TEL-STER Sp. z o.o.
ul. Stefana Stefańskiego 23
62-002 Suchy Las
Tel. +48 61 628 97 50
Fax. +48 61 639 37 11

TelCOMM 5.0

Sąd Rejonowy w Poznaniu, VIII Wydział Gospodarczy Krajowego Rejestru Sądowego
KRS: 0000178284; NIP: 7811738677; REGON: 634530767
Kapitał **Zakładowy**: 560 000 zł

TEL-STER Sp. z o.o.
ul. Stefana Stefańskiego 23
62-002 Suchy Las
Tel. +48 61 628 97 50
Fax. +48 61 639 37 11

Historia zmian dokumentu:

Data	Wersja dokumentu	Autor	Opis zmiany
2019.04.17	4.0	Michał Siatkowski	Wersja 4.0
2019.05.30	4.0	Elżbieta Sobkowiak	Dostosowanie formy do wymogów ISO
2019.12.18	4.2	Michał Siatkowski	Dodatkowe funkcjonalności
2020.07.07	4.5	Michał Siatkowski	Jeden P-Mode dla obu kierunków
2020.12.14	4.6	Michał Siatkowski	Aktualizacje bezpieczeństwa
2022.05.17	5.0	Michał Siatkowski	Implementacja AS2
2024.01.30	5.1	Michał Siatkowski	Aktualizacje dot. interfejsu do współpracy z aplikacją zewnętrzną

Spis treści

Specyfikacja

- Protokoły: AS4, AS2 w wersji 1.1
- Wzorce komunikacji (MEP) AS4: One-Way/Push, Two-Way/Push-Pull, One-Way/Pull jako partner inicjujący
- Algorytmy AS4:
    - Funkcje skrótu dla podpisu (hash): sha256, sha384, sha512
    - Podpis cyfrowy:
        - certyfikat RSA: rsa-sha256, rsa-sha384, rsa-sha512
        - certyfikat ECC: ecdsa-sha256, ecdsa-sha384, ecdsa-sha512
    - Szyfrowanie danych:
        - aes128-cbc, aes192-cbc, aes256-cbc
        - aes128-gcm, aes192-gcm, aes256-gcm
    - Szyfrowanie klucza:
        - certyfikat RSA: rsa-oaep-mgf1p, rsa-oaep
            - MGF: mgf1sha1, mgf1sha256, mgf1sha384, mgf1sha512
            - Funkcje skrótu (hash): sha1, sha256, sha384, sha512
        - certyfikat ECC: ECDH-ES
            - KDF: ConcatKDF
            - KW: kw-aes128, kw-aes192, kw-aes256
            - HMAC: sha1, sha256, sha384, sha512
    - Kompresja: gzip
- Algorytmy AS2:
    - Podpis cyfrowy:
        - sha256, sha384, sha512
        - sha256-rsassa-pss, sha384-rsassa-pss, sha512-rsassa-pss
    - Szyfrowanie: aes128-cbc, aes192-cbc, aes256-cbc
    - Kompresja: zlib
- [ReplyPattern]: Response (synchroniczność), Callback (asynchroniczność, dla AS4)
- [SecurityTokenReference] (AS4): BinarySecurityToken X509v3, BinarySecurityToken X509PKIPathv1, IssuerAndSerialNumber

- Automatyczna aktualizacja certyfikatów między partnerami: tak, dla AS4
- Wiele załączników w komunikacie: tak, dla AS4
- Jeden [P-Mode] dla obu kierunków (gdy partnerzy używają tego samego certyfikatu do podpisu i szyfrowania): tak
- Walidacja certyfikatów: OCSP/CRL (opcjonalnie)
- Certyfikat TLS serwera i klienta: tak (opcjonalnie)
- Architektura: Microsoft Windows Server, IIS, .NET Framework 4.8, HTTPS, TLS 1.2 i 1.3
- Baza danych: SQLite, Oracle
- Interakcja z aplikacją: przeglądarka internetowa dla użytkowników, usługa internetowa (Web Service) dla aplikacji zewnętrznych
- Język w aplikacji: polski, angielski

Wprowadzenie

Oprogramowanie TelCOMM jest narzędziem umożliwiającym wymianę dokumentów typu B2B za pomocą protokołów AS4 i AS2 w wersji 1.1.

Protokół AS4 (Applicability Statement 4) to standard opisujący bezpieczne i niezawodne przesyłanie plików przez publiczną sieć Internet. Protokół ten bazuje na powszechnie znanych i sprawdzonych rozwiązaniach, takich jak protokoły HTTP, TLS, SOAP oraz usługach sieciowych (Web Service). Reprezentuje otwarty standard wymiany danych typu B2B opisany w specyfikacji OASIS ebMS 3.0. Elementami odpowiedzialnymi za bezpieczeństwo i wiarygodność przesyłanych danych są podpisy cyfrowe oraz mechanizmy szyfrujące (WS-security).

Protokół AS2 (Applicability Statement 2) powstał wcześniej niż AS4, bazuje na protokołach HTTP i TLS, funkcjonalnie podobny do AS4, ale w praktycznym użyciu został w większości przez niego zastąpiony i obecnie używany jest raczej w celach kompatybilności ze starszymi systemami.

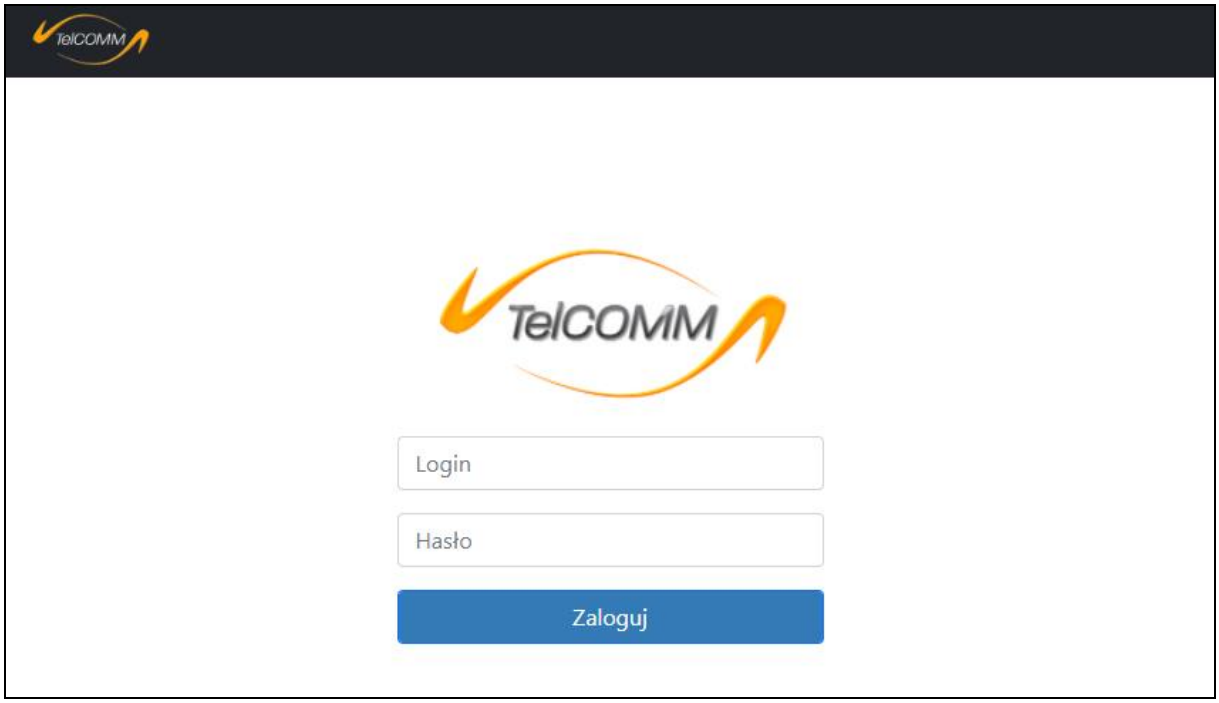
Architektura

Oprogramowanie TelCOMM jest rozwiązaniem adresowanym dla środowiska MS Windows Server, bazującym na podsystemie IIS oraz .NET Framework w wersji co najmniej 4.8. Aplikacja może być udostępniana jedynie w oparciu o protokół HTTPS, a wersja używanego protokołu TLS podczas wysyłania komunikatów to 1.2 lub 1.3. Korzysta ona z wbudowanej bazy danych **SQLite** lub ma możliwość korzystania z bazy danych **Oracle**. Program składa się z trzech komponentów:

- aplikacji interfejsowej dostępnej z poziomu przeglądarki internetowej, umożliwiającej konfigurację i interaktywną wymianę dokumentów,
- usługi internetowej /WebServices/Gateway.asmx umożliwiającej wymianę dokumentów z poziomu aplikacji zewnętrznej,
- usługi internetowej /MSH.asmx/Receive umożliwiającej odbiór dokumentów od partnerów.

