift Data API Service
- Route53 Recovery Cluster
- Schematy
- Service Quotas
- Synthetics

# OpenAPI Amazon AWS | Poświadczenia

AWS wymaga różnych rodzajów poświadczeń bezpieczeństwa w zależności od sposobu dostępu do AWS. Na przykład do zalogowania się do konsoli zarządzania AWS wymagana jest nazwa użytkownika i hasło, natomiast do **programowych wywołań AWS** potrzebne są **klucze dostępu**.

## Uwagi

- Należy zapisać w bezpiecznym miejscu: adres e-mail powiązany z kontem AWS, identyfikator konta AWS, hasło użytkownika root oraz klucze dostępu konta. Jeśli zapomnisz lub utracisz hasło użytkownika root, aby je zresetować, musisz mieć dostęp do adresu e-mail powiązanego z kontem. Jeśli zapomnisz lub utracisz klucze dostępu, musisz zalogować się na konto, aby utworzyć nowe.
- Zdecydowanie zaleca się utworzenie użytkownika IAM z uprawnieniami administratora do codziennych zadań w AWS i zablokowanie hasła oraz kluczy dostępu użytkownika root. Użytkownika root należy używać wyłącznie do zadań zastrzeżonych dla konta root.
- Dane uwierzytelniające są specyficzne dla konta. Jeśli masz dostęp do wielu kont AWS, każde konto posiada osobne dane uwierzytelniające.
- Nie udostępniaj swoich danych uwierzytelniających AWS osobom trzecim.

## Dostęp programistyczny

W celu wykonywania programowych wywołań do AWS należy podać klucze dostępu AWS.

Podczas tworzenia kluczy dostępu generowany jest identyfikator klucza dostępu (na przykład AKIAIOSFODNN7EXAMPLE) oraz tajny klucz dostępu (na przykład wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY). Tajny klucz dostępu jest dostępny do pobrania wyłącznie w momencie jego tworzenia. Jeśli klucz nie zostanie pobrany lub zostanie utracony, konieczne jest utworzenie nowego.

Można przypisać maksymalnie dwa klucze dostępu na użytkownika (użytkownik root lub użytkownik IAM). Posiadanie dwóch kluczy dostępu jest przydatne podczas ich rotacji. Po wyłączeniu klucza dostępu nie można go używać, jednak wlicza się on do limitu dwóch kluczy. Po usunięciu klucza dostępu znika on bezpowrotnie i nie można go przywrócić, lecz można zastąpić go nowym kluczem.

## Zarządzanie kluczami dostępu po zalogowaniu jako użytkownik root

1. Zaloguj się do konsoli AWS Management Console jako użytkownik root.
2. Na pasku nawigacyjnym w prawym górnym rogu wybierz nazwę lub numer konta, a następnie wybierz **Moje dane uwierzytelniające zabezpieczające**.
3. Rozwiń sekcję **Access keys (access key ID and secret access key)**.

4. Wykonaj jedną z poniższych czynności:

- Aby utworzyć klucz dostępu, należy wybrać opcję **Utwórz nowy klucz dostępu**. Jeśli posiadasz już dwa klucze dostępu, ten przycisk jest nieaktywny i przed utworzeniem nowego klucza należy usunąć jeden z istniejących. Po wyświetleniu monitu należy wybrać opcję **Pokaż klucz dostępu** lub **Pobierz plik klucza**. Jest to jedyna okazja do zapisania tajnego klucza dostępu. Po zapisaniu tajnego klucza dostępu w bezpiecznym miejscu należy wybrać opcję **Zamknij**.
- Aby dezaktywować klucz dostępu, należy wybrać opcję **Dezaktywuj**. Po wyświetleniu monitu o potwierdzenie należy wybrać opcję **Dezaktywuj**. Dezaktywowany klucz dostępu nadal wlicza się do limitu dwóch kluczy dostępu.
- Aby aktywować klucz dostępu, należy wybrać opcję **Aktywuj**.
- Aby usunąć klucz dostępu, gdy nie jest już potrzebny, należy skopiować identyfikator klucza dostępu, a następnie wybrać opcję **Usuń**. Przed usunięciem klucza dostępu należy wybrać opcję **Dezaktywuj**. Zaleca się sprawdzenie, czy klucz dostępu nie jest już używany, przed jego trwałym usunięciem. Aby potwierdzić usunięcie, należy wkleić identyfikator klucza dostępu w polu tekstowym, a następnie wybrać opcję **Usuń**.

## Zarządzanie kluczami dostępu po zalogowaniu jako użytkownik IAM

1. Zalogowanie się do konsoli zarządzania AWS jako użytkownik IAM.
2. Na pasku nawigacyjnym w prawym górnym rogu wybierz swoją nazwę użytkownika, a następnie wybierz **Moje poświadczenia bezpieczeństwa**.
3. Wykonaj jedną z poniższych czynności:
  - Aby utworzyć klucz dostępu, wybierz **Create access key**. Jeśli masz już dwa klucze dostępu, ten przycisk jest wyłączony i przed utworzeniem nowego klucza należy usunąć istniejący. Po wyświetleniu monitu wybierz opcję **Show secret access key** lub **Download .csv file**. To jest jedyna okazja do zapisania tajnego klucza dostępu. Po zapisaniu tajnego klucza dostępu w bezpiecznym miejscu wybierz opcję **Close**.
  - Aby dezaktywować klucz dostępu, wybierz **Make inactive**. Po wyświetleniu monitu o potwierdzenie wybierz opcję **Deactivate**. Zdezaktywowany klucz dostępu nadal wlicza się do limitu dwóch kluczy dostępu.
  - Aby aktywować klucz dostępu, wybierz **Make active**. Po wyświetleniu monitu o potwierdzenie wybierz opcję **Make active**.
  - Aby usunąć klucz dostępu, gdy nie jest już potrzebny, skopiuj jego identyfikator, a następnie wybierz opcję **Delete**. Spowoduje to dezaktywację klucza dostępu. Zaleca się sprawdzenie, czy klucz dostępu nie jest już używany, przed jego trwałym usunięciem. Aby potwierdzić usunięcie, wklej identyfikator klucza dostępu w polu tekstowym, a następnie wybierz opcję **Delete**.



## Konfiguracja sgcOpenAPI

Po uzyskaniu własnych kluczy dostępu AWS należy skonfigurować je w kliencie OpenAPI Amazon przed wykonaniem jakiegokolwiek żądania do serwerów Amazon AWS.

```
GetOpenAPIClient->AmazonOptions->AccessKey = "AKIAIOSFODNN7EXAMPLE";  
GetOpenAPIClient->AmazonOptions->SecretKey = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";
```

# OpenAPI Amazon AWS | S3

Amazon Simple Storage Service (Amazon S3) to usługa przechowywania obiektów oferująca wiodącą w branży skalowalność, dostępność danych, bezpieczeństwo i wydajność. Klienci każdej wielkości i z każdej branży mogą używać Amazon S3 do przechowywania i ochrony dowolnej ilości danych w różnych przypadkach użycia, takich jak jeziora danych, witryny internetowe, aplikacje mobilne, tworzenie kopii zapasowych i przywracanie danych, archiwizacja, aplikacje dla przedsiębiorstw, urządzenia IoT i analiza dużych zbiorów danych. Amazon S3 udostępnia funkcje zarządzania umożliwiające optymalizację, organizowanie i konfigurowanie dostępu do danych w celu spełnienia określonych wymagań biznesowych, organizacyjnych i dotyczących zgodności.

## ListBuckets

```
GetOpenAPIClient->AmazonOptions->AccessKey = "AKIAIOSFODNN7EXAMPLE";  
GetOpenAPIClient->AmazonOptions->SecretKey = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";  
GetOpenAPIClient->ListBuckets();
```

## GetObject

```
GetOpenAPIClient->AmazonOptions->AccessKey = "AKIAIOSFODNN7EXAMPLE";  
GetOpenAPIClient->AmazonOptions->SecretKey = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";  
GetOpenAPIClient->GetObject("bucket_name");
```

## PutObject

```
GetOpenAPIClient->AmazonOptions->AccessKey = "AKIAIOSFODNN7EXAMPLE";  
GetOpenAPIClient->AmazonOptions->SecretKey = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";  
GetOpenAPIClient->PutObject("bucket_name", "MyFile.txt", "payload");
```

# OpenAPI | Google Cloud

Klient **sgcOpenAPI Google** (`TsgcOpenAPI_Google_Client`) posiada własnego klienta OpenAPI, który dziedziczy po [TsgcOpenAPI\\_Client](#).

Ten komponent posiada właściwość o nazwie **GoogleOptions**, która zawiera wszystkie wymagane konfiguracje do połączenia z serwerami Google Cloud.

## GoogleOptions

Klient OpenAPI Google umożliwia uwierzytelnianie przy użyciu następujących metod:

1. **OAuth2 Code**: jest interaktywny, co oznacza konieczność interwencji użytkownika.
2. **JWT (konto usług)**: jest nieinteraktywny, dzięki czemu może działać np. jako usługa.

Uwierzytelnianie jest konfigurowane we właściwości `GoogleOptions.Authentication` i dopuszcza następujące wartości:

- **oagaOAuth2**: interaktywny.
- **oagaJWT**: nieinteraktywny. Ustawienia można importować z pliku JSON za pomocą metody **LoadSettingsFromFile**. Metoda ta automatycznie wypełni następujące właściwości:
  - **ClientEmail**
  - **PrivateKeyId**
  - **PrivateKey**
  - **ServiceAccountOptions**: podczas uruchamiania klienta przy użyciu konta usługi wymagane są następujące właściwości:
    - **TokenURI**: domyślnie wartość to `"https://oauth2.googleapis.com/token"`, ale jest aktualizowana przy użyciu metody `LoadSettingsFromFile`.
    - **Subject**: adres e-mail konta używany przy delegowaniu w całej domenie
    - **Scopes**: lista zakresów.

## Najczęstsze zastosowania

- **Konfiguracja**
  - [Google Cloud OAuth2](#)
  - [Google Cloud Service Accounts](#)
- **APIs**
  - [Google Cloud PubSub](#)
  - [Google Cloud Calendar](#)
  - [Google Cloud Drive](#)

## sgcOpenAPI Google Cloud SDK

Poniżej znajduje się lista aktualnie dostępnych interfejsów API.

- Abusive Experience Report API
- Accelerated Mobile Pages (AMP) URL API
- Access Approval API
- Access Context Manager API
- Ad Exchange Buyer API
- Ad Exchange Buyer API II
- Ad Experience Report API
- Admin SDK API
- AdMob API
- AdSense Host API
- AdSense Management API
- AI Platform Training & Prediction API
- Analytics Reporting API
- Android Device Provisioning Partner API
- Android Management API
- API Discovery Service
- API Gateway API
- API kluczy API
- Apigee API
- App Engine Admin API
- Apps Script API
- Area120 Tables API
- Artifact Registry API
- Assured Workloads API
- Authorized Buyers Marketplace API
- Kopia zapasowa dla GKE API
- Bare Metal Solution API
- BigQuery API
- BigQuery Connection API
- BigQuery Data Transfer API
- BigQuery Reservation API
- Binary Authorization API
- Blogger API v3
- Books API
- Calendar API
- Campaign Manager 360 API
- Certificate Authority API
- Certificate Manager API
- Chrome Management API
- Chrome Policy API
- Chrome UX Report API
- Chrome Verified Access API
- Cloud Asset API
- Cloud AutoML API
- Cloud Bigtable Admin API
- Cloud Billing API
- Cloud Billing Budget API
- Cloud Build API

- Cloud Channel API
- Cloud Composer API
- Cloud Data Fusion API
- Cloud Data Loss Prevention (DLP) API
- Cloud Dataplex API
- Cloud Dataproc API
- Cloud Datastore API
- Cloud Debugger API
- Cloud Deployment Manager V2 API
- Cloud DNS API
- Cloud Document AI API
- Cloud Domains API
- Cloud Filestore API
- Cloud Firestore API
- Cloud Functions API
- Cloud Healthcare API
- Cloud Identity API
- Cloud Identity-Aware Proxy API
- Cloud IDS API
- Cloud IoT API
- Cloud Key Management Service (KMS) API
- Cloud Life Sciences API
- Cloud Logging API
- Cloud Memorystore for Memcached API
- Cloud Monitoring API
- Cloud Natural Language API
- Cloud OS Login API
- Cloud Private Catalog
- Cloud Private Catalog Producer
- Cloud Pub/Sub API
- Cloud Resource Manager API
- Cloud Run Admin API
- Cloud Runtime Configuration API
- Cloud Scheduler API
- Cloud Search API
- Cloud Shell API
- Cloud Source Repositories API
- Cloud Spanner API
- Cloud Speech-to-Text API
- Cloud SQL Admin API
- Cloud Storage for Firebase API
- Cloud Storage JSON API
- Cloud Talent Solution API
- Cloud Tasks API
- Cloud Testing API
- Cloud Text-to-Speech API
- Cloud Tool Results API
- Cloud TPU API
- Cloud Trace API
- Cloud Translation API
- Cloud Video Intelligence API
- Cloud Vision API
- Compute Engine API
- Connectors API

- Contact Center AI Insights API
- Container Analysis API
- Content API for Shopping
- Custom Search API
- Data Labeling API
- API potoków danych
- Database Migration API
- Dataflow API
- Dataproc Metastore API
- Datastream API
- Dialogflow API
- Digital Asset Links API
- Display & Video 360 API
- Domains RDAP API
- DoubleClick Bid Manager API
- Drive Activity API
- Drive API
- Enterprise License Manager API
- API raportowania błędów
- Essential Contacts API
- Eventarc API
- API narzędzi do weryfikacji faktów
- Firebase App Check API
- Firebase Cloud Messaging API
- Firebase Cloud Messaging Data API
- Firebase Dynamic Links API
- Firebase Hosting API
- Firebase Management API
- Firebase ML API
- Firebase Realtime Database Management API
- Firebase Rules API
- Fitness API
- Game Services API
- Genomics API
- GKE Hub API
- Gmail API
- Gmail Postmaster Tools API
- Google Analytics Admin API
- API Google Analytics
- Google Analytics Data API
- Google Chat API
- Google Civic Information API
- Google Classroom API
- Google Cloud Data Catalog API
- Google Cloud Deploy API
- Google Cloud Memorystore for Redis API
- Google Cloud Support API
- Google Docs API
- Google Forms API
- Google Identity Toolkit API
- Google Keep API
- Google Mirror
- Google My Business API
- Google OAuth2 API

- Google Pay Passes API
- Google Play Android Developer API
- Google Play Custom App Publishing API
- Google Play Developer Reporting API
- Google Play EMM API
- Google Play Game Management
- Google Play Game Services
- Google Play Game Services Publishing API
- Google Play Integrity API
- Google Search Console API
- Google Sheets API
- API weryfikacji witryny Google
- Google Slides API
- Google Vault API
- Google Workspace Alert Center API
- Google Workspace Reseller API
- Google+ API
- Groups Migration API
- Groups Settings API
- HomeGraph API
- IAM Service Account Credentials API
- Idea Hub API
- Identity and Access Management (IAM) API
- Indexing API
- Knowledge Graph Search API
- Kubernetes Engine API
- Library Agent API
- Local Services API
- Managed Service for Microsoft Active Directory API
- Manufacturer Center API
- My Business Account Management API
- My Business Business Calls API
- My Business Business Information API
- My Business Lodging API
- My Business Notifications API
- My Business Place Actions API
- My Business Q&A API
- My Business Verifications API
- Network Connectivity API
- Network Management API
- Network Security API
- Network Services API
- Notebooks API
- On-Demand Scanning API
- Organization Policy API
- OS Config API
- PageSpeed Insights API
- Payments Reseller Subscription API
- People API
- Perspective Comment Analyzer API
- Playable Locations API
- Policy Analyzer API
- Policy Simulator API
- Policy Troubleshooter API

- Poly API
- Proximity Beacon API
- Pub/Sub Lite API
- Real-time Bidding API
- reCAPTCHA Enterprise API
- Recommendations AI (Beta)
- Recommender API
- Remote Build Execution API
- Replica Pool
- Resource Settings API
- Retail API
- Safe Browsing API
- SAS Portal API
- SAS Portal API (Testing)
- Search Ads 360 API
- Search Console API
- Secret Manager API
- Security Command Center API
- Security Token Service API
- Semantic Tile API
- Broker usług
- Service Consumer Management API
- Service Control API
- Service Directory API
- Service Management API
- Service Networking API
- Service Usage API
- Smart Device Management API
- Stackdriver Profiler API
- Storage Transfer API
- Street View Publish API
- Tag Manager API
- Tasks API
- Traffic Director API
- Transcoder API
- Version History API
- VM Migration API
- API dla programistów czcionek webowych
- Web Risk API
- Web Security Scanner API
- Workflow Executions API
- Workflows API
- YouTube Analytics API
- YouTube Data API v3
- YouTube Reporting API



# OpenAPI Google Cloud | OAuth2

Aby korzystać z komponentów OpenAPI Google Cloud i uwierzytelniać się przy użyciu OAuth2, należy najpierw uzyskać klucz OAuth2 z Google Cloud.

Poniżej opisano kroki umożliwiające uzyskanie kluczy Google OAuth2 oraz sposób ich konfiguracji w przykładowej aplikacji PubSub.

Najpierw **zaloguj się** do swojego **konta Google Cloud** i użyj istniejącego projektu lub utwórz nowy.

Następnie należy przejść do menu **Credentials** i nacisnąć przycisk **CREATE CREDENTIALS**, a następnie wybrać opcję **OAuth Client ID**.