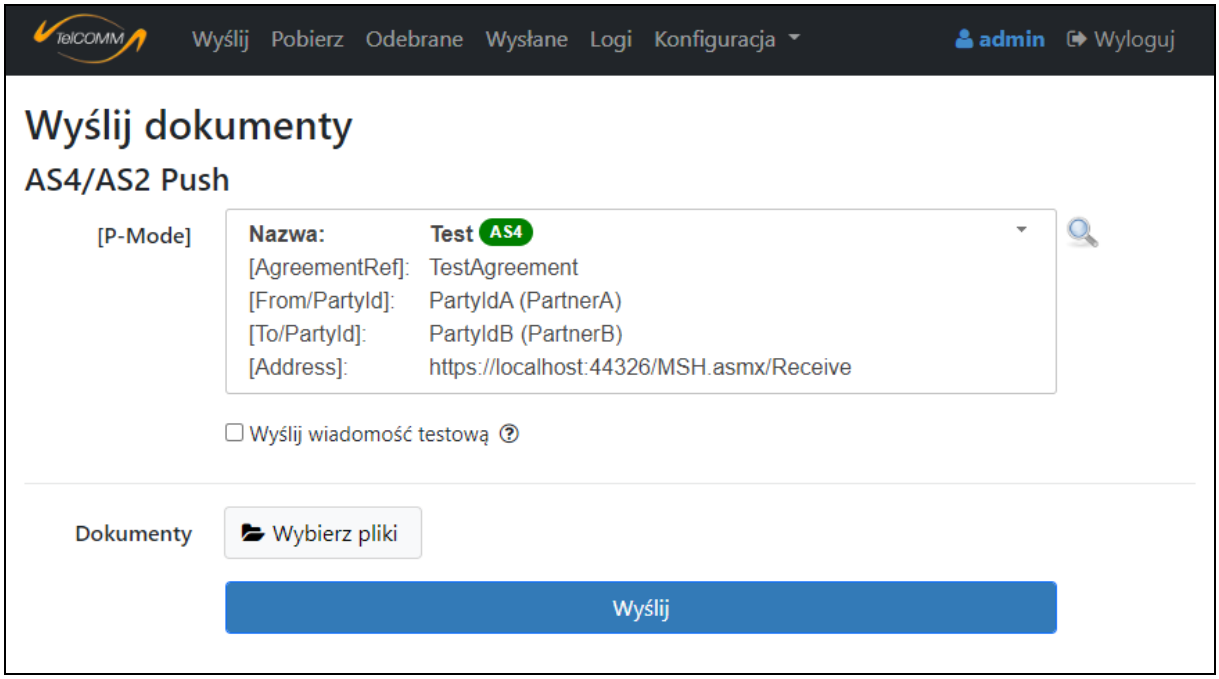
Aplikacja interfejsowa

Pierwszym ekranem aplikacji TelCOMM jest ekran logowania. Aby móc z niej korzystać należy się zalogować. W aplikacji istnieje nieusuwalny użytkownik, dla którego początkowy login i hasło to „admin”. Kliknięcie w ikonę TelCOMM w lewym górnym rogu aplikacji spowoduje wyświetlenie numeru wersji.



Rysunek 1. Ekran logowania

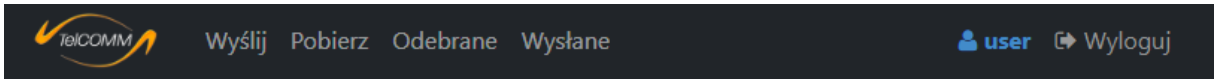
Po zalogowaniu pojawia się główny ekran aplikacji „Wyślij”.



Rysunek 2. Wygląd aplikacji po zalogowaniu

U góry ekranu znajduje się pasek nawigacyjny aplikacji wspólny dla ekranów po zalogowaniu. Po ikonie aplikacji znajduje się hiperłącze do ekranu „Wyślij” oraz „Pobierz”, a następnie do pozostałych ekranów aplikacji. Na końcu znajduje się nazwa zalogowanego użytkownika wraz z opcją wylogowania. Pasek ten różni się w zależności od tego czy zalogowany użytkownik jest

administratorem czy nie. Zwykły użytkownik ma dostęp do ekranów „Wyślij", „Pobierz", „Odebrane" i „Wysłane". Poniżej zrzut ekranu paska nawigacyjnego dla zwykłego użytkownika.



Rysunek 3. Pasek nawigacyjny użytkownika

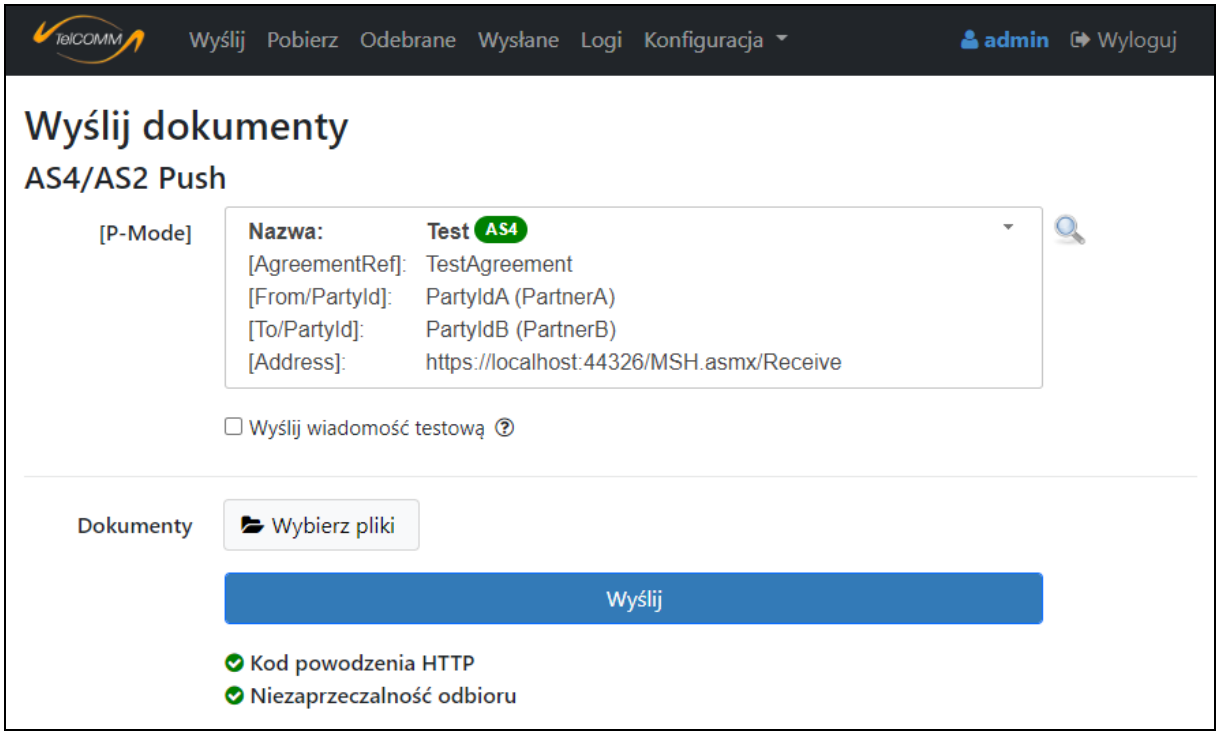
EKRAN „WYŚLIJ"

Służy do wysyłania plików za pomocą wzorca komunikacji **One-Way/Push**. Polega on na jednostronnym wysłaniu wiadomości od jednego partnera do drugiego. W celu wysłania komunikatu AS4/AS2 należy:

- wybrać wcześniej zdefiniowany [P-Mode] ([Processing mode]), w którym to zawarty jest komplet informacji na temat połączenia komunikacyjnego między partnerami,
- w polu „Dokumenty" wybrać plik lub pliki, które mają zostać wysłane.

Kliknięcie przycisku „Wyślij" inicjuje komunikację - dokumenty zostają wysłane zgodnie z protokołem komunikacyjnym AS4/AS2 oraz następuje analiza odpowiedzi od odbiorcy pod kątem:

- braku błędów wysłania wiadomości – kod powodzenia HTTP,
- niezaprzeczalności odbioru (Non Repudiation of Receipt) lub potwierdzenia odbioru (Reception Awareness), gdy odpowiedź przesyłana jest synchronicznie (Response) – możliwe jest również asynchroniczne odebranie odpowiedzi w osobnym komunikacie (Callback, dla AS4).



Rysunek 4. Ekran „Wyślij"– po analizie wysyłki

W przypadku otrzymania w odpowiedzi błędów komunikacji AS4/AS2 zostaną one wyświetlone na dole ekranu i zapisane w celu późniejszego podglądu na ekranie „Logi".

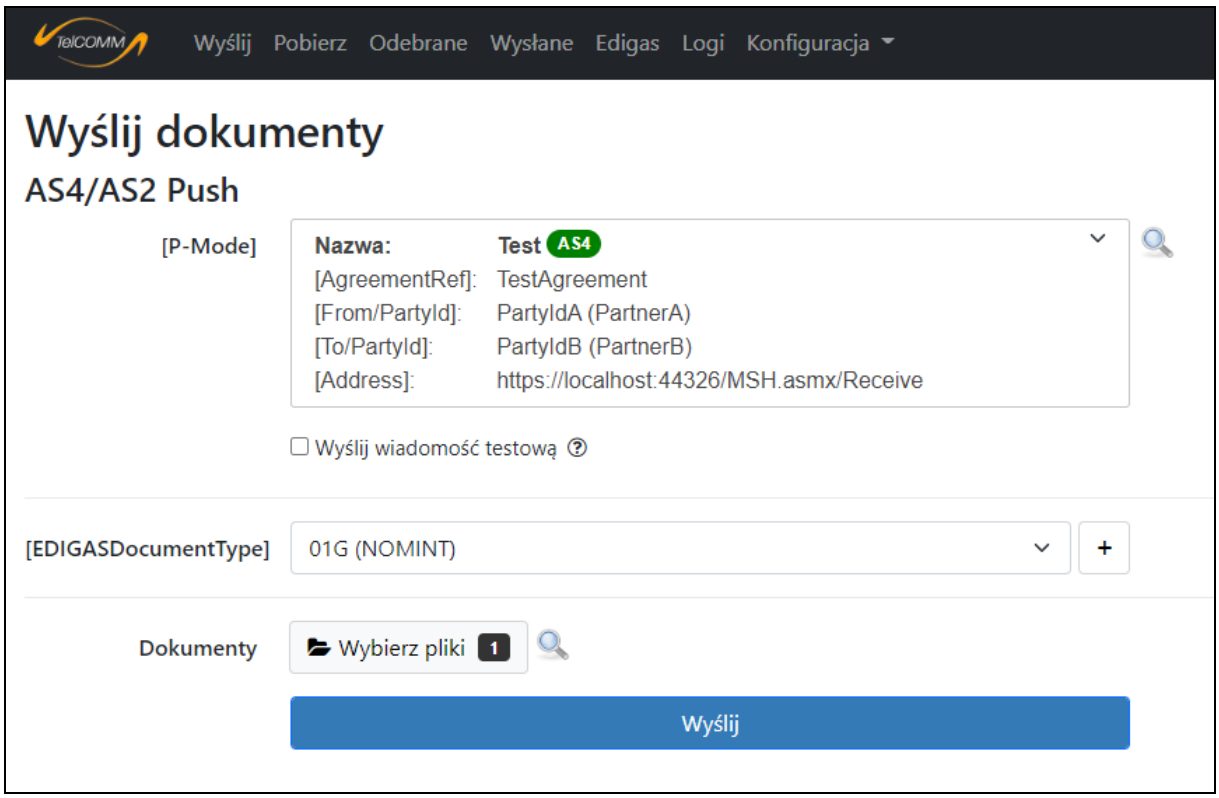
Przydatnymi opcjami ekranu wysyłania są:

- możliwość wysłania wiadomości z testowymi parametrami komunikacji AS4 w celu przetestowania połączenia między partnerami – należy zaznaczyć opcję „Wyślij wiadomość testową”,
- w sytuacji wysyłania wielu plików, możliwość wysłania ich pojedynczo w osobnych wiadomościach – opcja pojawi się po wybraniu więcej niż jednego pliku.