Wybrać typ aplikacji i ustawić nazwę opisową



W przypadku powodzenia otrzymasz swój **Client Id** i **Client Secret**.



**Nie udostępniaj nikomu danych OAuth2!**

Teraz skopiuj do przykładu OpenAPI Google Cloud



Więcej informacji na temat [konfiguracji OAuth2](#).

Po uwierzytelnieniu można **ponownie się uwierzytelnić**, wywołując najpierw metodę **ClearOAuth2Token** (czyszczącą wszystkie wewnętrzne tokeny OAuth2) , a następnie dowolne żądanie OpenAPI. Zostanie wyświetlona nowa przeglądarka internetowa umożliwiająca ponowne uwierzytelnienie na serwerach Google.

# OpenAPI Google Cloud | Konta usług

Aby użyć **komponentów OpenAPI Google Cloud** i **uwierzytelniać się za pomocą kont usług**, należy najpierw uzyskać certyfikat klucza prywatnego z Google Cloud.

Poniżej przedstawiono kroki umożliwiające uzyskanie certyfikatu Google Private Key oraz sposób jego konfiguracji w przykładowej aplikacji.

Najpierw **zaloguj się** do swojego **konta Google Cloud** i użyj istniejącego projektu lub utwórz nowy.



Wybierz **CREATE SERVICE ACCOUNT** i zostanie wyświetlona nowa strona, na której należy ustawić nazwę i opis konta usługi



Następnie należy wybrać co najmniej jedną rolę. Wybrana zostaje rola PubSub Admin, aby klient mógł publikować i subskrybować tematy, jednak można wybrać inną rolę z mniejszymi uprawnieniami.



Naciśnij CONTINUE, a następnie możesz przyznać dostęp innym użytkownikom



Naciśnij DONE po zakończeniu, a zostanie wyświetlony nowy rekord.



Następnym krokiem jest utworzenie nowego klucza; należy wybrać opcję Utwórz klucz w kolumnie akcji. Należy wybrać JSON, aby pobrać konfigurację w formacie JSON, a nowy klucz zostanie utworzony.



Na koniec należy jedynie uzupełnić dane dostarczone przez Google w kliencie OpenAPI PubSub. Można użyć **LoadSettingsFromFile**, aby wczytać plik konfiguracyjny JSON.



## Delegowanie na poziomie domeny

Jeśli posiadasz konto Google Workspace, administrator organizacji może autoryzować aplikację do dostępu do danych użytkowników w imieniu użytkowników domeny Google Workspace. Na przykład aplikacja korzystająca z **Google Calendar API** w celu dodawania wydarzeń do kalendarzy wszystkich użytkowników domeny Google Workspace używałaby konta usługi do dostępu do Google Calendar API w imieniu użytkowników. Autoryzowanie konta usługi do dostępu do danych w imieniu użytkowników domeny jest czasem nazywane "delegowaniem uprawnień w całej domenie" na rzecz konta usługi.

Aby delegować uprawnienia do całej domeny do konta usługi, administrator domeny Google Workspace musi wykonać następujące kroki:

- W konsoli administracyjnej domeny Google Workspace przejdź do **Menu główne > Bezpieczeństwo > Dostęp i kontrola danych > Kontrola API**.
- W panelu **Domain wide delegation** wybierz **Manage Domain Wide Delegation**.
- Kliknij **Dodaj nowy**.
- W polu **Client ID** wpisz **Client ID** konta usługi. Identyfikator klienta konta usługi można znaleźć na stronie kont usług.
- W polu **Zakresy OAuth (oddzielone przecinkami)** należy wprowadzić listę zakresów, do których aplikacja powinna uzyskać dostęp. Na przykład jeśli aplikacja potrzebuje pełnego dostępu na poziomie domeny do Google Drive API i Google Calendar API, należy wpisać: <https://www.googleapis.com/auth/drive>, <https://www.googleapis.com/auth/calendar>.
- Kliknij przycisk **Authorize**.

Po połączeniu i autoryzacji konta obszaru roboczego należy skonfigurować właściwość `GoogleOptions.ServiceAccountOptions` klienta OpenAPI:

- **Subject:** adres e-mail konta roboczego powiązanego z kontem usługi. Przykład: youremail@domain.com
- **Scopes:** lista zakresów. Przykład: <https://www.googleapis.com/auth/calendar>
- **TokenURI:** domyślnie <https://oauth2.googleapis.com/token>.

# OpenAPI Google Cloud | PubSub

Pub/Sub umożliwia usługom asynchroniczną komunikację z opóźnieniami rzędu 100 milisekund.

Pub/Sub jest używany do strumieniowej analityki i potoków integracji danych w celu pozyskiwania i dystrybucji danych. Jest również skuteczny jako oprogramowanie pośrednie zorientowane na wiadomości do integracji usług lub jako kolejka do równoległego przetwarzania zadań.

## Lista projektów według tematu (OAuth2)

```
GetOpenAPIClient->GoogleOptions->Authentication = oagaOAuth2;  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->AuthURL = "https://accounts.google.com/o/oauth2/auth"  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->TokenURL = "https://accounts.google.com/o/oauth2/token"  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientId = "google client id";  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientSecret = "google client secret";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->IP = "127.0.0.1";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->Port = 8080;  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->Scope->Text = "https://www.googleapis.com/auth/cloud-platform";  
GetOpenAPIClient->ListV1TopicsByProject("projects/pubsub-270909");
```

## Lista projektów według tematu (konto usługi)

```
GetOpenAPIClient->GoogleOptions->Authentication = oagaJWT;  
GetOpenAPIClient->LoadSettingsFromFile("google.json");  
GetOpenAPIClient->ListV1TopicsByProject("projects/pubsub-270909");
```

# OpenAPI Google Cloud Calendar

Calendar API umożliwia integrację aplikacji z Google Calendar, tworząc nowe możliwości angażowania użytkowników.

## Wyświetlanie zdarzeń kalendarza według kalendarza (OAuth2)

```
GetOpenAPIClient->GoogleOptions->Authentication = oagaOAuth2;  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->AuthURL = "https://accounts.google.com/o/oauth2/auth";  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->TokenURL = "https://accounts.google.com/o/oauth2/token";  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientId = "google client id";  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientSecret = "google client secret";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->IP = "127.0.0.1";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->Port = 8080;  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->Scope->Text = "https://www.googleapis.com/auth/calendar";  
GetOpenAPIClient->ListCalendarsEventsByCalendarId("email@mydomain.com", true, "json", 0, "");
```

## Lista zdarzeń według kalendarza (konto usługi)

```
GetOpenAPIClient->GoogleOptions->Authentication = oagaJWT;  
GetOpenAPIClient->LoadSettingsFromFile("google.json");  
GetOpenAPIClient->Authentication->ServiceAccountOptions->Subject = "email@mydomain.com";  
GetOpenAPIClient->Authentication->ServiceAccountOptions->Scopes->Text := "https://www.googleapis.com/auth/calendar";  
GetOpenAPIClient->ListCalendarsEventsByCalendarId("email@mydomain.com", true, "json", 0, "");
```

## Wstawianie zdarzenia kalendarza

```
GetOpenAPIClient->InsertCalendarsEventsByCalendarId("email@mydomain.com", "{\"summary\":\"Test Event\",\"description\":\"Me
```

# OpenAPI Google Cloud | Drive

Użyj jednostki `sgcOpenAPI_googleapis.com_drive_helpers`, jeśli chcesz przesyłać i/lub pobierać pliki z konta Google Drive.

## Przesyłanie plików

Należy użyć metody `UploadFile`, aby przesłać nowy plik na konto Google Drive. Zdarzenie **OnUpload** umożliwia śledzenie postępu przesyłania.

```
void __fastcall UploadFileToGoogleDrive()
{
    TOpenDialog *oDialog = new TOpenDialog(NULL);
    try
    {
        if (oDialog->Execute())
        {
            GetOpenAPIClient()->OnUpload = OnUploadEvent;
            ShowMessage(GetOpenAPIClient()->UploadFile(oDialog->FileName));
        }
    }
    __finally
    {
        delete oDialog;
    }
}

void __fastcall OnUploadEvent(TObject *Sender, TsgcOpenAPIProgressState aState, __int64 aValue)
{
    switch (aState)
    {
        case oahpStart:
            ProgressBar1->Visible = true;
            ProgressBar1->Position = 0;
            ProgressBar1->Max = aValue;
            break;
        case oahpProgress:
            ProgressBar1->Visible = true;
            ProgressBar1->Position = aValue;
            break;
        case oahpEnd:
            ProgressBar1->Visible = false;
            ProgressBar1->Position = 0;
            ProgressBar1->Max = 0;
            break;
    }
    Application->ProcessMessages();
}
```

## Pobierz pliki

Metodą **DownloadFile** należy użyć do pobierania plików z konta Google Drive. Zdarzenie **OnDownload** umożliwia śledzenie postępu pobierania.



```

void __fastcall DownloadFileFromGoogleDrive()
{
    TSaveDialog *oSaveDialog = new TSaveDialog(NULL);
    try
    {
        if (oSaveDialog->Execute())
        {
            GetOpenAPIClient()->OnDownload = OnDownloadEvent;
            GetOpenAPIClient()->DownloadFile(txtDownloadFileId->Text, oSaveDialog->FileName);
        }
    }
    __finally
    {
        delete oSaveDialog;
    }
}

void __fastcall OnDownloadEvent(TObject *Sender, TsgcOpenAPIProgressState aState, __int64 aValue)
{
    switch (aState)
    {
        case oahpStart:
            ProgressBar1->Visible = true;
            ProgressBar1->Position = 0;
            ProgressBar1->Max = aValue;
            break;
        case oahpProgress:
            ProgressBar1->Visible = true;
            ProgressBar1->Position = aValue;
            break;
        case oahpEnd:
            ProgressBar1->Visible = false;
            ProgressBar1->Position = 0;
            ProgressBar1->Max = 0;
            break;
    }
    Application->ProcessMessages();
}

```

## Lista folderów

Lista wszystkich folderów na koncie dysku. Należy użyć metody **ListFiles** i przekazać parametr q jako „application/vnd.google-apps.folder”.

```

GetOpenAPIClient().ListFiles("", true, "", false, false, 100, "", false, false, "", "", "", "", "", "", "", "", "", "", "mi

```

# OpenAPI | Microsoft

**sgcOpenAPI Microsoft Client** (`TsgcOpenAPI_Microsoft_Client`) posiada własnego klienta OpenAPI, który dziedziczy po [TsgcOpenAPI\\_Client](#).

Ten komponent posiada właściwość o nazwie **MicrosoftOptions**, która zawiera wszystkie wymagane konfiguracje do połączenia z serwerami Microsoft.

## MicrosoftOptions

Klient Microsoft OpenAPI umożliwia uwierzytelnianie przy użyciu następujących metod:

1. **OAuth2 Code:** jest interaktywny, co oznacza konieczność interwencji użytkownika.
2. **Poświadczenia OAuth2:** jest nieinteraktywne, więc może być uruchamiane na przykład jako usługa.

Uwierzytelnianie jest konfigurowane we właściwości `MicrosoftOptions.Authentication` i dopuszcza następujące wartości:

- **oamaOAuth2Code:** interaktywny.
- **oamaOAuth2Credentials:** nieinteraktywny.

Inne właściwości wymagane przez Microsoft są następujące:

- **TenantId:** wartość identyfikująca konto, którą można znaleźć na koncie Microsoft/Azure.

## Najczęstsze zastosowania

- **Konfiguracja**
  - [Microsoft Get Tenant](#)
  - [Microsoft Register Application](#)
  - [Microsoft OAuth2 Code](#)
  - [Microsoft OAuth2 Credentials](#)
- **API**
  - [Microsoft Graph](#)

## **sgcOpenAPI Microsoft APIs**

Poniżej znajduje się lista aktualnie dostępnych interfejsów API.

- Klient AutoSuggest
- Klient Computer Vision
- Klient niestandardowego wyszukiwania obrazów
- Klient wyszukiwania niestandardowego
- Klient predykcji Custom Vision
- Custom Vision Training Client
- Klient wyszukiwania encji
- Klient wyszukiwania obrazów
- Klient lokalnego wyszukiwania
- News Search Client
- Partial Graph API
- Klient sprawdzający pisownię
- Klient wyszukiwania wideo
- Klient wyszukiwania wizualnego
- Klient wyszukiwania w sieci

## **sgcOpenAPI Azure APIs**

Poniżej znajduje się lista aktualnie dostępnych interfejsów API.

- Klient API
- ACE Provisioning ManagementPartner
- ADHybridHealthService
- AdvisorManagementClient
- Anomaly Detector Client
- Klient wykrywacza anomalii
- ApiManagementClient
- AppConfiguratiOnManagementClient
- Application Insights Data Plane
- ApplicationClient
- ApplicationInsightsManagementClient
- AppPlatformManagementClient
- AppServiceCertificateOrders API Client
- AppServiceEnvironments API Client
- AppServicePlans API Client
- Artefakt
- AttestationClient
- AuthorizationManagementClient
- AutomationManagement
- AutomationManagementClient
- Azure Action Groups
- Azure Activity Log Alerts
- Azure Addons Resource Provider
- Azure Alerts Management Service Resource Provider
- Azure Bot Service

- Azure CDN WebApplicationFirewallManagement
- Azure Data Catalog Resource Provider
- Azure Data Lake Storage
- Dostawca zasobów Azure Data Migration Service
- Azure Dedicated HSM Resource Provider
- Azure DevOps
- Usługa Azure Enterprise Knowledge Graph
- Azure IoT Central
- Dostawca zasobów usług lokalizacyjnych Azure
- Azure Log Analytics
- Azure Log Analytics — Operations Management
- Azure Log Analytics Query Packs
- Azure Machine Learning Datastore Management Client
- Usługa zarządzania modelami Azure Machine Learning
- Przestrzeń robocza Azure Machine Learning
- Azure Maps Resource Provider
- Azure Media Services
- Azure Metrics
- Azure Migrate Hub
- Azure Migrate V2
- Klient zarządzania planami zobowiązań Azure ML
- Azure ML Web Services Management Client
- Azure Monitor Private Link Scopes
- Azure Reservation
- Azure Resource Graph
- Azure Resource Graph Query
- Azure SQL Database
- Specyfikacja Azure SQL Database API
- Azure SQL Database Backup
- Azure SQL Database Backup Long Term Retention Policy
- Zasady i reguły maskowania danych Azure SQL Database
- Konfiguracje odzyskiwania po awarii bazy danych Azure SQL
- Specyfikacja importu/eksportu Azure SQL Database
- Azure SQL Database replication links
- Specyfikacja Azure SQL Server API
- Azure SQL Server Backup Long Term Retention Vault
- Azure Stack Azure Bridge Client
- azureactivedirectory
- AzureAnalysisServices
- AzureBridgeAdminClient
- AzureDataManagementClient
- AzureDeploymentManager
- AzureDigitalTwinsManagementClient
- AzureStack Azure Bridge Client
- BackupManagementClient
- BatchAI
- BatchManagement
- BatchService
- BillingManagementClient
- BlockchainManagementClient
- BlueprintClient
- CdnManagementClient
- Klient API CertificateRegistrationProvider
- Klient API certyfikatów

- CognitiveServicesManagementClient
- CommerceManagementClient
- Compute Admin Client
- ComputeDiskAdminManagementClient
- ComputeManagementClient
- ComputeManagementConvenienceClient
- Computer Vision
- ConsumptionManagementClient
- ContainerInstanceManagementClient
- ContainerRegistryManagementClient
- ContainerServiceClient
- Content Moderator Client
- Cosmos DB
- CostManagementClient
- Customer Lockbox
- CustomerInsightsManagementClient
- customproviders
- Database Threat Detection Policy APIs
- DataBoxEdgeManagementClient
- DataBoxManagementClient
- DatabricksClient
- DataFactoryManagementClient
- DataLakeAnalyticsAccountManagementClient
- DataLakeAnalyticsCatalogManagementClient
- DataLakeAnalyticsJobManagementClient
- DataLakeStoreAccountManagementClient
- DataLakeStoreFileSystemManagementClient
- DataShareManagementClient
- DeletedWebApps API Client
- DeploymentAdminClient
- DeploymentScriptsClient
- DeviceServices
- DevSpacesManagement
- DevTestLabsClient
- Diagnostics API Client
- DiskResourceProviderClient
- DnsManagementClient
- Dostawca zasobów usług domenowych
- DomainRegistrationProvider API Client
- Klient API domen
- Dynamics Telemetry
- Engagement.ManagementClient
- EngagementFabric
- EventGridManagementClient
- EventHub2018PreviewManagementClient
- EventHubManagementClient
- Usługa wykonawcza
- ExpressRouteCrossConnection REST APIs
- FabricAdminClient
- Face Client
- FeatureClient
- Form Recognizer Client
- FrontDoorManagementClient
- GalleryManagementClient