W przypadku włączenia w konfiguracji aplikacji opcji „Wysyłanie dokumentów EDIGAS” dodatkowymi wariantami będzie:

- możliwość niewypełniania pola [EDIGASDocumentType], lecz pobrania go z załączonych plików Edig@s – opcja „Pobierz z załączonych dokumentów” w rozwijanej liście,
- kryjące się pod przyciskiem „+” dodatkowe pola „Identyfikator transakcji” i „Opis transakcji” służące do opisania wysłanych nominacji jako pojedynczej transakcji zawierającej jeden lub wiele plików. Umożliwia to opcjonalne grupowanie nominacji na ekranie „Edigas”.

Dodatkowo w trybie „Wysyłanie dokumentów EDIGAS” występuje możliwość wysyłania plików niebędących dokumentami Edig@s – w polu [EDIGASDocumentType] należy wybrać „Brak”. Przykład ekranu w trybie „Wysyłanie dokumentów EDIGAS” przedstawia poniższy rysunek.



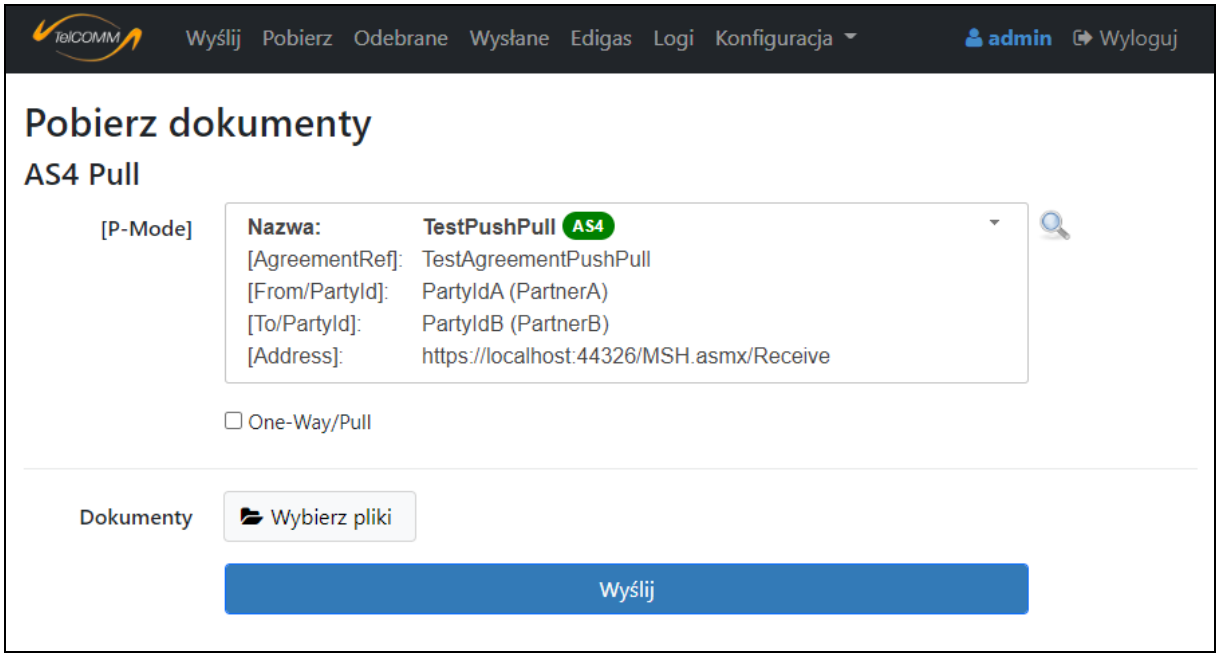
Rysunek 5. Ekran „Wyślij” w trybie "Wysyłanie dokumentów EDIGAS"

EKRAN „POBIERZ”

Służy do pobierania danych od partnera z użyciem wzorca komunikacji **One-Way/Pull** lub **Two-Way/Push-Pull**, który składa się z dwóch etapów. W pierwszym etapie wysyłany jest dokument z żądaniem określającym jakie dane partner ma udostępnić, a w drugim etapie dane te są w sposób automatyczny pobierane. One-Way/Pull obejmuje tylko drugi etap – utworzenie żądania Pull i dodanie go do puli żądań realizowanych automatycznie. W celu pobrania danych AS4 należy:

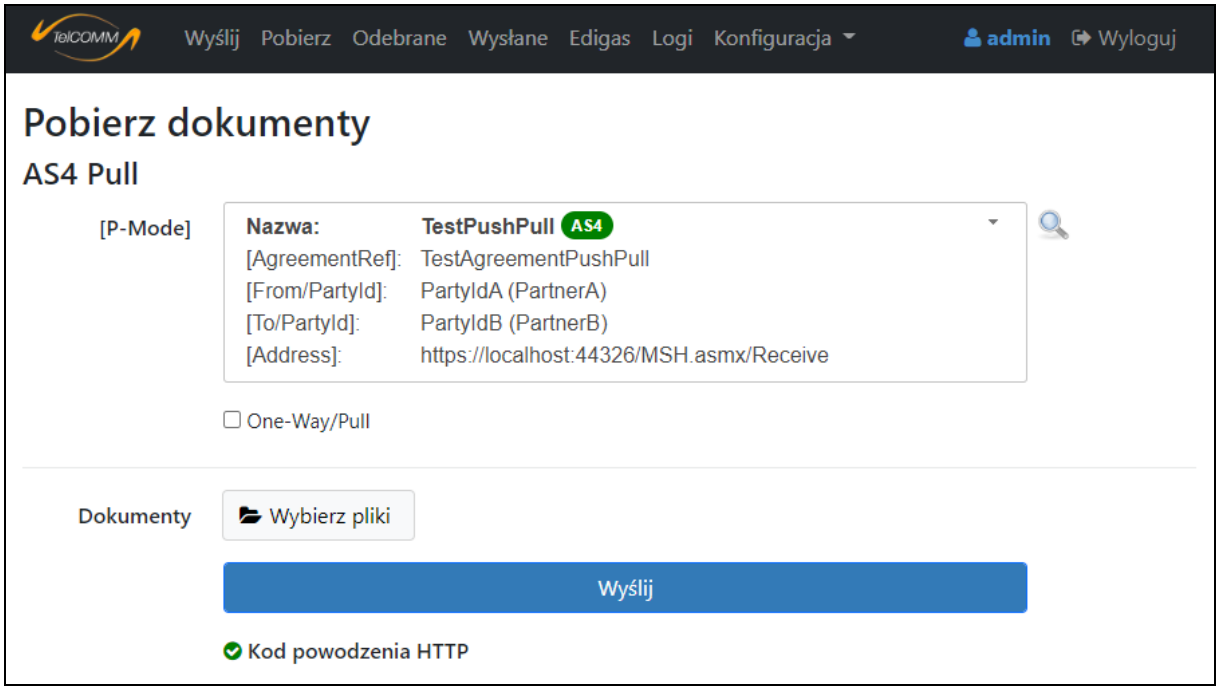
- wybrać wcześniej zdefiniowany [P-Mode] ([Processing mode]), w którym to zawarty jest komplet informacji na temat połączenia komunikacyjnego między partnerami,

- w polu „Dokumenty” wybrać plik lub pliki zawierające żądanie o udostępnienie konkretnych danych. Struktura pliku jest ustalana między partnerami.



Rysunek 6. Ekran „Pobierz”

Kliknięcie przycisku „Wyślij” inicjuje pierwszą fazę komunikacji – plik z żądaniem zostaje wysłany zgodnie z protokołem komunikacyjnym AS4 oraz następuje analiza odpowiedzi od odbiorcy pod kątem braku błędów wysłania wiadomości – kod powodzenia HTTP.



Rysunek 7. Ekran „Pobierz”– po analizie wysyłki

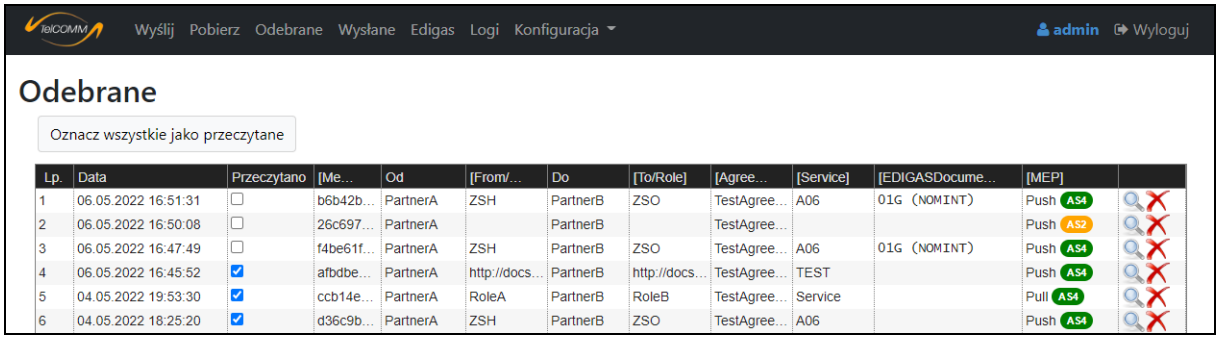
W przypadku otrzymania w odpowiedzi błędów komunikacji AS4 zostaną one wyświetlone na dole ekranu pobierania i zapisane w celu późniejszego podglądu na ekranie „Logi".

Gdy będzie zaznaczona opcja „One-Way/Pull" wykonane zostanie tylko pobranie danych bez uprzedniego wysłania żądania. Wtedy nie ma możliwości dodania dokumentów.

Dodatkową opcją ekranu pobierania jest, w przypadku wysyłania wielu żądań, możliwość wysłania ich pojedynczo w osobnych wiadomościach – opcja pojawi się po wybraniu więcej niż jednego pliku.

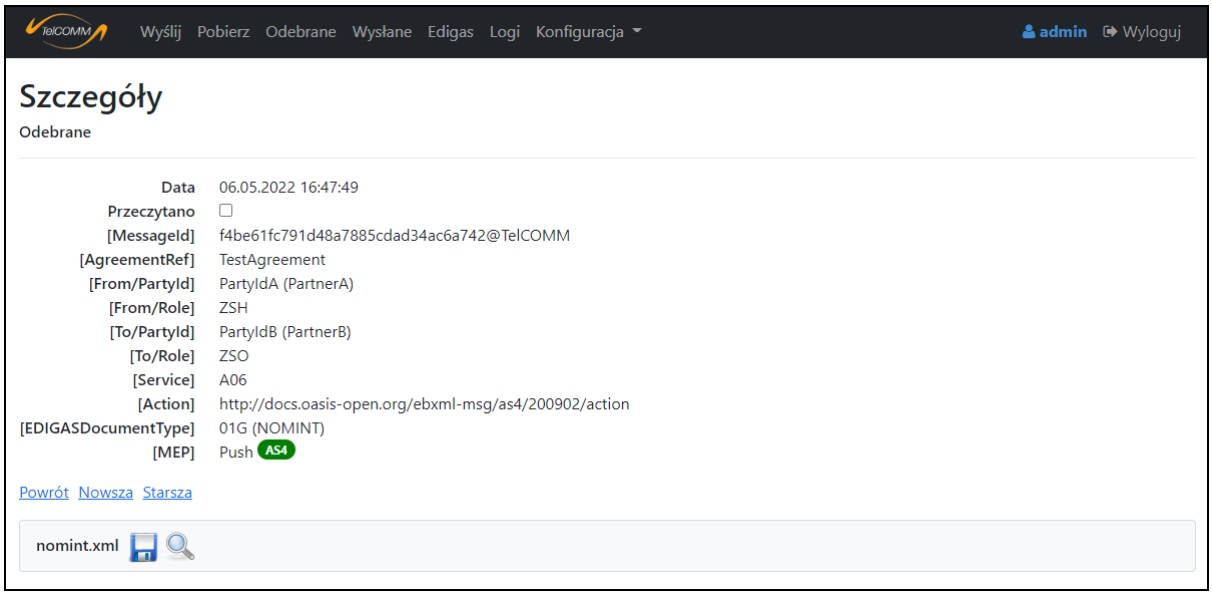
EKRAN „ODEBRANE"

Ekran „Odebrane" spełnia rolę skrzynki odbiorczej aplikacji. Dla każdej odebranej wiadomości/żądania istnieje możliwość nadania jej statusu przeczytanej/nieprzeczytanej – funkcjonalność ta ma przede wszystkim znaczenie przy korzystaniu z aplikacji przez funkcję usługi internetowej (Web Service), gdzie pobierane są wszystkie nieprzeczytane pozycje. Można również oznaczyć wszystkie pozycje jako przeczytane za pomocą przycisku na górze ekranu.



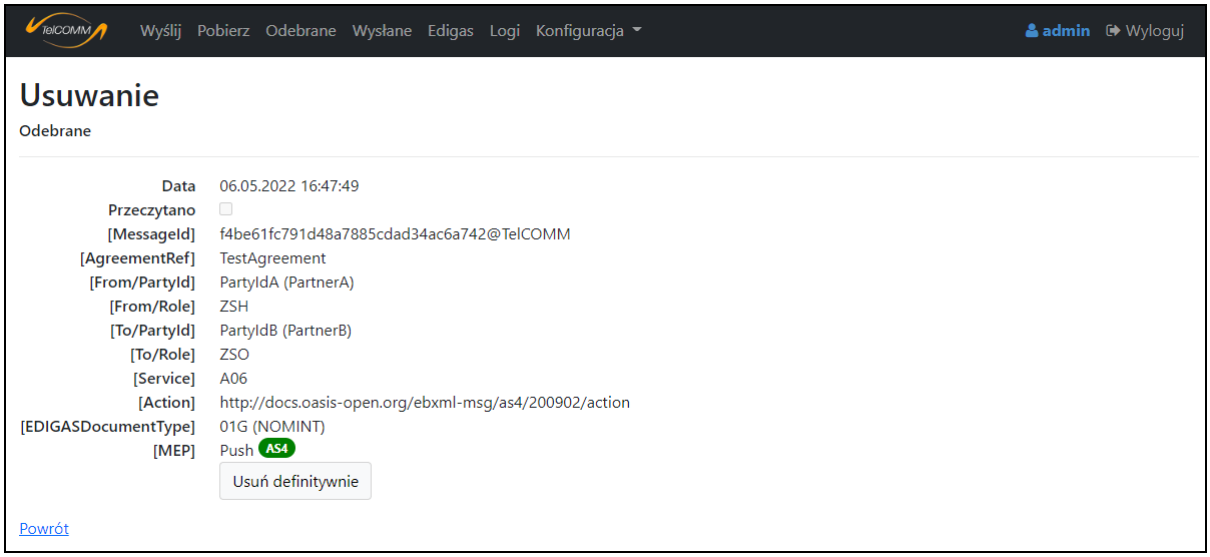
Rysunek 8. Ekran „Odebrane"

W obrębie aplikacji użyto zestawu ikonek , które odpowiednio służą do dodawania, podglądu szczegółów, edycji i usuwania danego elementu. Aby wyświetlić przesłane pliki w danym komunikacie AS4/AS2 należy przejść do szczegółów komunikatu. Można również przełączać się między pozycjami bezpośrednio na ekranie szczegółów korzystając z hiperłączy „Nowsza" i „Starsza".



Rysunek 9. Ekran „Odebrane" – szczegóły komunikatu

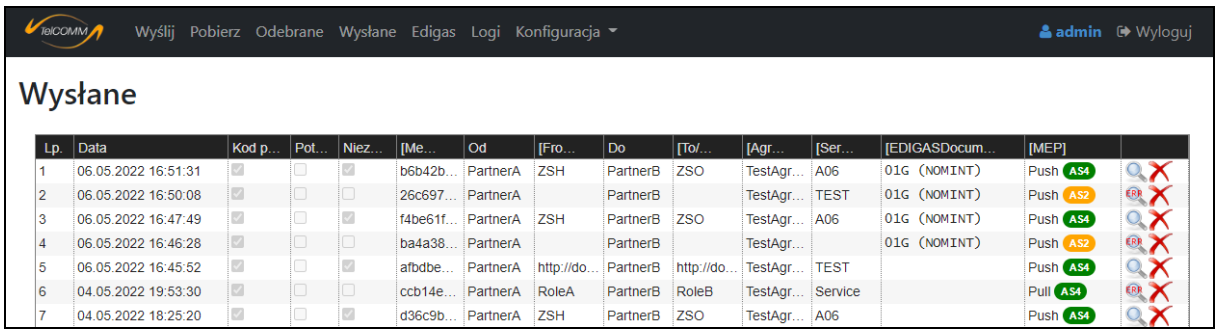
Na ekranie „Odebrane” istnieje również opcja usunięcia wiadomości. Kliknięcie przycisku powoduje wyświetlenie ekranu usuwania danego komunikatu i dopiero na nim należy potwierdzić jej usunięcie przyciskiem „Usuń definitywnie”, aby odpowiedni wpis z bazy danych został trwale usunięty. Taka procedura usuwania, polegająca na potwierdzeniu na osobnym ekranie, jest stosowana w obrębie całej aplikacji.



Rysunek 10. Ekran „Odebrane” – usuwanie wiadomości

EKRAN „WYSŁANE”

Ekran „Wysłane” wyświetla wszystkie komunikaty, które zostały wysłane, a jego struktura jest analogiczna do ekranu „Odebrane”, z tą różnicą, że zamiast opcji nadania statusu wiadomości przeczytanej/nieprzeczytanej znajdują się informacje dotyczące wyniku analizy odpowiedzi od odbiorcy, opisane wcześniej. Również w przypadku wysłania wiadomości z błędami pojawia się nieco zmodyfikowana ikona podglądu szczegółów wiadomości widoczna na poniższym rysunku.

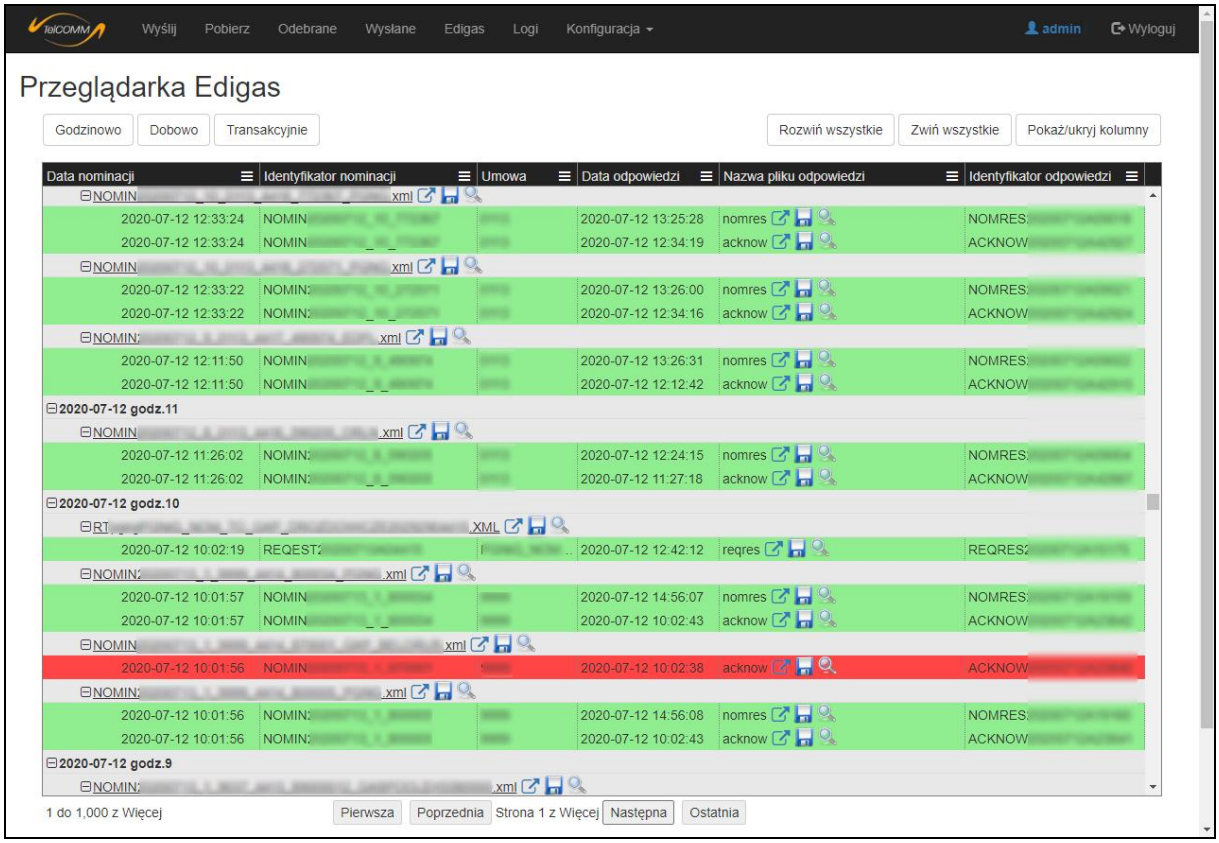


Rysunek 11. Ekran „Wysłane”