- Ustawienia diagnostyczne gościa
- Guest Diagnostic Settings Association
- GuestConfiguration
- HanaManagementClient
- HDInsightJobManagementClient
- HDInsightManagementClient
- HealthcareApisClient
- HybridComputeManagementClient
- HybridDataManagementClient
- HyperDrive
- InfrastructureInsightsManagementClient
- Ink Recognizer Client
- InstanceMetadataClient
- IntuneResourceManagementClient
- iotDpsClient
- iotHubClient
- IoTSpacesClient
- KeyVaultClient
- KeyVaultManagementClient
- KustoManagementClient
- LogicAppsManagementClient
- LogicManagementClient
- LUIS Authoring Client
- LUIS Programmatic
- Machine Learning Compute Management Client
- Klient zarządzania obszarami roboczymi uczenia maszynowego
- MaintenanceManagementClient
- ManagedLabsClient
- ManagedNetworkManagementClient
- ManagedServiceIdentityClient
- ManagedServicesClient
- Grupy zarządzania
- ManagementLinkClient
- ManagementLockClient
- MariaDBManagementClient
- Marketplace RP Service
- MarketplaceOrdering.Agreements
- MediaServicesManagementClient
- Microsoft Insights
- Microsoft NetApp
- Microsoft Storage Sync
- Microsoft.ResourceHealth
- Microsoft.Support
- MicrosoftSerialConsoleClient
- Rzeczywistość mieszana
- Klient zarządzania kontami zespołu ML
- MonitorManagementClient
- MySQLManagementClient
- NetworkAdminManagementClient
- NetworkExperiments
- NetworkManagementClient
- NotificationHubsManagementClient
- PeeringManagementClient
- Personalizer Client

- PolicyClient
- PolicyEventsClient
- PolicyMetadataClient
- PolicyStatesClient
- PolicyTrackedResourcesClient
- portal
- PostgreSQLManagementClient
- Power BI Embedded Management Client
- PowerBIDedicated
- PrivateDnsManagementClient
- Klient API dostawcy
- Klient QnAMaker
- QnAMaker Runtime Client
- Recommendations API Client
- RecoveryServicesBackupClient
- RecoveryServicesClient
- RedisManagementClient
- Przekaznik
- RemediationsClient
- ResourceHealthMetadata API Client
- ResourceManagementClient
- Run History APIs
- RunCommandsClient
- SchedulerManagementClient
- SeaBreezeManagementClient
- SearchIndexClient
- SearchManagementClient
- SearchServiceClient
- Security Center
- Analizy zabezpieczeń
- ServerManagement
- Service Fabric Client APIs
- Mapa usług
- ServiceBusManagementClient
- ServiceFabricManagementClient
- SharedImageGalleryServiceClient
- SignalRManagementClient
- SiteRecoveryManagementClient
- Software Plan RP
- SqlManagementClient
- SqlVirtualMachineManagementClient
- Klient zarządzania pamięcią podręczną Storage Cache
- StorageImportExport
- StorageManagementClient
- StorSimple8000SeriesManagementClient
- StorSimpleManagementClient
- StreamAnalyticsManagementClient
- SubscriptionClient
- SubscriptionDefinitionsClient
- SubscriptionsManagementClient
- Text Analytics Client
- TimeSeriesInsightsClient
- Klient API TopLevelDomains
- TrafficManagerManagementClient

- Zarządzanie aktualizacjami
- UpdateAdminClient
- UsageManagementClient
- VirtualMachineImageTemplate
- VirtualWANAsAServiceManagementClient
- Visual Studio Projects Resource Provider Client
- Visual Studio Resource Provider Client
- VM Insights Onboarding
- VMwareCloudSimple
- WebApplicationFirewallManagement
- Klient API WebApps
- WebSite Management Client
- windowsesu
- WorkbookClient
- Workload Monitor



# OpenAPI Microsoft | Tenant

Aby tworzyć aplikacje korzystające z platformy tożsamości Microsoft do zarządzania tożsamością i dostępem, wymagany jest dostęp do *dzierżawcy* Azure Active Directory (Azure AD). W dzierżawcy Azure AD rejestruje się i zarządza aplikacjami, konfiguruje ich dostęp do danych w Microsoft 365 i innych interfejsach API, a także włącza funkcje takie jak dostęp warunkowy.

Dzierżawa reprezentuje organizację. Jest to dedykowane wystąpienie Azure AD, które organizacja lub deweloper aplikacji otrzymuje na początku relacji z firmą Microsoft. Taka relacja może rozpocząć się na przykład od rejestracji w Azure, Microsoft Intune lub Microsoft 365.

Każda dzierżawa Azure AD jest odrębna i oddzielona od innych dzierżaw Azure AD. Posiada własną reprezentację tożsamości służbowych i szkolnych, tożsamości konsumenckich (w przypadku dzierżawy Azure AD B2C) oraz rejestracji aplikacji. Rejestracja aplikacji wewnątrz dzierżawy może zezwalać na uwierzytelnianie tylko z kont w ramach tej dzierżawy lub ze wszystkich dzierżaw.

## Użyj istniejącego dzierżawcy Azure AD

Aby sprawdzić dzierżawcę:

1. Zaloguj się do portalu Azure. Użyj konta, które będzie służyć do zarządzania aplikacją.
2. Należy sprawdzić prawy górny róg. Jeśli istnieje dzierżawa, użytkownik zostanie automatycznie zalogowany. Nazwa dzierżawy jest wyświetlana bezpośrednio pod nazwą konta.

Jeśli konto nie ma powiązanego dzierżawcy, pod nazwą konta zostanie wyświetlony identyfikator GUID. Nie będzie możliwe wykonywanie takich czynności jak rejestrowanie aplikacji, dopóki nie zostanie utworzony dzierżawca Azure AD.

## Utwórz nowego dzierżawcę Azure AD

Należy podać następujące informacje w celu utworzenia nowej dzierżawy:

- **Nazwa organizacji**
- **Domena początkowa** - Domeny początkowej <domainname>.onmicrosoft.com nie można edytować ani usuwać. Można później dodać niestandardową nazwę domeny.
- **Kraj lub region**

# OpenAPI Microsoft | Rejestracja aplikacji

Rejestracja aplikacji ustanawia relację zaufania między aplikacją a platformą tożsamości Microsoft. Zaufanie jest jednostronne: aplikacja ufa platformie tożsamości Microsoft, ale nie odwrotnie.

Wykonaj poniższe kroki, aby utworzyć rejestrację aplikacji:

1. Należy zalogować się do portalu Azure.
2. Jeśli użytkownik ma dostęp do wielu dzierżaw, należy użyć filtra **Katalogi i subskrypcje** w górnym menu, aby przełączyć się na dzierżawę, w której ma zostać zarejestrowana aplikacja.
3. Wyszukaj i wybierz **Azure Active Directory**.
4. W obszarze **Zarządzaj** należy wybrać **Rejestracje aplikacji** > **Nowa rejestracja**.
5. Wprowadź wyświetlaną **nazwę** aplikacji. Użytkownicy aplikacji mogą widzieć wyświetlaną nazwę podczas korzystania z niej, na przykład podczas logowania. Wyświetlaną nazwę można zmienić w dowolnym momencie i wiele rejestracji aplikacji może współdzielić tę samą nazwę. Automatycznie wygenerowany identyfikator Application (client) ID rejestracji aplikacji, a nie jej wyświetlana nazwa, jednoznacznie identyfikuje aplikację w platformie tożsamości.
6. Określa, kto może używać aplikacji — tzw. *odbiorca logowania*.
7. Wybierz **Zarejestruj** aby ukończyć wstępną rejestrację aplikacji



## Dodaj identyfikator URI przekierowania

A *identyfikator URI przekierowania* to lokalizacja, do której platforma tożsamości Microsoft przekierowuje klienta użytkownika i wysyła tokeny bezpieczeństwa po uwierzytelnieniu.

W produkcyjnej aplikacji internetowej URI przekierowania jest zazwyczaj publicznym punktem końcowym, pod którym działa aplikacja, np. `https://contoso.com/auth-response`. Podczas programowania często dodaje się również punkt końcowy lokalnego uruchamiania aplikacji, np. `https://127.0.0.1/auth-response` lub `http://localhost/auth-response`.

Ten **RedirectURI** zostanie użyty później do skonfigurowania **klienta sgcOpenAPI Microsoft**.

## Dodaj dane uwierzytelniające

Poświadczenia są używane przez poufne aplikacje klienckie uzyskujące dostęp do interfejsu API sieci Web. Przykładami poufnych klientów są aplikacje sieci Web, inne interfejsy API sieci Web lub aplikacje usługowe i demonowe. Poświadczenia umożliwiają aplikacji uwierzytelnianie jako jej samej, bez konieczności interakcji z użytkownikiem w czasie wykonywania.

Możliwe jest dodanie zarówno certyfikatów, jak i tajnych kluczy klienta (ciągów znaków) jako danych uwierzytelniających do rejestracji poufnej aplikacji klienckiej.



## Dodaj klucz tajny klienta

Niekiedy określany jako *hasło aplikacji*; tajny klucz klienta to ciąg znaków, którego aplikacja może użyć zamiast certyfikatu do identyfikacji własnej tożsamości.

Tajemnice klienta są uważane za mniej bezpieczne niż poświadczenia certyfikatów. Deweloperzy aplikacji czasem używają tajemnic klienta podczas lokalnego tworzenia aplikacji ze względu na łatwość użycia. Jednak do aplikacji działających w środowisku produkcyjnym należy używać poświadczeń certyfikatów.

1. W portalu Azure w sekcji **Rejestracje aplikacji** wybierz swoją aplikację.
2. Wybierz **Certyfikaty i klucze tajne** > **Klucze tajne klienta** > **Nowy klucz tajny klienta**.
3. Dodaj opis swojego klucza tajnego klienta.
4. Wybierz termin wygaśnięcia dla sekretu lub określ niestandardowy czas istnienia.
  - Czas istnienia sekretu klienta jest ograniczony do dwóch lat (24 miesiące) lub mniej. Nie można określić niestandardowego czasu istnienia dłuższego niż 24 miesiące.
  - Firma Microsoft zaleca ustawienie wartości wygaśnięcia krótszej niż 12 miesięcy.
5. Wybierz **Add**.
6. *Zapisz wartość sekretu* do użycia w kodzie aplikacji klienckiej. Ta wartość sekretu jest *wyświetlana tylko raz* i po opuszczeniu tej strony nie będzie już dostępna.

# OpenAPI Microsoft | OAuth2 Code

Korzystanie z przepływu OAuth2 Code Grant wymaga interakcji z użytkownikiem w celu zalogowania się i uzyskania wymaganych uprawnień.

Po uzyskaniu [identyfikatora najemcy](#) i [danych uwierzytelniających](#) można skonfigurować właściwości OAuth2 dla przepływu Code Grant.

Aby skonfigurować klienta OpenAPI dla OAuth2 Code Grant, należy ustawić właściwość **MicrosoftOptions.Authentication** na następującą wartość:

```
GetOpenAPIClient->MicrosoftOptions->Authentication = oamaOAuth2Code;
```

Następnie można skonfigurować parametry OAuth2



```
GetOpenAPIClient->MicrosoftOptions->Authentication = oamaOAuth2Code;
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->AuthURL = "https://login.microsoftonline.com/tenant_id";
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->TokenURL = "https://login.microsoftonline.com/tenant_id";
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientId = "90945b8d-f6b7-4b97-b2bd-21c3c90b5f3b";
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientSecret = "client_secret";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->IP = "127.0.0.1";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->Port = 8080;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSL = True;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Port = 8080;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->CertFile = "sgc.pem";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->KeyFile = "sgc.pem";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Password = "";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->RedirectURL = "https://localhost:8080";
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->Scope->Text = "openid";
```

Identyfikator dzierżawy (Tenant ID) musi być skonfigurowany w adresach URL uwierzytelniania i tokenów; wystarczy zastąpić prawidłowy Tenant Id w odpowiednim adresie URL.

Microsoft zezwala na adres URL **localhost** wyłącznie w przypadku nasłuchiwania na lokalnym adresie IP, dlatego należy skonfigurować **adres URL przekierowania** z użyciem localhost jako nazwy DNS, a nie adresu IP.

Wartość **scope** zależy od interfejsu API — należy sprawdzić dokumentację Microsoft / Azure dla każdego API.

**Za pierwszym razem**, gdy zostaje wysłane żądanie, przeglądarka internetowa prosi użytkownika o zalogowanie się na konto Microsoft. Jeśli użytkownik logował się wcześniej, żądanie HTTP zostanie wysłane bezpośrednio.

# OpenAPI Microsoft | Dane uwierzytelniające OAuth2

Korzystanie z przepływu OAuth2 Code Grant nie wymaga interakcji z użytkownikiem, więc jest odpowiednie dla usług, demonów lub dowolnych aplikacji działających bez udziału użytkownika.

Po uzyskaniu [identyfikatora najemcy](#) i [danych uwierzytelniających](#) można skonfigurować właściwości OAuth2 dla przepływu Code Grant.

Aby skonfigurować klienta OpenAPI dla OAuth2 Code Grant, należy ustawić właściwość **MicrosoftOptions.Authentication** na następującą wartość:

```
GetOpenAPIClient->MicrosoftOptions->Authentication = oama0Auth2Credentials;
```



```
GetOpenAPIClient->MicrosoftOptions->Authentication = oama0Auth2Credentials;
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->AuthURL = "https://login.microsoftonline.com/tenant_id";
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->TokenURL = "https://login.microsoftonline.com/tenant_id";
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientId = "90945b8d-f6b7-4b97-b2bd-21c3c90b5f3b";
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientSecret = "client_secret";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->IP = "127.0.0.1";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->Port = 8080;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSL = True;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Port = 8080;
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->CertFile = "sgc.pem";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->KeyFile = "sgc.pem";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Password = "";
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->RedirectURL = "https://localhost:8080";
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->Scope->Text = "https://graph.microsoft.com/.default";
```

Identyfikator dzierżawy (Tenant ID) musi być skonfigurowany w adresach URL uwierzytelniania i tokenów; wystarczy zastąpić prawidłowy Tenant Id w odpowiednim adresie URL.

Microsoft zezwala na adres URL **localhost** wyłącznie w przypadku nasłuchiwanie na lokalnym adresie IP, dlatego należy skonfigurować **adres URL przekierowania** z użyciem localhost jako nazwy DNS, a nie adresu IP.

Wartość **scope** zależy od interfejsu API — należy sprawdzić dokumentację Microsoft / Azure dla każdego API.

Dane uwierzytelniające OAuth2 nie wymagają żadnej interakcji użytkownika, dlatego przy pierwszym wywołaniu HTTP żadna przeglądarka nie zostanie otwarta.

# OpenAPI Microsoft | Graph

Microsoft Graph udostępnia REST API i biblioteki klienckie umożliwiające dostęp do danych następujących usług chmurowych Microsoft:

- Podstawowe usługi Microsoft 365: Bookings, Calendar, Delve, Excel, Microsoft 365 compliance eDiscovery, Microsoft Search, OneDrive, OneNote, Outlook/Exchange, People (kontakty Outlook), Planner, SharePoint, Teams, To Do, Workplace Analytics
- Usługi Enterprise Mobility + Security: Advanced Threat Analytics, Advanced Threat Protection, Azure Active Directory, Identity Manager i Intune
- Usługi systemu Windows: aktywności, urządzenia, powiadomienia, Universal Print
- Dynamics 365 Business Central

## Pobierz bieżącego użytkownika

```
GetOpenAPIClient->MicrosoftOptions->Authentication = oama0Auth2Code;  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->AuthURL = "https://login.microsoftonline.com/tenant_i  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->TokenURL = "https://login.microsoftonline.com/tenant_  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientId = "90945b8d-f6b7-4b97-b2bd-21c3c90b5f3b";  
GetOpenAPIClient->Authentication->OAuth2->OAuth2Options->ClientSecret = "client_secret";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->IP = "127.0.0.1";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->Port = 8080;  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSL = True;  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Port = 8080;  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->CertFile = "sgc.pem";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->KeyFile = "sgc.pem";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->SSLOptions->Password = "";  
GetOpenAPIClient->Authentication->OAuth2->LocalServerOptions->RedirectURL = "https://localhost:8080";  
GetOpenAPIClient->Authentication->OAuth2->AuthorizationServerOptions->Scope->Text = "openid";  
GetOpenAPIClient->meuser.GetUser();
```

# API

Poniższe API zostały wygenerowane przy użyciu generatora eSeGeCe sgcOpenAPI i są udostępniane bezpłatnie wszystkim użytkownikom. Kod źródłowy interfejsu jest dostarczany z wersją próbną, dzięki czemu można zobaczyć, czego oczekiwać po zakupie licencji na dowolne prywatne API, takie jak Google, Amazon lub Microsoft.

## Geolokalizacja

Abstrakcyjny interfejs API geolokalizacji IP pobiera adres IP i przekształca go w lokalizację, jak adres, strefa czasowa i inne.



# OpenAPI | AbstractApi Geolocation

## Czym jest API geolokalizacji IP?

Abstract IP Geolocation API pobiera adres IP i przekształca go w lokalizację, a także wiele innych szczegółów, takich jak adres, strefa czasowa i inne.

## Jakie są przykłady zastosowania API geolokalizacji IP?

Istnieje wiele zaawansowanych zastosowań API geolokalizacji IP oraz danych. Obejmują one między innymi:

- Automatycznie przekierowuj użytkowników do odpowiednich witryn lub podwitryn na podstawie ich lokalizacji
- Automatyczne wykrywanie i wyświetlanie lokalizacji, kraju lub strefy czasowej użytkownika bez konieczności jawnego dostosowywania przez użytkownika
- Dostosuj treść lub wygląd witryny internetowej lub aplikacji na podstawie lokalizacji użytkownika. Na przykład wyświetlaj lokalną pogodę, stawki podatkowe i VAT, walutę, wiadomości, dni ustawowo wolne od pracy itp.
- Filtrowanie użytkowników na podstawie ich lokalizacji, np. w przypadku braku możliwości świadczenia usług użytkownikom w określonym kraju.
- Wymaganie od użytkownika zaakceptowania określonych warunków wynikających z lokalnych przepisów, takich jak banery dotyczące plików cookie RODO dla obywateli Unii Europejskiej

## Skąd uzyskać klucz API?

Wystarczy zarejestrować się na [abstractapi.com](https://www.abstractapi.com), aby bezpłatnie otrzymać klucz API

<https://www.abstractapi.com/api/ip-geolocation-api>

## Przykładowy kod

Poniższy kod zwraca informacje o podanym adresie IP.

```
ShowMessage(GetOpenAPIClient->Retrieve_the_location_of_an_IP_address("asdfkjlkj32i3j2liwj3es", "88.5.12.4"));
```

# Dema | Czat serwera

Ten przykład pokazuje, jak zbudować czat serwerowy przy użyciu [TsgcWebSocketHTTPServer](#) i [WebSockets](#) jako protokołu komunikacyjnego.