EKRAN „EDIGAS”

Ekran „Edigas” dostępny jest tylko przy zaznaczonej opcji „Wysyłanie dokumentów EDIGAS” w konfiguracji aplikacji. Pozwala on przeglądać wysłane oraz odebrane dokumenty Edig@s na zasadzie parowania. Funkcja parowania pozwala powiązać dokument NOMINT z ACKNOW i NOMRES oraz REQEST z REQRES w sposób, jaki przedstawiono na poniższym rysunku. Dokumenty odpowiedzi wyróżnione są kolorem w zależności od tego czy mają status zatwierdzenia (kolor zielony) czy odrzucenia (czerwony). Dokumenty można posortować według informacji zawartych w pliku (Identyfikator i nr Umowy). Przyciski „Godzinowo”, „Dobowo” i „Transakcyjnie” pozwalają pogrupować dokumenty wg. godziny, daty lub według „Identyfikatora” i „Opisu Transakcji” (jeżeli

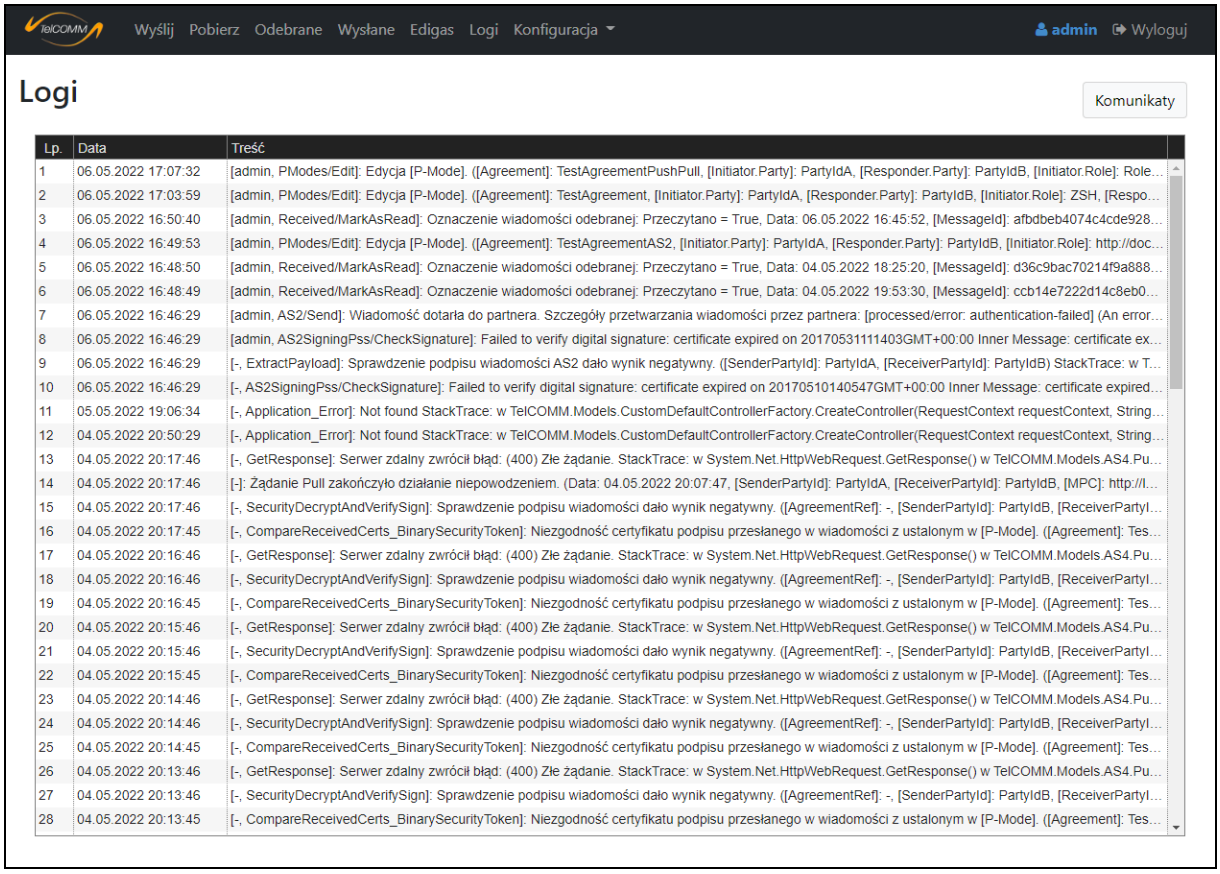
podano podczas wysłania). Przycisk „Rozwiń wszystkie” pozwala pokazać na ekranie wszystkie dokumenty, natomiast „Zwiń wszystkie” ukrywa dokumenty, dając możliwość wyświetlenia dokumentów tylko z wybranej grupy. Domyślnie nie wszystkie kolumny są widoczne na ekranie - opcja „Pokaż/ukryj kolumny” pozwala na zarządzanie widocznością kolumn. Kliknięcie w przycisk pozwala zapisać dokument na komputerze, przycisk pozwala podejrzeć zawartość pliku w karcie przeglądarki, a przycisk to hiperłącze do ekranu szczegółów wiadomości, w której został przesłany dany dokument.



Rysunek 12. Ekran „Edigas”

EKRAN „LOGI”

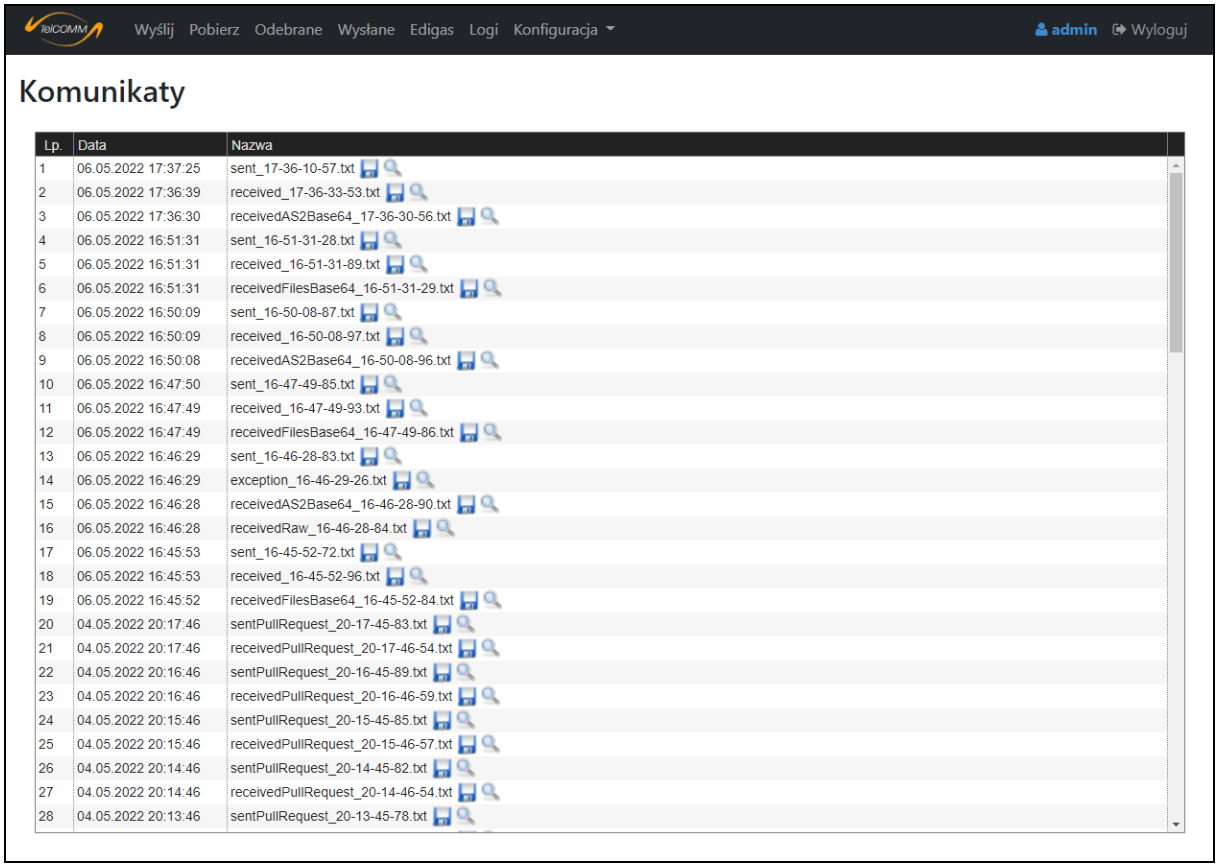
Kolejne ekrany dostępne są jedynie dla administratora. Ekran „Logi” zawiera błędy komunikacji AS4/AS2 oraz wybrane informacje na temat działań użytkowników. Dwukrotne kliknięcie w kolumnie „Treść” pozwala wyświetlić pełną treść komunikatu.