Za każdym razem, gdy nowy węzeł wysyła wiadomość, serwer odczytuje ją i rozsyła do wszystkich połączonych klientów.

## Uruchom serwer

Przed uruchomieniem serwera należy go skonfigurować, ustawiając port nasłuchiwania, wybierając, czy ma być używane bezpieczne połączenie, itd.

- Najpierw tworzę nową instancję [TsgcWebSocketHTTPServer](#).
- Jeśli serwer będzie używał bezpiecznych połączeń, wymaga certyfikatu PEM; należy podać lokalizację tego certyfikatu i port nasłuchiwania dla SSL. Można skonfigurować wersję TLS i API OpenSSL (w razie potrzeby)

```
// ... ssl
switch (cboOpenSSLAPI->ItemIndex)
{
    case 0:
        server->SSLOptions->OpenSSL_Options.APIVersion = TwsOpenSSLAPI->osslAPI_1_0;
        break;
    case 1:
        server->SSLOptions->OpenSSL_Options->APIVersion = TwsOpenSSLAPI->osslAPI_1_1;
        break;
}
switch (cboTLSVersion->ItemIndex)
{
    case 0:
        server->SSLOptions->Version = TwsTLSVersions->tlsUndefined;
        break;
    case 1:
        server->SSLOptions->Version = TwsTLSVersions->tls1_0;
        break;
    case 2:
        server->SSLOptions->Version = TwsTLSVersions->tls1_1;
        break;
    case 3:
        server->SSLOptions->Version = TwsTLSVersions->tls1_2;
        break;
    case 4:
        server->SSLOptions->Version = TwsTLSVersions->tls1_3;
        break;
    default:
        break;
}
```

- Domyślnie, po uruchomieniu serwera, będzie on nasłuchiwał na WSZYSTKICH adresach IP na porcie nasłuchu; zaleca się używanie właściwości binding, aby nasłuchiwać tylko na 1 konkretnym adresie IP.

```

With WSServer->Bindings->Add do
{
    Port = StrToInt(txtDefaultPort->Text);
    IP = txtHost->Text;
}

```

- Po skonfigurowaniu wszystkich opcji należy wywołać `Server.Active = true`, aby uruchomić serwer.

## Konfiguracja zdarzeń

- Należy używać zdarzeń **OnConnect** i **OnDisconnect**, aby śledzić moment, gdy klient łączy się z serwerem lub rozłącza.
- **Wiadomości** wysyłane przez **klienta do serwera** są odbierane przez zdarzenie **OnMessage**; należy użyć tej procedury obsługi zdarzenia, aby rozgłosić odebraną wiadomość do wszystkich klientów

```

private void OnMessageEvent(TsgcWConnection *Connection, string Text)
{
    server->Broadcast(Text);
}

```

## Wysyłanie żądań HTTP

WebSocket HTTP Server umożliwia obsługę protokołów WebSocket i HTTP na tym samym porcie nasłuchiwania, dzięki czemu przeglądarka może zażądać strony internetowej w celu uzyskania dostępu do serwera. **OnCommandGet** to zdarzenie używane do odczytu żądania HTTP i wysyłania odpowiedzi HTTP.

Należy użyć parametru `ARequestInfo` do odczytu żądania HTTP i `AResponseInfo` do zapisu odpowiedzi HTTP.

Zasadniczo należy użyć właściwości `ARequestInfo.Document`, aby odczytać żądany przez klienta dokument, a następnie wysłać odpowiedź korzystając z właściwości: `ResponseNo`, `ContentType` i `ContentText`.

**Przykład:** klient żąda dokumentu `"/jquery.js"`

```

private void OnCommandGetEvent(TsgcWConnection *Connection, TsgcWSHTTPRequestInfo *RequestInfo, ref TsgcWSHTTPResponseInfo ResponseInfo)
{
    if (RequestInfo->Document == "/jquery.js")
    {
        ResponseInfo->ContentType = "text/javascript";
        ResponseInfo->ContentText = pageJQuery.Content;
        ResponseInfo->ResponseNo = 200;
    }
}

```

} }

# Czat klienta

Ta demonstracja pokazuje, jak zbudować czat kliencki, używając [TsgcWebSocketClient](#), który łączy się z serwerem WebSocket, wysyła wiadomość, a wiadomość ta jest odbierana przez wszystkich połączonych klientów.

## Połącz z serwerem

- Najpierw należy utworzyć nową instancję [TsgcWebSocketClient](#).
- Następnie należy skonfigurować **Host** i **Port** serwera.
- Jeśli klient używa bezpiecznego połączenia, skonfiguruj właściwość **TLSOptions** komponentu.

```
if (chkTLS->Checked)
{
    WSClient->Port = StrToInt(txtSSLPort->Text)
}
else
{
    WSClient->Port = StrToInt(txtDefaultPort->Text);
}
WSClient->Host = txtHost->Text;
case cboOpenSSLAPI->ItemIndex of
{
    0: WSClient->TLSOptions->OpenSSL_Options->APIVersion = sslAPI_1_0;
    1: WSClient->TLSOptions->OpenSSL_Options->APIVersion = sslAPI_1_1;
};
case cboTLSVersion->ItemIndex of
{
    0: WSClient->TLSOptions->Version = tlsUndefined;
    1: WSClient->TLSOptions->Version = tls1_0;
    2: WSClient->TLSOptions->Version = tls1_1;
    3: WSClient->TLSOptions->Version = tls1_2;
    4: WSClient->TLSOptions->Version = tls1_3;
}
WSClient->TLS = chkTLS->Checked;
```

- Po skonfigurowaniu wszystkich opcji należy ustawić `Client.Active = true`, aby połączyć się z serwerem.

## Wyślij wiadomość do serwera

- Aby wysłać wiadomość do serwera, użyj **WriteData**: wyślij dowolną wiadomość tekstową, a serwer odeśle tę samą wiadomość jako odpowiedź.

```
WSClient->WriteData(txtName->Text + ": " + txtMessage->Text);
```

## Odbieranie wiadomości z serwera

- Za każdym razem, gdy klient odbiera nową wiadomość tekstową, **OnMessage** zdarzenie jest wywoływane.

```
private void OnMessageEvent(TsgcWSConnection *Connection, string Text)
{
    DoLog(Text);
}
```

# Przykłady | Klient

Ten przykład pokazuje, jak zbudować klienta WebSocket przy użyciu komponentu [TsgcWebSocketClient](#).

## Połącz z serwerem

- Najpierw należy utworzyć nową instancję [TsgcWebSocketClient](#).
- Następnie należy skonfigurować **Host** i **Port** serwera.
- Domyślnie klient łączy się przy użyciu **protokołu WebSocket**. Istnieje jednak możliwość skonfigurowania klienta tak, aby łączył się przy użyciu **zwykłego protokołu TCP**. Wystarczy ustawić **Specifications.RFC6455 = false**, a klient będzie używać zwykłego protokołu TCP zamiast protokołu WebSocket. Więcej informacji na stronie [Połączenia TCP](#).

```
WSClient->Host = txtHost->Text;
WSClient->Port = StrToInt(txtPort->Text);
WSClient->Options->Parameters = txtParameters->Text;
WSClient->TLS = chkTLS->Checked;
WSClient->Specifications->RFC6455 = chkOverWebSocket->Checked;
WSClient->Proxy->Enabled = chkProxy->Checked;
WSClient->Proxy->Username = txtProxyUsername->Text;
WSClient->Proxy->Password = txtProxyPassword->Text;
WSClient->Proxy->Host = txtProxyHost->Text;
if txtProxyPort->Text <> "" then
WSClient->Proxy->Port = StrToInt(txtProxyPort->Text);
WSClient->Extensions->PerMessage_Deflate->Enabled := chkCompressed->Checked;
// ... active
WSClient->Active = True;
```

## Zdarzenia klienta

Do sterowania przepływem klienta należy używać następujących zdarzeń: przy połączeniu, rozłączeniu, odebraniu wiadomości, wykryciu błędu...

```
private void OnExceptionEvent(TsgcWSConnection *Connection, Exception E)
{
    DoLog("#exception: " + E->Message);
}
private void OnConnectEvent(TsgcWSConnection *Connection)
{
    DoLog("#connected: " + Connection->IP);
}
private void OnMessageEvent(TsgcWSConnection *Connection, string Text)
{
    DoLog(Text);
}
private void OnDisconnectEvent(TsgcWSConnection *Connection, int CloseCode)
{
    DoLog("Disconnected (" + IntToStr(CloseCode) + "): " + Connection->IP);
}
private void OnErrorEvent(TsgcWSConnection *Connection, string Error)
{
    DoLog("#error: " + Connection->IP + " - " + Error);
}
```





# Wersje demonstracyjne | Klient MQTT

Ta demonstracja pokazuje, jak połączyć się z serwerem brokera MQTT. Wymaga komponentu [TsgcWebSocketClient](#) do obsługi protokołów WebSocket / TCP.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketClient](#), sprawdź [Demo klienta](#).
- Następnie utwórz nową instancję [TsgcWSPClient\\_MQTT](#).
- Następnie należy przypisać protokół MQTT do klienta WebSocket i skonfigurować opcje połączenia w kliencie WebSocket.

```
if (mqtt == null)
{
    mqtt = new TsgcWSPClient_MQTT();
    mqtt->OnMQTTBeforeConnect += OnMQTTBeforeConnectEvent;
    mqtt->OnMQTTConnect += OnMQTTConnectEvent;
    mqtt->OnMQTTDisconnect += OnMQTTDisconnectEvent;
    mqtt->OnMQTTSubscribe += OnMQTTSubscribeEvent;
    mqtt->OnMQTTUnSubscribe += OnMQTTUnSubscribeEvent;
    mqtt->OnMQTTPing += OnMQTTPingEvent;
    mqtt->OnMQTTPubAck += OnMQTTPubAckEvent;
    mqtt->OnMQTTPubComp += OnMQTTPubCompEvent;
    mqtt->OnMQTTPublish += OnMQTTPublishEvent;
    mqtt->OnMQTTPubRec += OnMQTTPubRecEvent;
    mqtt->Client = client;
}
mqtt->Client = client;
txtParameters->Text = "/";
chkTLS->Checked = false;
mqtt->Authentication->Enabled = false;
mqtt->Authentication->UserName = "";
mqtt->Authentication->Password = "";
mqtt->MQTTVersion = TwsMQTTVersion->mqtt311;
mqtt->HeartBeat->Interval = 5;
mqtt->HeartBeat->Enabled = true;
switch (Index)
{
    case 0: // esegece.com
        txtHost->Text = "www.esegece.com";
        txtPort->Text = "15675";
        txtParameters->Text = "/ws";
        mqtt->Authentication->Enabled = true;
        mqtt->Authentication->UserName = "sgc";
        mqtt->Authentication->Password = "sgc";
        chkOverWebSocket->Checked = true;
        break;
    case 1: // test.mosquitto.org
        txtHost->Text = "test.mosquitto.org";
        txtPort->Text = "1883";
        chkTLS->Checked = false;
        chkOverWebSocket->Checked = false;
        break;
    case 2: // mqtt.fluux.io
        txtHost->Text = "mqtt.fluux.io";
        txtPort->Text = "1883";
        chkTLS->Checked = false;
        chkOverWebSocket->Checked = false;
```

```

        mqtt->MQTTVersion = TwsMQTTVersion->mqtt5;
        break;
    case 3: // broker.hivemq.com
        txtHost->Text = "broker.mqttdashboard.com";
        txtPort->Text = "8000";
        txtParameters->Text = "/mqtt";
        chkTLS->Checked = false;
        chkOverWebSocket->Checked = true;
        mqtt->MQTTVersion = TwsMQTTVersion->mqtt5;
        break;
}

```

## Zdarzenia MQTT

Przepływ połączeń jest kontrolowany przez komponent klienta MQTT, dlatego należy obsługiwać zdarzenia MQTT, aby wiedzieć, kiedy klient jest połączony z brokerem, kiedy nowa wiadomość jest opublikowana, kiedy zostaje rozłączony itd.

```

private void OnMQTTConnectEvent(TsgcWSConnection *Connection, bool Session, int ReasonCode, string ReasonName, TsgcWSMQTTConnectProperties *Properties)
{
    DoLog("#MQTT Connect");
    chkTLS->Enabled = false;
    chkCompressed->Enabled = false;
    if (FMQTTSubscribeTopic != "")
    {
        mqtt->Subscribe(FMQTTSubscribeTopic);
        FMQTTSubscribeTopic = "";
    }
}

```

```

private void OnMQTTPublishEvent(TsgcWSConnection *Connection, string Topic, string Text, TsgcWSMQTTPublishProperties *Properties)
{
    DoLog(Topic + ": " + Text);
}

```

```

private void OnMQTTSubscribeEvent(TsgcWSConnection *Connection, int PacketIdentifier, TsgcWSSUBACKS *Codes, TsgcWSMQTTSubscribeProperties *Properties)
{
    DoLog("#Subscribe: " + IntToStr(PacketIdentifier));
}

```

```

private void OnMQTTDisconnectEvent(TsgcWSConnection *Connection, int ReasonCode, string ReasonName, TsgcWSMQTTDisconnectProperties *Properties)
{
    DoLog("#disconnected");
    chkTLS->Enabled = true;
    chkCompressed->Enabled = true;
}

```

# Przykłady | Klient SocketIO

Ta demonstracja pokazuje, jak połączyć się z serwerem Socket.IO. Wymaga komponentu [TsgcWebSocketClient](#) do obsługi protokołów WebSocket / TCP.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketClient](#), sprawdź [Demo klienta](#).
- Następnie utwórz nową instancję [TsgcWSAPI\\_SocketIO](#).
- Następnie należy przypisać API Socket.IO do klienta WebSocket i skonfigurować opcje połączenia w kliencie WebSocket.

```
if (socketio == null)
{
    socketio = new TsgcWSAPI_SocketIO();
    socketio->Client = client;
}
socketio->Client = client;
txtParameters->Text = "/";
chkTLS->Checked = true;
chkOverWebSocket->Checked = true;
```

## Wyślij wiadomości

Socket.IO używa komponentu **TsgcWebSocketClient** do wysyłania wiadomości do serwera; wystarczy wywołać metodę **WriteData** i przekazać jako parametr wiadomość JSON przeznaczoną dla serwera socket.io

```
client->WriteData("42[\"new message\", \"\" + txtSocketIOMessage.Text + \"\"]");
```

## Odbieranie wiadomości

Wiadomości odbierane w trakcie nawiązywania połączenia są obsługiwane przez TsgcWebSocketClient, dlatego należy użyć tego komponentu do odczytu wiadomości wysyłanych z serwera i sprawdzania, czy połączenie jest aktywne.

```
private void OnMessageEvent(TsgcWSConnection *Connection, string Text)
{
    DoLog(Text);
}
```

# Demos | Monitor serwera

To demo pokazuje, jak zaktualizować 3 monitory HTML przy użyciu WebSocket jako protokołu. Serwer ma wewnętrzny timer, który losowo aktualizuje wartości wskaźników i przesyła aktualizację za pomocą wiadomości WebSocket. Wiadomość ta jest odczytywana przez klienta JavaScript i aktualizuje wartość wskaźnika.

## Konfiguracja

- Najpierw utwórz nową instancję [TsgcWebSocketServer](#), zapoznaj się z [demonstracją czatu serwera](#).
- Następnie utwórz nowy timer i co 500 milisekund aktualizuj wartości: pamięci, sieci lub procesora. Wyślij aktualizację do wszystkich połączonych klientów.
- W kliencie JavaScript należy odczytać wiadomość wysłaną przez serwer i zaktualizować wartość miernika.

```
<pre><code>
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Server Monitor Demo</title>
  <script src="http://127.0.0.1:5413/sgcWebSockets.js"></script>
  <script src="http://127.0.0.1:5413/esegece.com.js"></script>
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
  <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"></script>
  <script type='text/javascript' src='https://www.google.com/jsapi'></script>
  <style>
    #status {
      padding: 5px;
      color: #fff;
      background: #ccc;
    }
    #status.fail {
      background: #c00;
    }
    #status.offline {
      background: #c00;
    }
    #status.online {
      background: #0c0;
    }
  </style>
  <script type='text/javascript'>
    var vMemory;
    var vCpu;
    var vNetwork;
    var chart;
    var data;
    var options;
    var ws;

    vMemory=30;
    vCpu=55;
    vNetwork=68;
    google.load('visualization', '1', {packages:['gauge']});
    google.setOnLoadCallback(drawChart);
    function drawChart() {
```

```

data = google.visualization.arrayToDataTable([
    ['Label', 'Value'],
    ['Memory', vMemory],
    ['CPU', vCpu],
    ['Network', vNetwork]
]);
options = {
    width: 400, height: 120,
    redFrom: 90, redTo: 100,
    yellowFrom: 75, yellowTo: 90,
    minorTicks: 5
};
chart = new google.visualization.Gauge(document.getElementById('chart_div'));
chart.draw(data, options);
}

function updateChart() {
    data = google.visualization.arrayToDataTable([
        ['Label', 'Value'],
        ['Memory', vMemory],
        ['CPU', vCpu],
        ['Network', vNetwork]
    ]);
    chart.draw(data, options);
}

function subscribe(Channel)
{
    if (document.getElementById(Channel).checked) {
        ws.subscribe(Channel);
    } else {
        ws.unsubscribe(Channel);
    }
}

function wsmonitor()
{
    if ("WebSocket" in window)
    {
        ws = new SGCWS("ws://127.0.0.1:5413");
        ws.on('open', function(evt){
            document.getElementById('status').innerHTML = "Socket Open";
            document.getElementById('status').className = "online";
            ws.subscribe("memory");
            ws.subscribe("cpu");
            ws.subscribe("network");
        });
        ws.on('close', function(evt){
            document.getElementById('status').innerHTML = "Socket Closed";
            document.getElementById('status').className = "offline";
        });
        ws.on('sgcEvent', function(evt){
            if (evt.channel == "memory") {
                vMemory = parseInt(evt.message);
            } else if (evt.channel == "cpu") {
                vCpu = parseInt(evt.message);
            } else if (evt.channel == "network") {
                vNetwork = parseInt(evt.message);
            }
            updateChart();
        });
        ws.on('error', function(evt){
            document.getElementById('status').innerHTML = "Socket Error";
            document.getElementById('status').className = "fail";
        });
    }
}

```

```

        }
    };
}
}
</script>
</head>
<body>
<div data-role="page" id="wsdemo_monitor">
    <div data-role="header" data-theme="b">
        <h1>Server Monitor</h1>
        <a href="#home" data-icon="home" data-iconpos="notext" data-direction="reverse" class="ui-btn-left jqm-home"></a>
    </div><!-- /header -->
    <div data-role="content">
        <h2>Press Start to Get Monitor Data</h2>
        <p id="status" class="success"></p>
        <h4>Select which data you want to receive: Memory - CPU - Network</h4>
        <a href="javascript:wsmonitor()" data-role="button" data-inline="true">Start</a>
        <div id='chart_div'></div>
        <div data-role="fieldcontain">
            <fieldset data-role="controlgroup" data-type="horizontal">
                <input type="checkbox" name="memory" id="memory" class="custom" checked="True" onclick="javascript:wsmonitor()" />
                <label for="memory">Memory</label>
                <input type="checkbox" name="cpu" id="cpu" class="custom" checked="True" onclick="javascript:wsmonitor()" />
                <label for="cpu">CPU</label>
                <input type="checkbox" name="network" id="network" class="custom" checked="True" onclick="javascript:wsmonitor()" />
                <label for="network">Network</label>
            </fieldset>
        </div>
    </div><!-- /content -->
    <div data-role="footer" class="footer-docs" data-theme="c">
        <p>&copy; 2020 eSeGeCe.com</p>
    </div>
</div><!-- /page -->
</body>
</html>
</code></pre>

```

# Dema | Migawki serwera

Ta demonstracja pokazuje, jak wysyłać obrazy z serwera do klienta i jak wszyscy klienci otrzymują ten sam obraz za pomocą metody broadcast komponentu serwera.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketHTTPServer](#) i zapoznać się z [demonstracją czatu serwera](#).
- Włącz kompresję, aby wysyłać mniej bajtów podczas przesyłania wiadomości do klientów

```
Server.Extensions.PerMessage_Deflate.Enabled = true
```

- Następnie co 5 sekund serwer rozgłasza strumień obrazów do wszystkich podłączonych klientów

```
void DoBroadcastStream()
{
    TBitmap *oBitmap = new TBitmap();
    try
    {
        // load bitmapmap
        if (WSServer->Active)
        {
            TMemoryStream *oStream = new TMemoryStream();
            Try
            {
                {
                    oBitmap->SaveToStream(oStream);
                    oStream->Seek(0, soFromBeginning);
                    WSServer->Broadcast(oStream);
                }
            }__finally
            {
                FreeAndNil(oStream);
            }
        }
    }__finally
    {
        oBitmap->FreeImage;
        FreeAndNil(oBitmap);
    }
}
```

# Demonstracje | Migawki klienta

Niniejsza demonstracja pokazuje, jak odczytywać binarne wiadomości WebSocket przy użyciu [TsgcWebSocketClient](#), który łączy się z serwerem WebSocket i odbiera strumień będący obrazem wyświetlanym użytkownikowi.

## Połącz z serwerem

- Najpierw należy utworzyć nową instancję [TsgcWebSocketClient](#).
- Następnie należy skonfigurować **Host** i **Port** serwera.
- Włącz kompresję, aby odbierać mniej bajtów podczas przesyłania wiadomości z serwera.

```
Client.Extensions.PerMessage_Deflate.Enabled = true
```

- Obraz wysłany przez serwer dociera jako strumień, dlatego do odczytu obrazów należy użyć zdarzenia **OnBinary**.

```
void OnClientBinary(TsgcWSConnection *Connection, TMemoryStream *Data)
{
    TBitmap *oBitmap = new TBitmap();
    try
    {
        oBitmap->LoadFromStream(Data);
        Image1->Picture->Assign(oBitmap);
        memoLog->Lines->Add(
            "#image uncompressed size: " + IntToStr(Data->Size) +
            ". Total received: " + IntToStr(Connection->RecBytes));
    }
    __finally
    {
        FreeAndNil(oBitmap);
    }
}
```



# Demonstracje | Przesyłanie pliku

To demo pokazuje, jak przesłać plik z przeglądarki internetowej na serwer przy użyciu protokołu WebSocket.

## Konfiguracja

- Najpierw utwórz nową instancję [TsgcWebSocketServer](#), zapoznaj się z [demonstracją czatu serwera](#).
- Plik dotrze do serwera jako strumień binarny, dlatego do odczytania pliku należy obsłużyć zdarzenie **OnBinary**.

```
void OnServerBinary(TsgcWSConnection *Connection, const TMemoryStream *Data)
{
    if (FFileName == "")
        FFileName = FormatDateTime("yyyymmddhhnnsszz", Now) + ".dat";
    TFileStream *oFile = new TFileStream(FFileName, fmCreate);
    try
    {
        oFile->CopyFrom(Data, Data->Size);
    }
    __finally
    {
        delete oFile;
    }
    memoLog->Lines->Add("Received File: " + FFileName);
}
```

- Aby poznać nazwę pliku, można przed jego wysłaniem przesłać wiadomość tekstową zawierającą tę nazwę.

```
void OnServerMessage(TsgcWSConnection *Connection, const String Text)
{
    if (LeftStr(Text, Length(CS_UPLOAD_FILE)) == CS_UPLOAD_FILE)
        FFileName = MidStr(Text, Length(CS_UPLOAD_FILE) + 1, Length(Text));
    else
        memoLog->Lines->Add("Message Received (" + Connection->Guid + "): " + Text);
}
```

Poniżej przedstawiono kod **javascript** do wysyłania pliku przy użyciu **websockets**:

```
<script type='text/javascript'>
var ws;
function DoOpen()
{
    if ("WebSocket" in window)
    {
        ws = new sgcWebSocket("ws://127.0.0.1:5418");
        ws.on('open', function(evt){
            ws.binaryType = "arraybuffer";
            document.getElementById('status').innerHTML = "Socket Open";
            document.getElementById('status').className = "online";
        });
        ws.on('close', function(evt){
            document.getElementById('status').innerHTML = "Socket Closed";
        });
    }
}
```

```

        document.getElementById('status').className = "offline";
    }
    );
    ws.on('error', function(evt){
        document.getElementById('status').innerHTML = "Socket Error";
        document.getElementById('status').className = "fail";
    }
    );
}
}
function DoClose()
{
    ws.close();
}
function DoUploadFile() {
    var file = document.getElementById('filename').files[0];
    var reader = new FileReader();
    var rawData = new ArrayBuffer();

    reader.onloadend = function() {

    }
    reader.onload = function(e) {
        ws.send("uploadfile:" + file.name);
        rawData = e.target.result;
        ws.send(rawData);
        document.getElementById('status').innerHTML = "File Uploaded";
        document.getElementById('status').className = "online";
    }
    reader.readAsArrayBuffer(file);
}
}
</script>

```

# Wersje demonstracyjne | Uwierzytelnianie serwera

Ta demonstracja pokazuje, jak używać uwierzytelniania serwera. Aby dowiedzieć się więcej o różnych typach uwierzytelniania, należy zapoznać się z artykułem o [uwierzytelnianiu](#).

## Uwierzytelnianie

- Należy najpierw utworzyć nową instancję [TsgcWebSocketServer](#). Włącz właściwość Authentication: `server.Authentication.Enabled = true;`
- Następnie w procedurze obsługi zdarzenia **OnAuthentication** należy sprawdzić, czy nazwa użytkownika i hasło są poprawne. Jeśli są poprawne, należy ustawić właściwość `Authenticated` na `true`, w przeciwnym razie na `false`.

```
private void OnAuthenticationEvent(TsgcWSConnection *Connection, string User, string Password, ref bool Authenticated)
{
    if ((User == "user") && (Password == "1234"))
    {
        Authenticated = true;
    }
}
```

# Demos | KendoUI\_Grid

Ta demonstracja pokazuje, jak działa siatka KendoUI Grid z WebSockets jako protokołem i przeglądarką internetową jako klientem. Jest to siatka JavaScript aktualizowana w momencie dokonania zmiany przez dowolnego klienta; zmiany te są rozsyłane za pomocą protokołu WebSocket do wszystkich podłączonych klientów, dzięki czemu wszyscy klienci widzą w czasie rzeczywistym te same dane, w tym wszystkie zmiany wprowadzone przez innych klientów.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketHTTPServer](#) i zapoznać się z [demonstracją czatu serwera](#).
- Następnie należy obsługiwać zdarzenie OnCommandGet, aby wysyłać wymagane pliki żądane przez klientów przeglądarki internetowej.

```
void OnCommandGet(TidContext *AContext, TidHTTPRequestInfo *ARequestInfo, TidHTTPResponseInfo *AResponseInfo)
{
    if (ARequestInfo->Document == "/jquery.mobile.css")
    {
        AResponseInfo->ContentText = pagejQueryMobileCSS->Content;
        AResponseInfo->ContentType = "text/css";
        AResponseInfo->ResponseNo = 200;
    }
    else if (ARequestInfo->Document == "/jquery.js")
    {
        AResponseInfo->ContentText = pagejQuery->Content;
        AResponseInfo->ContentType = "text/javascript";
        AResponseInfo->ResponseNo = 200;
    }
    else if (ARequestInfo->Document == "/jquery.mobile.js")
    {
        AResponseInfo->ContentText = pagejQueryMobile->Content;
        AResponseInfo->ContentType = "text/javascript";
        AResponseInfo->ResponseNo = 200;
    }
    else
    {
        if (AContext->Connection->Socket->Binding->Port == WSServer->SSLOptions->Port)
            FRequestSSL = true;
        else
            FRequestSSL = false;
        AResponseInfo->ContentText = pageKendoUI_Grid->Content;
        AResponseInfo->ContentType = "text/html";
        AResponseInfo->ResponseNo = 200;
    }
}
```

## Aktualizacje WebSockets

Gdy klient aktualizuje rekord siatki, zmiana ta jest przesyłana do wszystkich połączonych klientów za pomocą protokołu WebSocket. Zdarzenie OnMessage służy do otrzymywania powiadomień o zmianach w siatce. Wiadomości są w formacie JSON, wystarczy więc odczytać tekst JSON, zdekodować go i wysłać odpowiedź do drugiego peera.

```

void OnServerMessage(TsgcWSConnection *Connection, const String Text)
{
    TsgcJSON *oJSON = new TsgcJSON();
    try
    {
        oJSON->Read(Text);
        // ... read
        if (oJSON->Node["type"]->Value == "read")
        {
            TsgcObjectJSON *oArray = oJSON->AddArray("data");
            for (int i = 0; i < 20; i++)
            {
                TsgcJSON *oObj = oArray->JSONObject->AddObject(IntToStr(i))->JSONObject;
                oObj->AddPair("ContactID", i);
                oObj->AddPair("ContactName", ContactName[i]);
                oObj->AddPair("ContactTitle", ContactTitle[i]);
                oObj->AddPair("CompanyName", CompanyName[i]);
                oObj->AddPair("Country", Country[i]);
            }
            Connection->WriteData(oJSON->Text);
        }
        // ... update
        else if (oJSON->Node["type"]->Value == "update")
        {
            WSServer->Broadcast(StringReplace(Text, "\"type\": \"update\"", "\"type\": \"push-update\"", []), "", "",
                Connection->Guid);
            Connection->WriteData(Text);
        }
        // ... destroy
        else if (oJSON->Node["type"]->Value == "destroy")
        {
            WSServer->Broadcast(StringReplace(Text, "\"type\": \"destroy\"", "\"type\": \"push-destroy\"", []), "", "",
                Connection->Guid);
            Connection->WriteData(Text);
        }
        // ... create
        else if (oJSON->Node["type"]->Value == "create")
        {
            String vText = StringReplace(Text, "null", FormatDateTime("yyyymmddhhnnsszzz", Now), []);
            WSServer->Broadcast(StringReplace(vText, "\"type\": \"create\"", "\"type\": \"push-create\"", []), "", "",
                Connection->Guid);
            Connection->WriteData(vText);
        }
    }
    __finally
    {
        oJSON->Free();
    }
}

```

## Dema | ServerSentEvents

To demo pokazuje, jak działają zdarzenia Server-Sent Events na serwerze WebSocket. `sgcWebSockets` umożliwia obsługę przez serwer więcej niż jednego protokołu na tym samym porcie nasłuchiwania.

Więcej informacji na temat [zdarzeń wysyłanych przez serwer](#).

To demo pokazuje, jak serwer będzie co sekundę wysyłać czas do wszystkich połączonych klientów za pomocą Server Sent Events.

Po uruchomieniu serwera **rozwłasza** do wszystkich połączonych klientów wiadomość zawierającą **czas serwera**. Każdorazowo po odebraniu tej wiadomości klient wyświetla ją użytkownikowi.

```
private void Timer1Timer(TObject *sender)
{
    server->Broadcast("data: " + "Server Time: " + FormatDateTime("hh:nn:ss", Now));
}
```

Poniżej przedstawiono **kod JavaScript** obsługujący połączenie WebSocket:

```
socket = new sgcWebSocket('sse', '', 'sse');
socket.on('open', function(evt){
    document.getElementById('status').innerHTML = "Socket Open";
    document.getElementById('status').className = "online";
});
socket.on('close', function(evt){
    document.getElementById('status').innerHTML = "Socket Closed";
    document.getElementById('status').className = "offline";
});
socket.on('message', function(evt){
    document.getElementById('log').innerHTML = evt.message;
});
socket.on('error', function(evt){
    document.getElementById('status').innerHTML = "Socket Error";
    document.getElementById('status').className = "fail";
});
```

# Dema | Serwer WebRTC

Ten przykład pokazuje, jak zbudować serwer wideokonferencji przy użyciu [TsgcWebSocketHTTPServer](#) i WebRTC jako biblioteki JavaScript.

Demo używa protokołu WebSocket do sygnalizacji WebRTC i publicznych serwerów STUN/TURN. W środowiskach produkcyjnych należy korzystać z własnych serwerów STUN/TURN. Zarejestrowani użytkownicy mogą pobrać Coturn dla Windows, który jest już skompilowany i zawiera wszystkie wymagane biblioteki do uruchomienia na własnych serwerach.

Klient musi być przeglądarką internetową obsługującą połączenia WebRTC.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketHTTPServer](#) i zapoznać się z [demonstracją czatu serwera](#).
- Następnie utwórz nową instancję [TsgcWSPServer\\_WebRTC](#).
- Następnie należy przypisać protokół WebRTC do serwera WebSocket oraz skonfigurować hosta i port serwera.

```
WSServer->Port = StrToInt(txtDefaultPort->Text);
// ... bindings
With WSServer->Bindings->Add do
{
    Port = StrToInt(txtDefaultPort->Text);
    IP = txtHost>Text;
}
// ... active
WSServer->Active = true;
```

Demo wymaga strony HTML z indeksem używanej do wyświetlenia strony głównej WebRTC; strona ta jest dostarczana wraz z demo.

## Uruchom w przeglądarce internetowej

Po skonfigurowaniu serwera uruchom go i wybierz jedną z dostępnych przeglądarek internetowych. Otworzy się nowa sesja przeglądarki z prośbą o rozpoczęcie nowej sesji. Po pomyślnym nawiązaniu połączenia będzie widoczny własny obraz wideo, a jeśli ten sam adres URL zostanie otwarty w innej przeglądarce, widoczni będą obaj połączeni uczestnicy.

Demo działa domyślnie bez SSL, co jest prawidłowe wyłącznie dla połączeń localhost. W przypadku witryn produkcyjnych należy używać połączeń SSL. Sprawdź [demo czatu serwera](#), aby skonfigurować SSL po stronie serwera.

# Dema | Serwer AppRTC

Ta demonstracja pokazuje, jak zbudować serwer wideokonferencyjny przy użyciu [TsgcWebSocketHTTPServer](#) i biblioteki JavaScript AppRTC.

Demo używa protokołu WebSocket do sygnalizacji WebRTC i publicznych serwerów STUN/TURN. W środowiskach produkcyjnych należy korzystać z własnych serwerów STUN/TURN. Zarejestrowani użytkownicy mogą pobrać Coturn dla Windows, który jest już skompilowany i zawiera wszystkie wymagane biblioteki do uruchomienia na własnych serwerach.