Lp.	Data	Treść
1	06.05.2022 17:07:32	[admin, PModes/Edit]: Edycja [P-Mode]. ([Agreement]: TestAgreementPushPull, [Initiator.Party]: PartyIdA, [Responder.Party]: PartyIdB, [Initiator.Role]: Role...
2	06.05.2022 17:03:59	[admin, PModes/Edit]: Edycja [P-Mode]. ([Agreement]: TestAgreement, [Initiator.Party]: PartyIdA, [Responder.Party]: PartyIdB, [Initiator.Role]: ZSH, [Respo...
3	06.05.2022 16:50:40	[admin, Received/MarkAsRead]: Oznaczenie wiadomości odebranej: Przeczytano = True, Data: 06.05.2022 16:45:52, [MessageId]: afbdbeb4074c4cde928...
4	06.05.2022 16:49:53	[admin, PModes/Edit]: Edycja [P-Mode]. ([Agreement]: TestAgreementAS2, [Initiator.Party]: PartyIdA, [Responder.Party]: PartyIdB, [Initiator.Role]: http://doc...
5	06.05.2022 16:48:50	[admin, Received/MarkAsRead]: Oznaczenie wiadomości odebranej: Przeczytano = True, Data: 04.05.2022 18:25:20, [MessageId]: d36c9bac70214f9a888...
6	06.05.2022 16:48:49	[admin, Received/MarkAsRead]: Oznaczenie wiadomości odebranej: Przeczytano = True, Data: 04.05.2022 19:53:30, [MessageId]: ccb14e7222d14c8eb0...
7	06.05.2022 16:46:29	[admin, AS2/Send]: Wiadomość dotarła do partnera. Szczegóły przetwarzania wiadomości przez partnera: [processed/error: authentication-failed] (An error...
8	06.05.2022 16:46:29	[admin, AS2SigningPss/CheckSignature]: Failed to verify digital signature: certificate expired on 20170531111403GMT+00:00 Inner Message: certificate ex...
9	06.05.2022 16:46:29	[-, ExtractPayload]: Sprawdzenie podpisu wiadomości AS2 dało wynik negatywny. ([SenderPartyId]: PartyIdA, [ReceiverPartyId]: PartyIdB) StackTrace: w T...
10	06.05.2022 16:46:29	[-, AS2SigningPss/CheckSignature]: Failed to verify digital signature: certificate expired on 20170510140547GMT+00:00 Inner Message: certificate expired...
11	05.05.2022 19:06:34	[-, Application_Error]: Not found StackTrace: w TelCOMM.Models.CustomDefaultControllerFactory.CreateController(RequestContext requestContext, String...
12	04.05.2022 20:50:29	[-, Application_Error]: Not found StackTrace: w TelCOMM.Models.CustomDefaultControllerFactory.CreateController(RequestContext requestContext, String...
13	04.05.2022 20:17:46	[-, GetResponse]: Serwer zdalny zwrócił błąd: (400) Złe żądanie. StackTrace: w System.Net.HttpWebRequest.GetResponse() w TelCOMM.Models.AS4.Pu...
14	04.05.2022 20:17:46	[-]: Żądanie Pull zakończyło działanie niepowodzeniem. (Data: 04.05.2022 20:07:47, [SenderPartyId]: PartyIdA, [ReceiverPartyId]: PartyIdB, [MPC]: http://l...
15	04.05.2022 20:17:46	[-, SecurityDecryptAndVerifySign]: Sprawdzenie podpisu wiadomości dało wynik negatywny. ([AgreementRef]: -, [SenderPartyId]: PartyIdB, [ReceiverPartyI...
16	04.05.2022 20:17:45	[-, CompareReceivedCerts_BinarySecurityToken]: Niezgodność certyfikatu podpisu przesłanego w wiadomości z ustalonym w [P-Mode]. ([Agreement]: Tes...
17	04.05.2022 20:16:46	[-, GetResponse]: Serwer zdalny zwrócił błąd: (400) Złe żądanie. StackTrace: w System.Net.HttpWebRequest.GetResponse() w TelCOMM.Models.AS4.Pu...
18	04.05.2022 20:16:46	[-, SecurityDecryptAndVerifySign]: Sprawdzenie podpisu wiadomości dało wynik negatywny. ([AgreementRef]: -, [SenderPartyId]: PartyIdB, [ReceiverPartyI...
19	04.05.2022 20:16:45	[-, CompareReceivedCerts_BinarySecurityToken]: Niezgodność certyfikatu podpisu przesłanego w wiadomości z ustalonym w [P-Mode]. ([Agreement]: Tes...
20	04.05.2022 20:15:46	[-, GetResponse]: Serwer zdalny zwrócił błąd: (400) Złe żądanie. StackTrace: w System.Net.HttpWebRequest.GetResponse() w TelCOMM.Models.AS4.Pu...
21	04.05.2022 20:15:46	[-, SecurityDecryptAndVerifySign]: Sprawdzenie podpisu wiadomości dało wynik negatywny. ([AgreementRef]: -, [SenderPartyId]: PartyIdB, [ReceiverPartyI...
22	04.05.2022 20:15:45	[-, CompareReceivedCerts_BinarySecurityToken]: Niezgodność certyfikatu podpisu przesłanego w wiadomości z ustalonym w [P-Mode]. ([Agreement]: Tes...
23	04.05.2022 20:14:46	[-, GetResponse]: Serwer zdalny zwrócił błąd: (400) Złe żądanie. StackTrace: w System.Net.HttpWebRequest.GetResponse() w TelCOMM.Models.AS4.Pu...
24	04.05.2022 20:14:46	[-, SecurityDecryptAndVerifySign]: Sprawdzenie podpisu wiadomości dało wynik negatywny. ([AgreementRef]: -, [SenderPartyId]: PartyIdB, [ReceiverPartyI...
25	04.05.2022 20:14:45	[-, CompareReceivedCerts_BinarySecurityToken]: Niezgodność certyfikatu podpisu przesłanego w wiadomości z ustalonym w [P-Mode]. ([Agreement]: Tes...
26	04.05.2022 20:13:46	[-, GetResponse]: Serwer zdalny zwrócił błąd: (400) Złe żądanie. StackTrace: w System.Net.HttpWebRequest.GetResponse() w TelCOMM.Models.AS4.Pu...
27	04.05.2022 20:13:46	[-, SecurityDecryptAndVerifySign]: Sprawdzenie podpisu wiadomości dało wynik negatywny. ([AgreementRef]: -, [SenderPartyId]: PartyIdB, [ReceiverPartyI...
28	04.05.2022 20:13:45	[-, CompareReceivedCerts_BinarySecurityToken]: Niezgodność certyfikatu podpisu przesłanego w wiadomości z ustalonym w [P-Mode]. ([Agreement]: Tes...

Rysunek 13. Ekran „Logi”

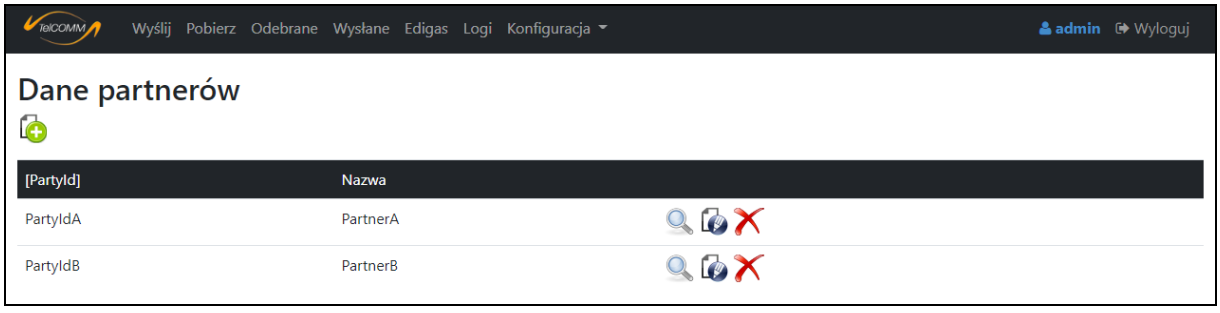
Z ekranu „Logi” można przejść na dodatkowy ekran „Komunikaty”, na którym istnieje możliwość podejrzenia całych żądań i odpowiedzi HTTP, które były wysyłane i odbierane przez aplikację. Ekran służy szczegółowej analizie surowych komunikatów AS4 i AS2.



Rysunek 14. Ekran „Komunikaty”

EKRAN „PARTNERZY”

Na ekranie „Konfiguracja->Partnerzy” istnieje możliwość powiązania wartości [PartyId] z przyjazną nazwą, która będzie wyświetlana w obrębie aplikacji zamiast identyfikatora partnera. W komunikacji Edig@s dla AS4 identyfikatorem tym jest kod EIC.

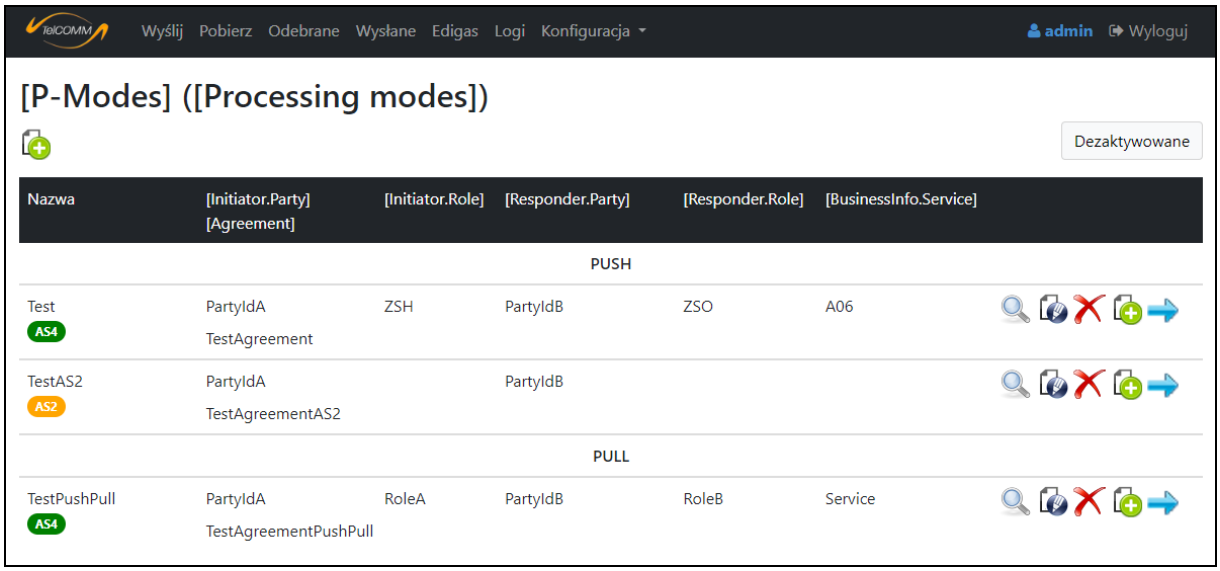


Rysunek 15. Ekran „Partnerzy”