Klient musi być przeglądarką internetową obsługującą połączenia WebRTC.

## Konfiguracja

- Najpierw należy utworzyć nową instancję [TsgcWebSocketHTTPServer](#) i zapoznać się z [demonstracją czatu serwera](#).
- Następnie należy utworzyć nowe wystąpienie [TsgcWSPServer\\_AppRTC](#).
- Następnie należy przypisać protokół AppRTC do serwera WebSocket i skonfigurować hosta oraz port serwera. WebRTC wymaga bezpiecznych połączeń, dlatego konieczne jest użycie certyfikatu PEM i skonfigurowanie właściwości SSLOptions składnika.

```
WSServer->Port = StrToInt(txtDefaultPort->Text);
// ... bindings
With WSServer->Bindings->Add do
{
    Port = StrToInt(txtDefaultPort->Text);
    IP = txtHost>Text;
}
// ... properties
WSPAppRTC->AppRTC->RoomLink = "https://" + txtHost->Text + ":" + txtDefaultPort->Text + "/r/";
WSPAppRTC->AppRTC->WebSocketURL = "wss://" + txtHost->Text + ":" + txtDefaultPort->Text;
// ... active
WSServer->Active = true;
```

- AppRTC.RoomLink to adres URL, pod który przeglądarka internetowa zostanie przekierowana w celu zalogowania się do pokoju
- AppRTC.WebSocketURL to adres URL połączenia WebSocket.
- Serwery IceServers można skonfigurować w protokole serwera AppRTC.

Demo wymaga strony indeksowej HTML używanej do wyświetlenia strony głównej AppRTC; strona ta jest dołączona do demo.



## Uruchom w przeglądarce internetowej

Po skonfigurowaniu serwera należy go uruchomić i wybrać jedną z dostępnych przeglądarek internetowych. Otworzy ona nową sesję przeglądarki z prośbą o dołączenie do nowego pokoju. Po dołączeniu do pokoju i pomyślnym połączeniu zostanie wyświetlony link, który należy otworzyć w innej przeglądarce, aby rozpocząć nową wideokonferencję.



# Dema | Klient Telegram

Ten przykład pokazuje, jak nawiązać połączenie z Telegram, otrzymywać wszystkie kontakty, wysyłać wiadomości tekstowe, wysyłać obrazy... i wiele więcej

## Konfiguracja

- Najpierw utwórz nową instancję [TsgcTDLib\\_Telegram](#).
- Przed próbą połączenia z Telegram należy przekazać do składnika klienckiego pewne parametry, takie jak API Hash, API Id... Po ustawieniu wszystkich wymaganych parametrów należy ustawić właściwość Active = true, aby nawiązać połączenie.

```
telegram->Telegram->API->ApiHash = txtApiHash->Text;
telegram->Telegram->API->ApiId = txtApiId->Text;
telegram->Telegram->PhoneNumber = "";
telegram->Telegram->BotToken = "";
if (chkLoginBot->Checked)
{
    telegram->Telegram->BotToken = txtBotToken->Text;
}
else
{
    telegram->Telegram->PhoneNumber = txtPhoneNumber->Text;
}
telegram->Active = true;
```

- Gdy klient próbuje połączyć się z Telegram, zazwyczaj wymagany jest kod, dlatego należy obsłużyć zdarzenie **OnTelegramAuthenticationCode** i zwrócić parametr Code z wartością podaną przez konto Telegram.

```
private void OnAuthenticationCodeEvent(TObject *Sender, ref string Code)
{
    Code = InputBox("Telegram", "Introduce Telegram Code");
}
```

## Wysyłanie wiadomości Telegram

Aby wysłać wiadomość Telegram (tekstową, plik, obraz...), zawsze należy najpierw ustawić ChatId, do którego ma zostać wysłana wiadomość, a następnie podać parametr będący treścią tekstową, nazwą pliku itd.

```
// send text message
sgcTelegram->SendTextMessage("456413", "Hello From sgcWebSockets!!!");
```

```
// send file message
sgcTelegram->SendDocumentMessage("383784", "c:\\yourfile.txt");
```

## Odbieranie wiadomości Telegram

Wiadomości odebrane przez klienta Telegram są obsługiwane przez dedykowane procedury obsługi zdarzeń. Istnieje zdarzenie wywoływane po odebraniu nowej wiadomości tekstowej, nowego dokumentu, zdjęcia itd.

```
private void OnMessageTextEvent(TObject *Sender, TsgcTelegramMessageText *MessageText)
{
    DoLogMessage(MessageText->ChatId, MessageText->SenderId->ToString(), MessageText->Text);
}
```

```
private void OnMessageDocumentEvent(TsgcTDLib_Telegram *Sender, TsgcTelegramMessageDocument *MessageDocument)
{
    DoLogMessage(MessageDocument->ChatId, MessageDocument->SenderId->ToString(), MessageDocument->FileName);
}
```

# Coturn

Coturn

**Począwszy od wersji sgcWebSockets 4.5.2 ENTERPRISE Edition, możliwe jest zbudowanie własnego serwera STUN/TURN przy użyciu Delphi/CBuilder.**

To darmowa, otwarta implementacja serwerów TURN i STUN.

Serwer TURN jest serwerem i bramą do przekierowania ruchu VoIP przez NAT. Może być również używany jako ogólny serwer i brama do przekierowania dowolnego ruchu sieciowego.

Obsługiwane platformy docelowe projektu to:

- **Linux**
- **Mac OS X**
- **Windows** (Cygwin): skompilowane pliki binarne są dostępne dla zarejestrowanych użytkowników.

## Konfiguracja systemu Windows

Najpierw należy pobrać skompilowane pliki binarne ze swojego konta. Dostępne są 2 wersje: win32 i win64. Wybierz żadaną platformę i rozpakuj pliki binarne do folderu. Zostaną utworzone następujące pliki:

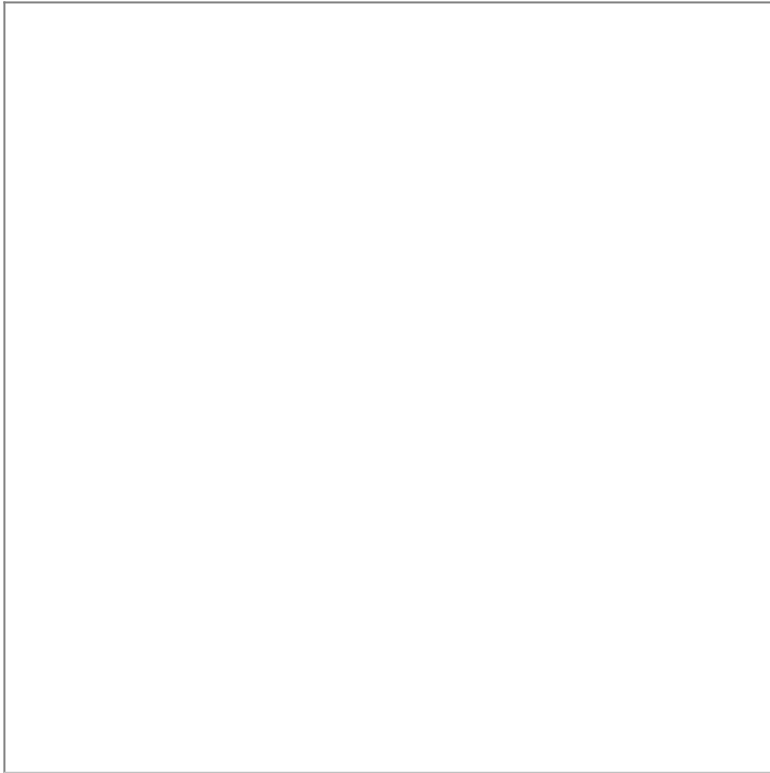
**1.** Niektóre biblioteki cygwin są wymagane do uruchomienia aplikacji; należy wdrożyć te biblioteki razem z serwerem coturn.

**2.** Niektóre aplikacje konsolowe:

2.1 turnserver.exe: jest główną aplikacją konsolową służącą do uruchamiania serwera TURN/STUN

2.2 Inne aplikacje: służą do celów konfiguracyjnych lub testowych.

**3.** Turnserver.conf: plik konfiguracyjny serwera coturn.



### **turnserver.conf**

To jest plik konfiguracyjny serwera coturn; po jego otwarciu widoczna jest domyślna konfiguracja.

### **Prosta konfiguracja**

Serwer ma następujący publiczny adres IP 80.15.44.123 i nasłuchuje na porcie 80. Dane uwierzytelniające do połączenia: nazwa użytkownika = demo, hasło = secret

Ustaw następującą konfigurację:

```
listening-ip=80.15.44.123
```

```
listening-port=80
```

```
realm=yourrealm.com
```

```
user=demo:secret
```

### **Konfiguracja z włączonym TLS**

Serwer ma publiczny adres IP 80.15.44.123 i nasłuchuje na portach 80 i 443 (połączenia TLS). Dane uwierzytelniające do połączenia: nazwa użytkownika = demo, hasło = secret. Nazwa certyfikatu (musi być w formacie PEM) to certificate.crt, a klucz prywatny to private.key.

Ustaw następującą konfigurację:

```
listening-ip=80.15.44.123
```

```
listening-port=80
```

```
realm=yourrealm.com
```

```
tls-listening-port=443
```

```
cert=certificate.crt
```

```
pkey=private.key
```

```
user=demo:secret
```

Dostępne są dodatkowe konfiguracje — wystarczy otworzyć plik turnserver.conf i zapoznać się z udokumentowanymi sekcjami.

## **Uruchom coturn**

Po skonfigurowaniu można uruchomić serwer, wykonując plik turnserver.exe. Zostanie otwarta nowa aplikacja konsolowa i utworzony plik dziennika. Można zwiększyć poziom szczegółowości aplikacji konsolowej (uzyskać bardziej szczegółowe komunikaty), włączając opcję "verbose" w pliku turnserver.conf.



# Licencja

## Umowa licencyjna użytkownika końcowego komponentów eSeGeCe

Umowa licencyjna użytkownika końcowego („EULA”) komponentów eSeGeCe („eSeGeCe”) jest prawną umową między użytkownikiem (osobą fizyczną lub podmiotem) a Autorem eSeGeCe, dotyczącą wszystkich komponentów eSeGeCe, które mogą obejmować powiązane komponenty oprogramowania, nośniki, materiały drukowane oraz dokumentację „online” lub elektroniczną („komponenty eSeGeCe”). Instalując, kopiując lub w inny sposób używając komponentów eSeGeCe, użytkownik zgadza się być związany warunkami niniejszej EULA. Niniejsza umowa licencyjna stanowi całość porozumienia dotyczącego programu między użytkownikiem a Autorem eSeGeCe (zwanym „LICENCJODAWCĄ”) i zastępuje wszelkie wcześniejsze propozycje, oświadczenia lub porozumienia między stronami. Jeśli użytkownik nie zgadza się z warunkami niniejszej EULA, nie powinien instalować ani używać komponentów eSeGeCe.

Komponenty eSeGeCe są chronione prawem autorskim i międzynarodowymi traktatami dotyczącymi praw autorskich, a także innymi prawami i traktatami dotyczącymi własności intelektualnej. Komponenty eSeGeCe są licencjonowane, a nie sprzedawane.

Jeśli potrzebny jest KOD ŹRÓDŁOWY, należy uiścić opłatę rejestracyjną. Nie wolno przekazywać kluczy licencyjnych ani pełnych edycji eSeGeCe (w tym edycji DCU oraz edycji z kodem źródłowym) żadnym osobom trzecim ani podmiotom. Nie wolno również korzystać z kluczy licencyjnych ani pełnych edycji eSeGeCe uzyskanych od osób trzecich lub podmiotów zewnętrznych.

### 1. UDZIELENIE LICENCJI

Komponenty eSeGeCe są licencjonowane w następujący sposób:

#### (a) Instalacja i użytkowanie.

LICENSER przyznaje prawo do instalowania i używania kopii komponentów eSeGeCe na komputerze z ważnie licencjonowaną kopią systemu operacyjnego, dla którego komponenty eSeGeCe zostały zaprojektowane [np. Windows 2000, Windows 2003, Windows XP, Windows ME, Windows Vista, Windows 7, Windows 8, Windows 10].

#### (b) Bezpłatna licencja.

Można tworzyć komercyjne aplikacje oparte na komponentach eSeGeCe i dystrybuować je wraz z plikami wykonywalnymi bez konieczności uiszczania tantiem.

#### (c) Modyfikacje (tylko edycje ze źródłami).

Dopuszczalne jest wprowadzanie modyfikacji, ulepszeń, dzieł pochodnych i/lub rozszerzeń licencjonowanego KODU ŹRÓDŁOWEGO dostarczonego na warunkach określonych w niniejszej umowie licencyjnej.

#### (d) Kopie zapasowe.

Można również wykonywać kopie komponentów eSeGeCe w zakresie niezbędnym do celów tworzenia kopii zapasowych i archiwalnych.

### 2. OPIS INNYCH PRAW I OGRANICZEŃ

#### (a) Zachowanie informacji o prawach autorskich.

Nie wolno usuwać ani zmieniać żadnych informacji o prawach autorskich na żadnych kopiach komponentów eSeGeCe.

(b) Dystrybucja.

Zarejestrowanych kopii komponentów eSeGeCe nie wolno rozprowadzać wśród osób trzecich. Edycje ewaluacyjne dostępne do pobrania z oficjalnych stron internetowych eSeGeCe mogą być swobodnie rozprowadzane.

Możliwe jest tworzenie komponentów/kontrolki ActiveX/bibliotek zawierających komponenty eSeGeCe dla własnych aplikacji, jednak NIE WOLNO ich rozpowszechniać ani udostępniać osobom trzecim.

(c) Zakaz dystrybucji KODU ŹRÓDŁOWEGO (dotyczy wyłącznie edycji Source).

NIE wolno rozpowszechniać ani publikować KODU ŹRÓDŁOWEGO, żadnych modyfikacji, rozszerzeń, dzieł pochodnych ani rozszerzeń w postaci KODU ŹRÓDŁOWEGO stronom trzecim.

KOD ŹRÓDŁOWY nie może być w żadnej części rozpowszechniany, publikowany, ujawniany ani w inny sposób udostępniany osobom trzecim.

(d) Zakaz inżynierii wstecznej, dekompilacji i dezasemblacji.

Nie wolno dokonywać inżynierii wstecznej, dekompilacji ani dezasemblacji komponentów eSeGeCe, z wyjątkiem sytuacji i wyłącznie w zakresie, w jakim takie działanie jest wyraźnie dozwolone przez obowiązujące prawo niezależnie od niniejszego ograniczenia.

(e) Wynajem.

Nie wolno wynajmować, dzierżawić ani pożyczać komponentów eSeGeCe.

(f) Usługi wsparcia.

LICENSER może świadczyć usługi wsparcia związane z komponentami eSeGeCe ("Usługi wsparcia"). Wszelki dodatkowy kod oprogramowania dostarczony w ramach Usług wsparcia jest uważany za część komponentów eSeGeCe i podlega warunkom niniejszej EULA.

eSeGeCe jest licencjonowane do użytku przez tylko jednego dewelopera jednocześnie. Wsparcie techniczne jest świadczone wyłącznie jednemu konkretnemu deweloperowi.

(g) Zgodność z obowiązującymi przepisami prawa.

Należy przestrzegać wszystkich obowiązujących przepisów prawa dotyczących korzystania z komponentów eSeGeCe.

### 3. ROZWIĄZANIE UMOWY

Bez uszczerbku dla jakichkolwiek innych praw LICENCJODAWCA może wypowiedzieć niniejszą EULA, jeśli nie zostaną spełnione jej warunki. W takim przypadku należy zniszczyć wszystkie kopie komponentów eSeGeCe będące w posiadaniu użytkownika.

### 4. PRAWA AUTORSKIE

Wszelkie tytuły, w tym między innymi prawa autorskie, do składników eSeGeCe i ich kopii należą do LICENCJODAWCY lub jego dostawców. Wszelkie tytuły i prawa własności intelektualnej do treści dostępnych za pośrednictwem składników eSeGeCe stanowią własność odpowiednich właścicieli treści i mogą być chronione obowiązującymi przepisami o prawie autorskim lub innymi przepisami i umowami doty-



czącymi własności intelektualnej. Niniejsza umowa EULA nie przyznaje żadnych praw do korzystania z takich treści. Wszelkie prawa nieprzyznanej wyraźnie są zastrzeżone przez LICENCJODAWCĘ.

## 5. BRAK GWARANCJI

LICENSER wyraźnie zrzeka się wszelkich gwarancji dotyczących komponentów eSeGeCe. Komponenty eSeGeCe są dostarczane „w stanie takim, w jakim są” bez żadnej wyraźnej ani dorozumianej gwarancji jakiegokolwiek rodzaju, w tym między innymi wszelkich gwarancji przydatności handlowej, nienaruszalności praw lub przydatności do określonego celu. LICENSER nie gwarantuje ani nie ponosi odpowiedzialności za dokładność lub kompletność jakichkolwiek informacji, tekstu, grafiki, linków lub innych elementów zawartych w komponentach eSeGeCe. LICENSER nie udziela żadnych gwarancji w odniesieniu do szkód, które mogą wynikać z transmisji wirusa komputerowego, robaka, bomby zegarowej, bomby logicznej lub innego podobnego programu. LICENSER wyraźnie zrzeka się ponadto wszelkich gwarancji lub oświadczeń wobec autoryzowanych użytkowników lub wobec osób trzecich.

## 6. OGRANICZENIE ODPOWIEDZIALNOŚCI

W żadnym wypadku LICENCJODAWCA nie ponosi odpowiedzialności za żadne szkody (w tym między innymi utracone zyski, przerwy w działalności lub utratę informacji) wynikające z używania lub niemożności używania komponentów eSeGeCe przez „Autoryzowanych Użytkowników”, nawet jeśli LICENCJODAWCA został poinformowany o możliwości wystąpienia takich szkód. W żadnym wypadku LICENCJODAWCA nie będzie ponosić odpowiedzialności za utratę danych ani za szkody pośrednie, szczególne, przypadkowe, wynikowe (w tym utracone zyski) ani inne szkody wynikające z umowy, czynu niedozwolonego lub innych podstaw prawnych. LICENCJODAWCA nie ponosi żadnej odpowiedzialności w związku z treścią komponentów eSeGeCe ani żadną ich częścią, w tym między innymi za błędy lub pominięcia w nich zawarte, zniesławienie, naruszenie praw do wizerunku publicznego, prywatności, praw do znaków towarowych, przerwę w działalności, szkodę osobową oraz utratę prywatności, praw moralnych lub ujawnienie informacji poufnych.

# Indeks

AbstractApi Geolocation [1281](#)

Add Telegram Proxy [777](#)

alokacja adresu IP klienta TURN [1183](#)

Alokacje serwera TURN [1190](#)

ALPN [133](#)

Amazon SQS [1115](#)

AMQP Odbieranie wiadomości [369](#)

AMQP1 — wysyłanie wiadomości [412](#)

AMQP1 Łączy [404](#)

AMQP1 Read Message [414](#)

AMQP1 Sender Links [406](#)

Anthropic [883](#)

Anthropic Citations [903](#)

Anthropic Extended Thinking [897](#)

Anthropic Prompt Caching [901](#)

Anthropic Structured Outputs [906](#)

Anthropic Tool Use [893](#)

Anthropic Vision [891](#)

API [489](#), [515](#), [517](#), [620](#), [674](#), [701](#), [712](#), [1280](#)

API 3Commas [654](#)

API Binance [491](#), [508](#), [515](#)

API Binance Futures [508](#), [515](#)

API Bitfinex [620](#)

API Bitget [716](#)

API Bitmex [610](#)

API Blockchain [684](#)

API Bybit [669](#)

API Cex [686](#)

API Cex Plus [696](#)

API Coinbase [521](#)

API Coinbase Pro [521](#)

API Crypto.com [725](#)

API Deribit [722](#)

API Discord [701](#)

API Forex [674](#)

API GateIO [719](#)

API HTX [728](#)

API Kraken [546](#), [548](#), [556](#), [560](#), [562](#), [566](#), [568](#), [581](#), [590](#), [592](#)

API Kraken Futures [566](#), [568](#), [581](#), [590](#), [592](#)

API Kraken Futures REST Private [592](#)

API Kraken Futures REST Public [590](#)

API Kraken REST Private [562](#)

API Kraken REST Public [560](#)

API Kucoin [626](#)

API Kucoin Futures [640](#)

API MEXC [708](#)

API MEXC Futures [712](#)

API OKX [659](#)

API OpenAI [706](#)

API Pusher [599](#)

API SignalR [542](#)

API SignalRCore [534](#)

API SocketIO [517](#)

API Telegram [756](#)

API Whatsapp [731](#)

API XTB [665](#)

APIs [491](#), [500](#), [503](#), [508](#), [521](#), [527](#), [530](#), [534](#), [542](#), [546](#), [548](#), [556](#), [560](#), [562](#), [566](#), [568](#), [581](#), [590](#), [592](#), [599](#), [610](#), [681](#), [684](#), [686](#), [708](#), [756](#)

Aplikacje OpenAI [855](#)

APNs [999](#), [1001](#), [1003](#)

    Certificate-Based Connection [1003](#)

    Połączenie oparte na tokenach [1001](#)

Apple Push Notifications [997](#)

Authentication [207](#)

Authorization Code Grant [1049](#)

Authorization Code Grant (RFC 6749) [1022](#)

automatyczny zakaz [263](#)

Autoryzacja WebAuthn [1106](#)

Bezpieczeństwo WhatsApp [741](#)

Bezpieczne połączenia [108](#)

Biała lista zapory sieciowej [261](#)

Biblioteka OpenAPI [1223](#)

Binance — pobieranie danych rynkowych [502](#)

Binance — wypłata [507](#)

Binance Connect [500](#)

Binance Subscribe [501](#)

Binance Trade Spot [504](#)

Binary Message [179](#)

Bitmex Connect WebSocket API [615](#)

Bitmex Subskrybuj kanał WebSocket [616](#)

Błąd [226](#)

Błędy instalacji [38](#)

Bot [772](#)

Bot Detection [271](#)

Broadcast [118](#)

Brute Force [263](#)

Build [63](#), [64](#)

Build Android Application [63](#)

Build iOS Application [64](#)

Certificate-Based Connection [1003](#)

    APNs [1003](#)

Certyfikaty OpenSSL [173](#)

Certyfikaty P12 OpenSSL [90](#)

Certyfikaty SChannel [174](#)

Certyfikaty z podpisem własnym [243](#)

Channels [118](#), [501](#), [528](#), [682](#)

Chatbot Embeddings [879](#)

CIDR [261](#)

Client [179](#), [184](#), [187](#), [378](#), [379](#), [1174](#), [1175](#), [1185](#), [1285](#), [1306](#)

Client Chat [1285](#)

Client Credentials Grant (RFC 6749) [1026](#)

Client Exceptions [184](#)

Client Kafka Connect [378](#)

Client Kafka Disconnect [379](#)

Client Proxies [187](#)

Client Send Text [179](#)

Clients [378](#), [379](#), [1285](#)

Coinbase Pro — dane rynkowe [529](#)

Coinbase Pro Connect [527](#)

Coinbase Pro Place Orders [532](#)

Coinbase Pro Private REST API [530](#)

Coinbase Pro Subscribe [528](#)

CommitOffset [384](#)

CommitSync [382](#)

Consume [381](#)

Consumer [381](#)

Consumer Group [382](#)

Coturn [1308](#)

Crawler Detection [271](#)

CreateTopic [383](#)

Cross-Site Scripting [265](#)

CryptoHopper [781](#)

Czarna lista zapory sieciowej [261](#)

Czas prywatnych żądań Coinbase Pro [531](#)

Czas prywatnych żądań Kucoin [639](#)

Czas żądań prywatnych Binance [506](#)

Czas żądań prywatnych Kucoin Futures [653](#)

Czat serwera [1282](#)

Czat Telegram [772](#)

Dane rynku Forex [680](#)

Datacenter Detection [271](#)

Datasnap [1209](#)

DeepSeek [916](#)

DeepSeek Vision [920](#)

Deflate-Frame [795](#)

DeleteTopic [383](#)

Device Authorization Grant (RFC 8628) [1030](#), [1057](#)

DNSBL [271](#)

Dodatkowe właściwości OpenAPI [1219](#)

Dokumenty Anthropic [899](#)

Domyślny JavaScript protokołu [451](#)

Dostawca OAuth2 Azure AD [1067](#)

Dostrajanie OpenAI [852](#)

DPoP — Demonstrating Proof of Possession (RFC 9449) [1032](#)

Edycje [19](#)

Elicitation klienta MCP [831](#)

Embeddingi Tworzenie wektorów [878](#)

EPOLL [131](#)

FCrDNS [271](#)

Files [120](#), [218](#)

Filtrowanie IP [261](#)

Fired [250](#)

Flash [123](#)

Flow [58](#)

Threading [58](#)

Forex Account [679](#)

Forex Connect [676](#), [681](#)

Forex Handel [677](#)

Forex Subscribe [682](#)

Fragmentowane wiadomości [156](#)

Gemini [912](#)

Gemini — strukturyzowane dane wyjściowe [929](#)

Gemini CountTokens [931](#)

Gemini EmbedContent [932](#)

Gemini Embeddings [932](#)

Gemini Function Calling [933](#)

Gemini Tool Use [933](#)

Gemini Vision [927](#)

Generate [998](#)

Remote Notification APNs [998](#)

GetCommittedOffset [384](#)

GetEarliestOffset [384](#)

GetLatestOffset [384](#)

GetMetadata [383](#)

Google Calendar [1134](#), [1141](#), [1142](#), [1143](#)

Google Calendar RefreshToken [1143](#)

Google Calendar Sync Events [1142](#)

Google Calendar Synchronizacja kalendarzy [1141](#)

Google Cloud FCM [1145](#)

Google Cloud Pub [1125](#)

Google Service Accounts [1123](#)

Grok [941](#)

Grok Vision [945](#)

GroupId [382](#)

Grupy [127](#)

Handel API Binance Futures [515](#)

Handel kontraktami Kucoin Futures [650](#)

HeartBeat [111](#)

HighPerfAcceptsPerWorker [238](#)

History [679](#)

HTML [1042](#)

HTTP [56](#), [117](#), [135](#), [217](#), [218](#), [226](#), [227](#), [244](#), [975](#), [981](#)

HTTP Dispatch Files [218](#)

HTTP Server Stream Video [229](#)

HTTP/2 [219](#), [220](#), [222](#), [982](#), [983](#), [984](#), [985](#), [988](#), [992](#)

HTTP/2 Alternate Service [222](#)

HTTP/2 Częściowe odpowiedzi [984](#)

HTTP/2 Nagłówki [985](#)

HTTP/2 Pobieranie pliku [983](#)

HTTP/2 Server Push [220](#), [982](#)

HTTP1 [1005](#)

HTTP2 [977](#)

HTTPAPI [241](#), [244](#), [245](#), [250](#)

HTTPAPI — wysyłanie odpowiedzi tekstowej [246](#)

HTTPAPI Disable HTTP [244](#)

HTTPAPI FineTune [238](#)

HTTPAPI Nagłówki niestandardowe [245](#)

HTTPAPI OnDisconnect [250](#)

HTTPAPI Wysyłanie odpowiedzi z plikiem [248](#)

ICE [1192](#)

ICE — zbieranie kandydatów [1195](#)

Identyfikator numeru telefonu WhatsApp [738](#)

Indy [95](#)

Indy SChannel [199](#), [230](#)

Instalacja [21](#)

Instalacja pakietu [35](#)

Instalacja pakietu sgcIndy [46](#)

Install Setup [27](#)

Interfejsy API [654](#)

IOCP [129](#)

IoT [954](#), [955](#), [967](#)

IoT Amazon MQTT Client [955](#)

Jak złożyć zlecenie w Bitmex [617](#)

JWT [1074](#)

Kafka [378](#), [379](#), [380](#), [381](#), [382](#), [383](#), [384](#)

Kafka Consume Messages [381](#)

Kafka Consumer Groups [382](#)

Kafka Offsets [384](#)

Kafka Produce Messages [380](#)

Kanały [359](#), [1186](#)

kanały AMQP [359](#)

Katalogi główne serwera MCP [813](#)

KendoUI\_Grid [1300](#)

klienci [1289](#)

Klienci [339](#), [356](#), [473](#), [987](#), [989](#)

    Wysyłanie plików [473](#)

klient [990](#), [1186](#), [1289](#)

Klient [161](#), [164](#), [165](#), [177](#), [178](#), [182](#), [185](#), [186](#), [214](#), [336](#), [339](#), [340](#), [356](#), [358](#), [473](#), [986](#), [987](#), [989](#), [1173](#), [1183](#), [1184](#), [1287](#), [1291](#), [1296](#)

Klient — utrzymywanie otwartego połączenia [165](#)

Klient AMQP1 — limit czasu bezczynności połączenia [394](#)

Klient AMQP1 Azure MessageBus [398](#)

Klient IoT Azure MQTT [967](#)

Klient Javascript WebAuthn [1109](#)

Klient MQTT Połączenie [336](#)

Klient MQTT Wersja [340](#)

Klient OAuth2 dla aplikacji desktopowych [1019](#)

Klient OAuth2 dla aplikacji internetowych [1018](#)

Klient SocketIO [1291](#)

Klient utrzymuje aktywne połączenie [987](#)

Klient wysyła binarną wiadomość [178](#)

Klient Wysyłanie tekstu [177](#)

Klient Wysyłanie wiadomości tekstowej [177](#)

klucze Google OAuth2 [1121](#)

Kod autoryzacji z PKCE (RFC 7636) [1024](#)

Kod OAuth2 Microsoft OpenAPI [1276](#)

Kolejki [140](#), [363](#)

kolejki AMQP [363](#)

Kompilacja [60](#), [61](#)

Kompilacja aplikacji OSX [61](#)

Kompresja [122](#)

Konfiguracja instalacji [43](#)

Konfiguruj ZLib [49](#)

Konto piaskownicy Coinbase Pro [533](#)

Konto usługi Google Calendar [1144](#)

Korzenie klienta MCP [827](#)

Korzystanie z DLL [101](#)

Kucoin — pobieranie danych rynkowych [635](#)

Kucoin — połączenie z WebSocket API [633](#)

Kucoin — subskrypcja kanału WebSocket [634](#)

Kucoin Futures — pobieranie danych rynkowych [648](#)

Kucoin Futures — prywatne REST API [649](#)

Kucoin Futures — subskrypcja kanału WebSocket [647](#)

Kucoin Futures Połącz WebSocket API [646](#)

Kucoin Private REST API [636](#)

Kucoin Trade Spot [637](#)

Łączą odbiorcy AMQP1 [409](#)

łączenie z serwerem WebSocket [160](#)

Łącznik Anthropic MCP [910](#)

Ładowanie dodatkowych funkcji OpenSSL [93](#)

Licencja [1311](#)

Lightstreamer [681](#), [682](#)

LoadBalancing [153](#)

Logi [115](#)

MaxConnectionsPerIP [268](#)

MaxMessagesPerSec [268](#)

MCP [796](#)

MCP Server Elicitation [817](#)

Menedżer pamięci [71](#)

Message Filtering [265](#)

metoda [981](#)

migawki klienta [1296](#)

migawki serwera [1295](#)

Mistral AI [947](#)

Mistral Models [952](#)

Mistral Vision [951](#)

Modele Anthropic [895](#)

Modele DeepSeek [921](#)

Modele Gemini [928](#)

Modele Grok [946](#)

Modele Ollama [939](#)

Moderacja OpenAI [848](#)

Monitor serwera [1292](#)

Monity klienta MCP [823](#)

Monity serwera MCP [805](#)

MQTT [336](#), [338](#), [339](#), [340](#), [341](#), [342](#), [344](#), [345](#), [346](#), [348](#), [349](#), [955](#), [967](#), [1289](#)

MQTT Czyszczenie zachowanych wiadomości [349](#)

MQTT Publish [341](#), [348](#)

MQTT Publish Subscribe [341](#)

MQTT Receive Messages [346](#)

Narzędzia klienta MCP [821](#)

Narzędzia serwera MCP [801](#)

Nawigacja WebView2 [1152](#)

Niestandardowe subprotokołu [104](#)

Notification Requests [999](#)

    Sending [999](#)

OAuth2 [992](#), [1011](#), [1040](#), [1042](#), [1043](#), [1044](#), [1045](#), [1046](#), [1048](#), [1121](#)

OAuth2 Dostosuj logowanie [1042](#)

OAuth2 None Authenticate URLs [1048](#)

OAuth2 Register Apps [1044](#)

Obiekty niestandardowe [125](#)

Ochrona przed flooding [268](#)

Oczekujące żądania klienta [989](#)

Odbieranie wiadomości AMQP [371](#)

Odbieranie wiadomości binarnych [181](#), [213](#)

Odbieranie wiadomości tekstowych [180](#), [212](#)

Odpowiedź na rejestrację WebAuthn [1093](#)

Odpowiedź uwierzytelniania WebAuthn [1101](#)

Odświeżanie tokena (Refresh Token Grant) [1055](#)

Odtwarzanie tokenów dostępu OAuth2 [1045](#)

Ograniczanie przepustowości [268](#)

Ollama [935](#)

Ollama Embeddings [940](#)

ompClassic [238](#)

ompHighPerf [238](#)

OpenAI [833](#)

OpenAI Audio [847](#), [866](#)

OpenAI Batch [853](#)

[OpenAI Chat](#) [845](#)

[OpenAI Edit](#) [846](#)

[OpenAI RealTime](#) [849](#)

[OpenAI Responses](#) [850](#)

[OpenAI Speech](#) [851](#)

[OpenAI Uploads](#) [854](#)

[OpenAPI](#) [1214](#)

[OpenAPI Amazon AWS](#) [1240](#)

[OpenAPI Amazon AWS S3](#) [1250](#)

[OpenAPI API](#) [1226](#)

[OpenAPI Client](#) [1227](#)

[OpenAPI Command Line](#) [1221](#)

[OpenAPI Google Cloud](#) [1251](#)

[OpenAPI Google Cloud Calendar](#) [1263](#)

[OpenAPI Google Cloud OAuth2](#) [1257](#)

[OpenAPI Google Cloud PubSub](#) [1262](#)

[OpenAPI Google Drive](#) [1264](#)

[OpenAPI konta usług Google Cloud](#) [1259](#)

[OpenAPI Microsoft](#) [1266](#)

[OpenAPI Microsoft — rejestracja aplikacji](#) [1274](#)

[OpenAPI Microsoft Graph](#) [1279](#)

[OpenAPI Microsoft OAuth2 Credentials](#) [1278](#)

[OpenAPI Microsoft Tenant](#) [1273](#)

[OpenAPI Parser Pascal](#) [1216](#)

[OpenSSL](#) [74](#), [78](#), [81](#), [83](#), [85](#), [173](#)

[OpenSSL Android](#) [83](#)

[OpenSSL iOS](#) [85](#)