EKRAN „[P-MODES]”

Ekran „Konfiguracja->[P-Modes]” ([Processing modes]) służy definiowaniu wszystkich informacji związanych z połączeniem między partnerami komunikacji. Dane te są ściśle związane z protokołem AS4 i jest to kluczowy ekran konfiguracyjny w aplikacji. [P-Modes] podzielone są na dwie grupy: PUSH – dotyczy wysyłania, PULL – dotyczy pobierania. W aplikacji można również wykorzystać [P-Mode] do zdefiniowania połączenia AS2. Istnieje także możliwość przejścia na ekran [P-Mode] wyłączonych z użycia w aplikacji klikając przycisk Dezaktywowane ([P-Mode] na

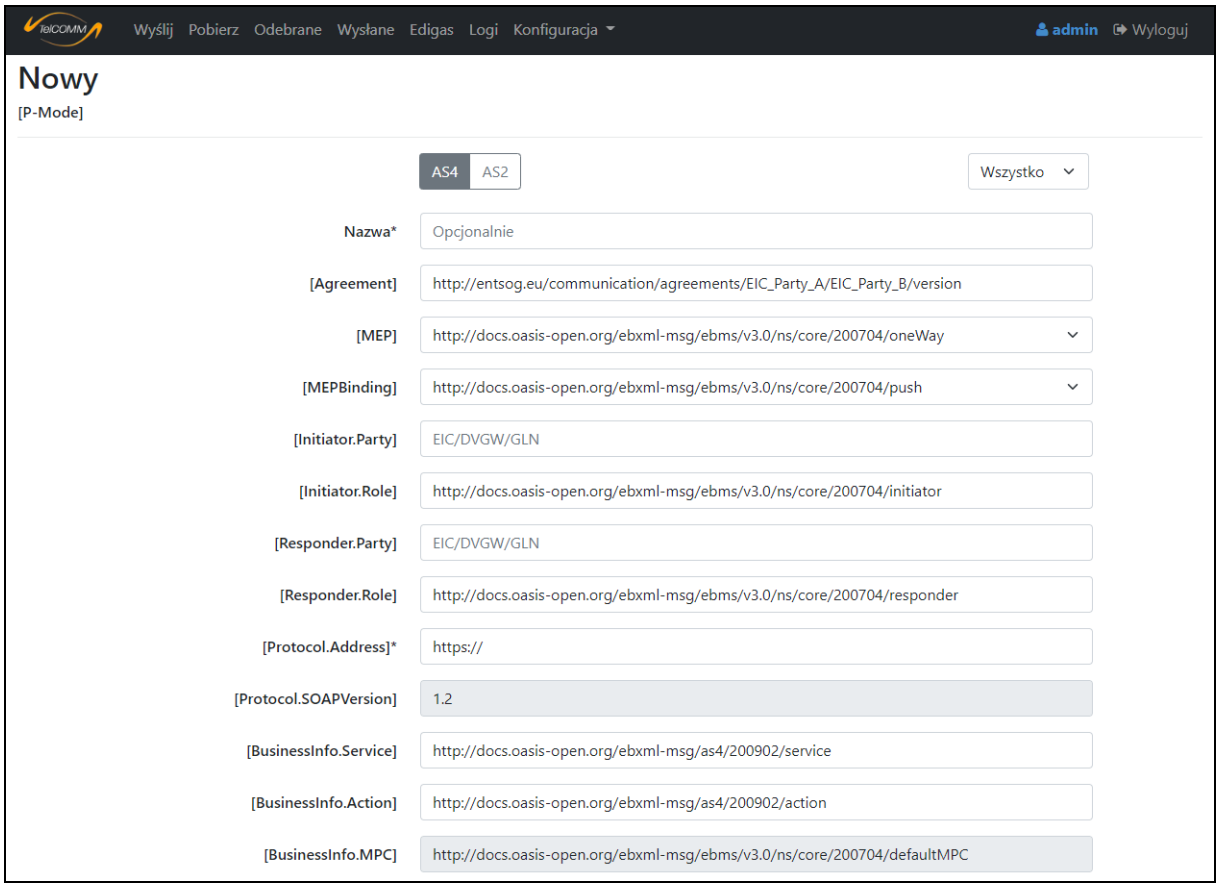
ekranie usuwania można trwale usunąć z bazy danych lub jedynie dezaktywować, z ewentualną możliwością późniejszego przywrócenia do działania).



Rysunek 16. Ekran „[P-Modes]"

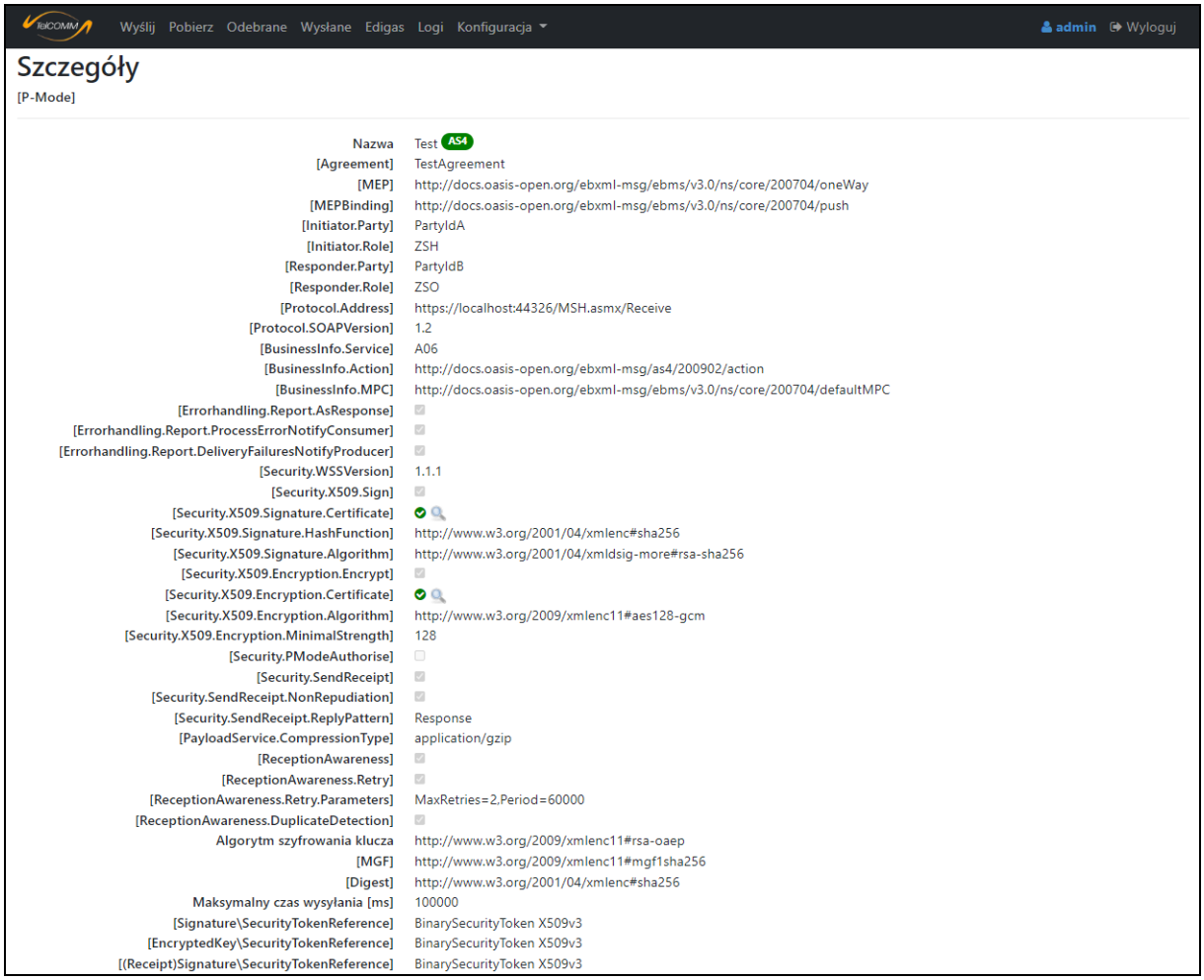
Jedną z informacji zawartych w [P-Mode] są certyfikaty partnerów komunikacji. Importowanie certyfikatu odbywa się na ekranie tworzenia i edycji [P-Mode]. Może się ono odbyć na dwa sposoby: poprzez podanie przyjaznej nazwy lub części pola podmiot certyfikatu zainstalowanego wcześniej w magazynie certyfikatów Windows albo ewentualnie poprzez wczytanie pliku certyfikatu wraz z podaniem hasła do pliku, skutkującego zapisem w bazie danych aplikacji. [P-Mode] odzwierciedla komunikację w jednym kierunku, dlatego jeśli komunikacja ma przebiegać w obydwu kierunkach, teoretycznie należałoby zdefiniować dwa [P-Mode], jednak przez to, że w praktyce używany jest ten sam certyfikat przez partnerów do podpisu i szyfrowania, wystarczy zdefiniować [P-Mode] w jednym kierunku. W przypadku definiowania [P-Mode] w obu kierunkach w [P-Mode] służącym do wysyłania certyfikat podpisu musi posiadać klucz prywatny, po to aby można było nim podpisać wysyłaną wiadomość, natomiast w [P-Mode] służącym do odbierania klucz prywatny musi być zawarty w certyfikacie szyfrowania, aby odszyfrować wiadomość odebraną. Stąd konieczność podania hasła w przypadku wczytywania, ponieważ plik certyfikatu posiadający klucz prywatny zostaje zabezpieczony hasłem. W przypadku definiowania jednego [P-Mode] dla obu kierunków lepiej zdefiniować [P-Mode] wysyłający, ewentualnie można odbierający.
Na ekranie tworzenia nowego [P-Mode] dla AS4 niektóre wartości są odgórnie ustalone w oparciu o dokumentację protokołu, dla wielu pozostałych pojawiają się wartości domyślne, które mogą zostać zmienione na inne. Ważnym polem jest [Agreement], którego wartość staje się identyfikatorem umowy w komunikacji między partnerami. Pola [MEP] i [MEPBinding] służą określeniu wzorca komunikacji, w polach [Initiator.Party] i [Responder.Party] należy podać identyfikator partnera – w komunikacji Edig@s jest nim kod EIC, a [Protocol.Address] odpowiada adresowi URL na jaki wysyłane są dokumenty.
W komunikacji AS2 pola [Initiator.Party] i [Responder.Party] zostają zamienione na [AS2-From] i [AS2-To], a pole [Agreement] używane jest jedynie wewnętrznie w aplikacji.
Jest również kilka pól dodatkowych, niezawartych w dokumentacji, jak np. Nazwa, Algorytm szyfrowania klucza (dla AS4) czy Certyfikat TLS serwera bądź klienta. Nazwa oprócz tego, że pozwala na łatwiejszą identyfikację danego [P-Mode] na ekranach aplikacji, jest używana podczas wysyłania wiadomości przez usługę internetową (Web Service). Algorytm szyfrowania klucza pozwala na wybranie standardowego szyfrowania z użyciem certyfikatu RSA lub szyfrowania z użyciem certyfikatu ECC. Wczytanie certyfikatu TLS serwera nie jest wymagane, ponieważ