[OpenSSL OSX](#) [81](#)

[OpenSSL Own CA Certificates](#) [88](#)

[OpenSSL Weryfikacja certyfikatu](#) [92](#)

[OpenSSL Windows](#) [78](#)

[OperatingMode](#) [238](#)

[Orders](#) [679](#)

[Osadzenia Mistral](#) [953](#)

[Otwarte połączenie klienta](#) [161](#)

[Para kandydatów ICE](#) [1196](#)

[Parametry połączenia WebSocket](#) [100](#)

[Partie Anthropic](#) [896](#)

[PDF-Back-Cover](#) [1327](#)

[PDF-Front-Cover](#) [1](#)

[PerMessage-Deflate](#) [794](#)

[Pinecone](#) [880](#)

[Pliki](#) [154](#), [472](#), [473](#), [474](#), [983](#), [1297](#)

[Pliki Anthropic](#) [908](#)

[Pliki cookie WebView2](#) [1158](#)

[Pliki protokołu](#) [468](#)

[Pobieranie plików WebView2](#) [1160](#)

[Połącz bezpieczny serwer](#) [172](#)

[Połącz serwer TCP](#) [169](#)

[Połącz z Mosquitto](#) [338](#)

[Połączenia klientów](#) [206](#)

[Połączenia TIME\\_WAIT](#) [170](#)

[Połączenie klienta AMQP](#) [356](#)

[Połączenie klienta AMQP1](#) [391](#)

[Połączenie oparte na tokenach](#) [1001](#)

[APNs](#) [1001](#)

[Poll](#) [381](#)

[Porzucone rozłączenia](#) [167](#)



Positions [679](#)

Post Big Files [120](#)

Poświadczenia Amazon AWS dla sgcOpenAPI [1247](#)

Powiązania [119](#), [193](#)

Powiązania serwera [193](#)

Próbkowanie klienta MCP [829](#)

Próbkowanie serwera MCP [815](#)

Produce [380](#)

Producer [380](#)

Protocol AMQP [350](#)

Protocol Dataset [456](#), [461](#), [464](#), [466](#)

Protocol Dataset Javascript [461](#)

Protocol Dataset Notify Updates [466](#)

Protocol Kafka [375](#)

Protocol MQTT [332](#)

Protocol Presence [475](#)

Protocol STOMP [415](#)

Protocol WAMP2 [441](#)

Protocol WebRTC [423](#), [425](#)

Protocol WebRTC Javascript [425](#)

Protocols [332](#), [350](#), [375](#), [415](#), [423](#), [425](#), [441](#), [456](#), [461](#), [466](#), [475](#)

Protokół AMQP1 [385](#)

Protokół AppRTC [420](#)

Protokół domyślny [446](#), [451](#)

Protokół E2EE [484](#)

Protokół Presence [480](#)

Protokół Presence Javascript [480](#)

Protokół rejestracji klienta [186](#)

Protokół WAMP [427](#), [431](#)

Protokół WAMP Javascript [431](#)

Protokoły [104](#), [186](#), [325](#), [328](#), [420](#), [427](#), [431](#), [446](#), [451](#), [464](#), [468](#), [480](#)

Protokoły Javascript [328](#)

Proxy [155](#), [187](#), [777](#)

Prywatne punkty końcowe dostawcy OAuth2 [1069](#)

Prywatne REST API Binance [503](#)

Przegląd [50](#)

Przekazywanie żądań HTTP [135](#)

Przekierowania WebSocket [171](#)

przesyłanie pliku [1297](#)

Przyczyna rozłączenia HTTP/2 [988](#)

Przykład serwera [1040](#)

Przyznanie poświadczeń klienta (Client Credentials Grant) [1051](#)

Publikowanie MQTT [345](#)

Publikowanie wiadomości AMQP [368](#)

Publikowanie wiadomości MQTT [345](#)

QoS AMQP [372](#)

Quality [137](#)

Service [137](#)

Quality Of Service [137](#)

QueueLength [238](#)

QuickStart HTTP [56](#)

QuickStart WebSockets [53](#)

RCON [779](#)

Rejestracja [778](#)

rejestracja użytkownika Telegram [778](#)

Rejestracja WebAuthn [1087](#)

Remote Notification APNs [998](#)

Generate [998](#)

Replikacja tabeli Protocol Dataset [464](#)

Resource Owner Password Credentials Grant [1053](#)

Resource Owner Password Credentials Grant (RFC 6749) [1028](#)

REST API [676](#)

Retransmisje UDP klienta STUN [1173](#)

Reverse DNS [271](#)

Rezerwacja adresu URL HTTPAPI [236](#)

rozłączenie AMQP klienta [358](#)

Rozłączenie klienta AMQP1 [393](#)

Rozszerzenia [793](#)

RTCMultiConnection [787](#)

RTCPeerConnection Data [1208](#)

RTCPeerConnection DTLS [1206](#)

RTCPeerConnection ICE [1204](#)

RTCPeerConnection Klient WebSocket [1201](#)

RTCPeerConnection STUN TURN [1202](#)

SChannel [199](#), [230](#)

SChannel — pobieranie informacji o połączeniu [176](#)

SChannel Server [199](#), [230](#)

Sending [999](#)

Notification Requests [999](#)

Server [194](#), [204](#), [207](#), [217](#), [982](#), [1043](#), [1046](#), [1178](#), [1304](#)

Server AppRTC [1304](#)

Server Authentication [207](#), [1046](#)

Server Endpoints [1043](#)

Server Plain TCP [204](#)

Server Requests [217](#)

Server SSL SChannel [199](#), [230](#)

Server Startup Shutdown [194](#)

Server-Sent Events [149](#)

ServerSentEvents [1302](#)

Service [137](#), [222](#), [1123](#)

Quality [137](#)

serwer [212](#), [213](#)

Serwer [192](#), [193](#), [195](#), [196](#), [203](#), [205](#), [209](#), [210](#), [214](#), [219](#), [227](#), [241](#), [338](#), [472](#), [1040](#), [1177](#), [1189](#), [1190](#), [1282](#), [1292](#), [1295](#), [1299](#), [1303](#)

Wysyłanie plików [472](#)

Serwer o wysokiej wydajności [66](#)

Serwer Odczyt nagłówków [214](#)

Serwer OpenAPI [1233](#)

Serwer OpenAPI Code-First [1238](#)

Serwer OpenAPI w podejściu Spec-First [1236](#)

Serwer SSL [196](#)

Serwer WebSocket RTCPeerConnection [1200](#)

Serwer wysyła wiadomość binarną [210](#)

Serwer wysyła wiadomość tekstową [209](#)

Sesje AMQP1 [401](#)

Sesje klienta MQTT [339](#)

Sesje serwera [227](#)

Sesje serwera MCP [800](#)

SkipIOCPOnSuccess [238](#)

Słupki cenowe [680](#)

SQL Injection [265](#)

SSL serwera [241](#)

Stan połączenia klienta AMQP1 [395](#)

Stop Limit [677](#)

STUN [1170](#), [1173](#), [1174](#), [1175](#), [1177](#), [1178](#)

STUN Client Attributes [1175](#)

STUN Client Long Term Credentials [1174](#)

STUN Server Alternate Server [1178](#)

STUN Server Long Term Credentials [1177](#)

Sub [1125](#)

SubProtocol [146](#)

Subscribe [381](#)

Subskrybenci WAMP [435](#)

Subskrypcja MQTT [344](#)

Sygnalizacja RTCPeerConnection [1203](#)

TCP Connections [144](#)

Telegram — pobieranie członków SuperGroup [775](#)

Telegram Client [1306](#)

Telegram Sponsored Messages [773](#)

Tematy Kafka [383](#)

Tematy MQTT [342](#)

Test przeglądarki internetowej [103](#)

Threading [58](#)

    Flow [58](#)

Throttle [148](#)

Ticki cenowe [680](#)

Token Introspection (RFC 7662) [1061](#)

Token WhatsApp [739](#)

Transakcje [142](#)

Transakcje AMQP [373](#)

Treść odpowiedzi [226](#)

TsgcAIChat [872](#)

TsgcAIDatabaseVectorFile [876](#)

TsgcAIDatabaseVectorPinecone [877](#)

TsgcAIOpenAIAssistant [857](#)

TsgcAIOpenAIAssistant Function Calling [864](#)

TsgcAIOpenAIAssistant Streaming [862](#)

TsgcAIOpenAIAssistant Wyszukiwanie plików [859](#)

TsgcAIOpenAIChatBot [873](#)

TsgcAIOpenAIEmbeddings [875](#)

TsgcAIOpenAITranslator [874](#)

TsgcAudioPlayerMCI [868](#)

TsgcAudioRecorderMCI [867](#)

TsgcBotClassification [271](#)

TsgcHTTP\_API\_Anthropic [888](#)

TsgcHTTP\_API\_DeepSeek [918](#)

TsgcHTTP\_API\_Gemini [915](#)

TsgcHTTP\_API\_Grok [943](#)

TsgcHTTP\_API\_Mistral [949](#)

TsgcHTTP\_API\_Ollama [937](#)

TsgcHTTP\_API\_OpenAI [843](#)

TsgcHTTP\_JWT\_Client [1076](#)

TsgcHTTP\_JWT\_Server [1080](#)

TsgcHTTP\_OAuth2\_Client [1013](#)

TsgcHTTP\_OAuth2\_Client\_Google [1020](#)

TsgcHTTP\_OAuth2\_Client\_Microsoft [1021](#)

TsgcHTTP\_OAuth2\_Server [1036](#)

TsgcHTTP\_OAuth2\_Server\_Provider [1065](#)

TsgcHTTP2Client [979](#)

TsgcHTTP2ConnectionClient [994](#)

TsgcHTTP2RequestProperty [995](#)

TsgcHTTP2ResponseProperty [996](#)

TsgcICEClient [1193](#)

TsgcIWWebSocketClient [320](#)

TsgcIWWSPClient\_Dataset [460](#)

TsgcIWWSPClient\_sgc [450](#)

TsgcRTCPeerConnection [1197](#)

TsgcServerHTTPAPI\_FineTune [238](#)

TsgcSTUNClient [1171](#)

TsgcSTUNServer [1176](#)

TsgcTCPClient [188](#)

TsgcTextToSpeechAmazon [871](#)

TsgcTextToSpeechGoogle [870](#)

TsgcTextToSpeechSystem [869](#)

TsgcTURNClient [1181](#)

TsgcTURNServer [1187](#)

TsgcUDPClient [1168](#)

TsgcUDPServer [1169](#)

TsgcWebPush\_Client [792](#)

TsgcWebSocketClient [158](#)

TsgcWebSocketClient\_WinHTTP [251](#)

TsgcWebSocketFirewall [252](#), [261](#), [263](#), [265](#), [268](#), [271](#)

TsgcWebSocketHTTPServer [215](#)

TsgcWebSocketHTTPServer\_Sessions [227](#)

TsgcWebSocketLoadBalancerServer [317](#)

TsgcWebSocketProxyServer [319](#)

TsgcWebSocketServer [190](#)

TsgcWebSocketServer\_HTTPAPI [233](#)

TsgcWebView2 [1149](#)

TsgcWSAPIClient\_MCP [819](#)

TsgcWSAPIKeyManager [302](#)

TsgcWSAPIServer\_MCP [797](#)

TsgcWSAPIServer\_WebAuthn [1082](#)

TsgcWSAPIServer\_WebPush [791](#)

TsgcWSCircuitBreaker [289](#)

TsgcWSConnection [322](#)

TsgcWSHTTP2WebBrokerBridgeServer [1212](#)

TsgcWSHTTPWebBrokerBridgeServer [1210](#)

TsgcWSMessageFile [471](#)

TsgcWSPClient\_AMQP [352](#)

TsgcWSPClient\_AMQP1 [387](#)

TsgcWSPClient\_Dataset [459](#)

TsgcWSPClient\_E2EE [488](#)

TsgcWSPClient\_Files [470](#)

TsgcWSPClient\_Kafka [376](#)

TsgcWSPClient\_MQTT [334](#)

TsgcWSPClient\_Presence [479](#)

TsgcWSPClient\_sgc [449](#)

TsgcWSPClient\_STOMP [416](#)

TsgcWSPClient\_STOMP\_ActiveMQ [418](#)

TsgcWSPClient\_STOMP\_RabbitMQ [417](#)

TsgcWSPClient\_WAMP [430](#)

TsgcWSPClient\_WAMP2 [442](#)

TsgcWSPPresenceMessage [477](#)

TsgcWSPServer\_AppRTC [421](#)

TsgcWSPServer\_Dataset [458](#)

TsgcWSPServer\_E2EE [487](#)

[TsgcWSPServer\\_Files](#) [469](#)

[TsgcWSPServer\\_Presence](#) [476](#)

[TsgcWSPServer\\_sgc](#) [448](#)

[TsgcWSPServer\\_WAMP](#) [429](#)

[TsgcWSPServer\\_WebRTC](#) [424](#)

[TsgcWSRateLimiter](#) [275](#)

[TsgcWSServer\\_HTTPAPI\\_WebBrokerBridge](#) [1213](#)

[TURN](#) [1179](#), [1183](#), [1184](#), [1185](#), [1186](#), [1189](#), [1190](#)

[TURN Client Send Indication](#) [1185](#)

[TURN Server Long Term Credentials](#) [1189](#)

[Tworzenie uprawnień klienta TURN](#) [1184](#)

[Unieważnienie tokenu \(RFC 7009\)](#) [1059](#)

[uruchamianie serwera](#) [192](#)

[Usługa Windows](#) [102](#)

[Ustawienia WebView2](#) [1162](#)

[utrzymywanie aktywności serwera](#) [195](#)

[utrzymywanie aktywnych połączeń](#) [203](#)

[Uwierzytelnianie](#) [105](#), [182](#)

[Uwierzytelnianie dostawcy OAuth2](#) [1070](#)

[Uwierzytelnianie klienta](#) [182](#), [990](#)

[Uwierzytelnianie klienta AMQP1](#) [397](#)

[uwierzytelnianie serwera](#) [1299](#)

[Uwierzytelnianie WebAuthn](#) [1097](#)

[Uzgadnianie WebSocket klienta](#) [185](#)

[Uzupełnienie OpenAI](#) [844](#)

[Wait Response](#) [348](#)

[Walidacja DPOP \(RFC 9449\)](#) [1063](#)

[WAMP](#) [435](#), [436](#), [437](#), [439](#)

[WAMP RPC Progress Results](#) [439](#)

[WAMP Simple RPC](#) [437](#)

[WatchDog](#) [113](#)

[Wątki serwera HTTP/2](#) [223](#)

[WebAuthn](#) [1081](#)

[WebAuthn Authorization HTTP](#) [1107](#)

[WebAuthn Authorization WebSocket](#) [1108](#)

[WebAuthn MDS](#) [1104](#)

[Webhook WhatsApp](#) [740](#)

[WebPush](#) [789](#)

[WebRTC](#) [1303](#)

[WebSockets](#) [53](#), [98](#), [100](#), [160](#), [171](#), [185](#), [500](#), [501](#), [527](#), [528](#), [548](#), [556](#), [568](#), [581](#), [681](#), [682](#)

[WebSockets Private](#) [556](#), [581](#)

[WebSockets Public](#) [548](#), [568](#)

[WebView2 Advanced Features](#) [1164](#)

[WebView2 JavaScript](#) [1155](#)

[Weryfikacja certyfikatu serwera](#) [202](#)

[WhatsApp — odbieranie wiadomości i powiadomień o statusie](#) [750](#)

[WhatsApp — pobieranie multimediów](#) [755](#)

[WhatsApp — wysyłanie plików](#) [752](#)

[WhatsApp — wysyłanie wiadomości interaktywnych](#) [745](#)

[WhatsApp Tworzenie aplikacji](#) [737](#)

[WhatsApp Wysyłanie Wiadomości Szablonowych](#) [748](#)

[Wiadomości Anthropic](#) [889](#)

[Wiadomości DeepSeek](#) [919](#)

[Wiadomości Gemini](#) [925](#)

[Wiadomości Grok](#) [944](#)

[Wiadomości Mistral](#) [950](#)

Wiadomości Ollama [938](#)

Wprowadzenie [14](#)

wydawcy WAMP [436](#)

Wymiana AMQP [361](#)

Wynik rejestracji WebAuthn [1095](#)

Wynik uwierzytelniania WebAuthn [1103](#)

Wyślij pogrubioną wiadomość Telegram [771](#)

Wyślij wiadomość faktury Telegram [774](#)

Wyślij wiadomość Telegram z przyciskami [769](#), [770](#)

Wyślij wiadomość Telegram z przyciskami inline [769](#)

wysyłanie dużych plików [474](#)

Wysyłanie plików [472](#), [473](#)

    Klienci [473](#)

    Serwer [472](#)

Wysyłanie plików do klientów [473](#)

Wysyłanie plików do serwera [472](#)

Wysyłanie wiadomości WhatsApp [742](#)

Wyszukiwanie w sieci Anthropic [904](#)

XSS [265](#)

Żądania dostawcy OAuth2 [1073](#)

żądanie HTTP [981](#)

Żądanie rejestracji WebAuthn [1091](#)

Żądanie uwierzytelnienia WebAuthn [1100](#)

zakaz IP [263](#)

Zamknięcie połączenia przez klienta [164](#), [986](#)

Zamknięcie połączenia przez serwer [205](#)

Zasoby klienta MCP [825](#)

Zasoby serwera MCP [809](#)

Zdarzenia WebSocket [98](#)

Zlecenia [677](#)

Zliczanie tokenów Gemini [931](#)

Znaleziono [772](#)