domyślnie podczas wysyłania certyfikat znajdujący się po stronie serwera jest sprawdzany czy jest prawidłowy (ma to miejsce gdy adres URL odbiorcy rozpoczyna się od „https://”, a nie „http://”). Można go wczytać gdy jedyną walidację jaką chcemy wykonać to sprawdzenie czy certyfikat po stronie serwera jest identyczny co wczytany (mamy wtedy silniejszą weryfikację, ale nie ma sprawdzenia czy certyfikat jest prawidłowy, czyli np. czy nie wygasł). Załączenie certyfikatu TLS klienta przyda się w niestandardowym przypadku gdy partner zgłosi taką potrzebę.



Rysunek 17. Część ekranu tworzenia nowego [P-Mode]

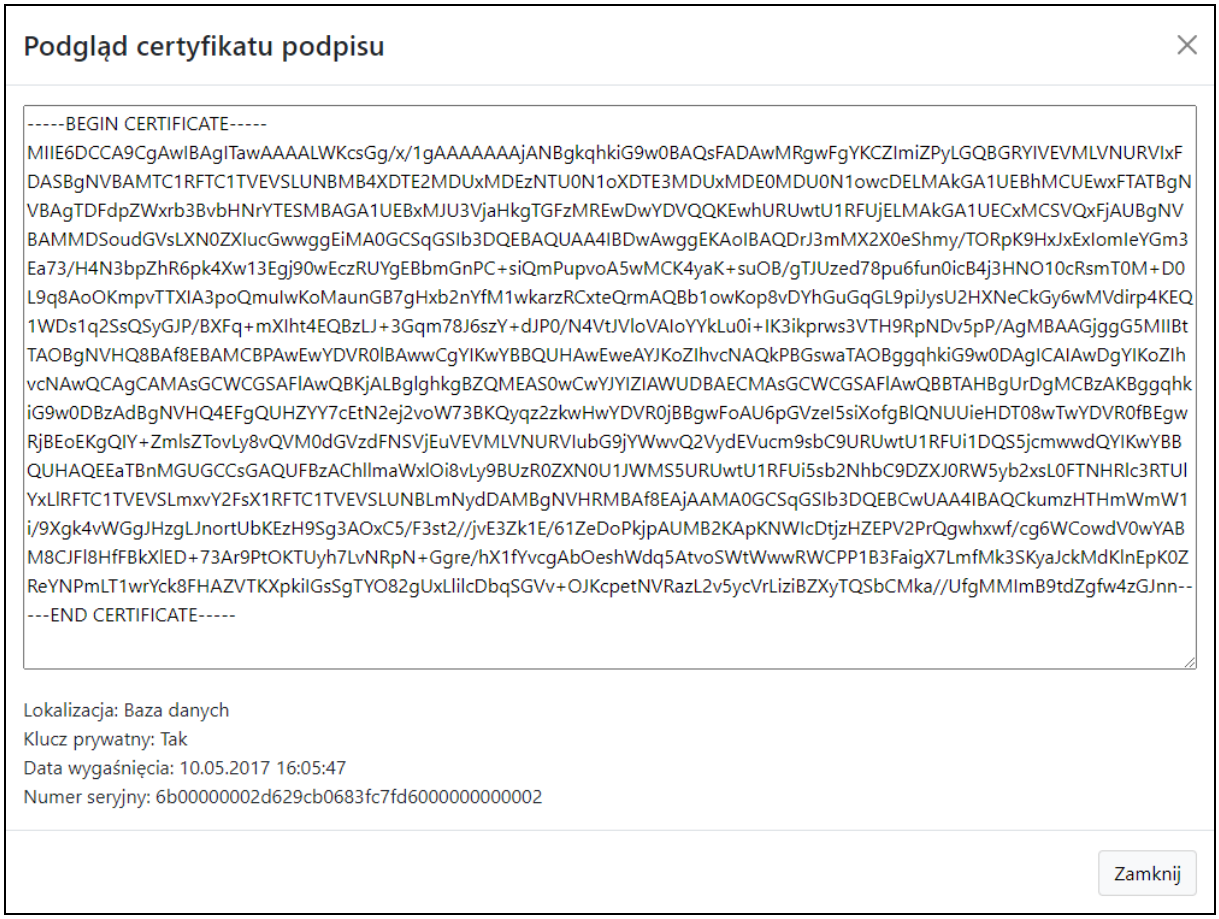
Na ekranie szczegółów istnieje możliwość podglądu właściwości [P-Mode], w tym także podglądu certyfikatów podpisu, szyfrowania i TLS.



Rysunek 18. Ekran szczegółów [P-Mode]

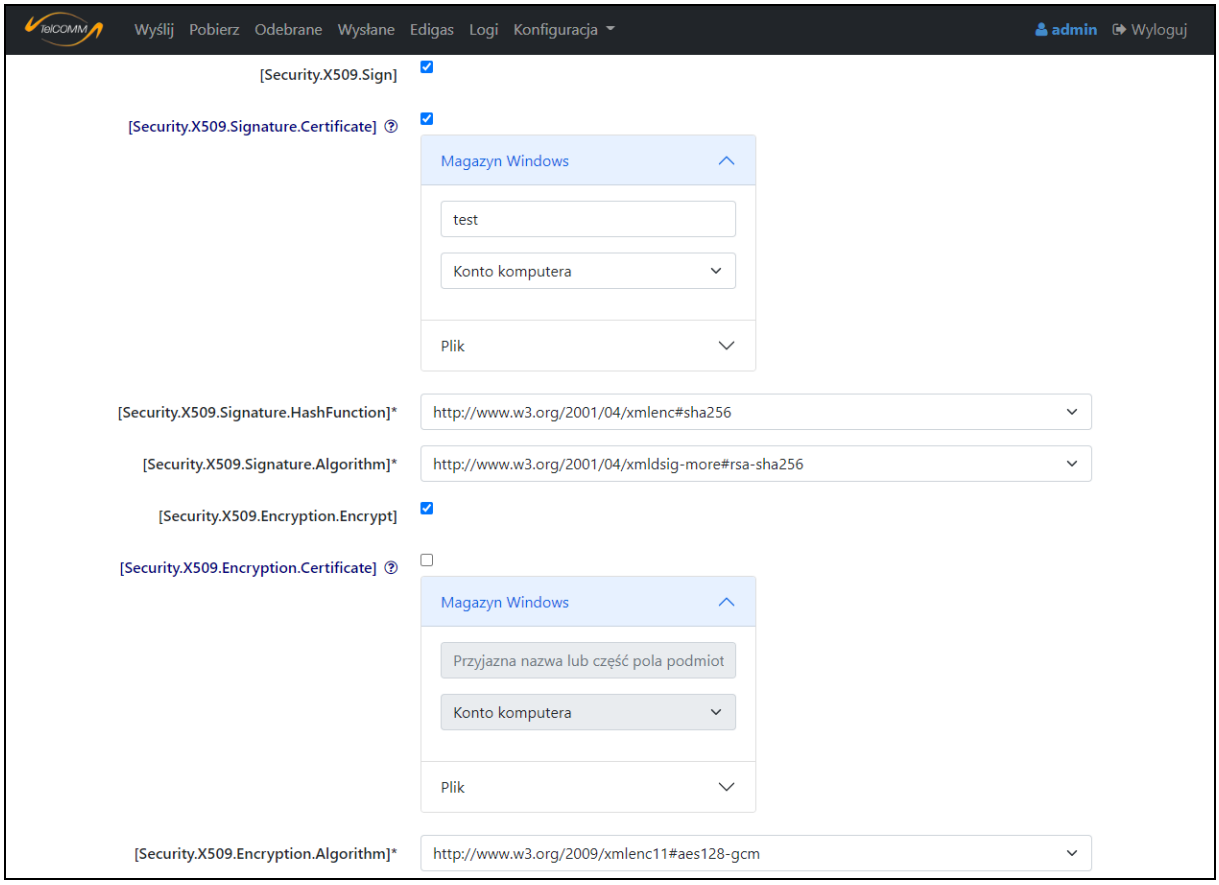
Zawartość okna „Podgląd certyfikatu” przedstawia certyfikat z kluczem publicznym w standardzie X.509 szyfrowany algorytmem Base-64 wraz z informacjami uzupełniającymi (szczegóły na poniższym rysunku). W przypadku włączonej w konfiguracji aplikacji opcji „Walidacja certyfikatów” pojawia się dodatkowa informacja, jeśli walidacja certyfikatu zakończyła się wynikiem negatywnym. Własne certyfikaty należy przekazać partnerowi komunikacji bez klucza prywatnego (!) w postaci tekstowej (zawartość okna) lub pliku (.cer, .pem), aby korzystając z zawartego w certyfikacie klucza publicznego mógł on:

- zweryfikować podpis wysłanej do niego wiadomości,
- zaszyfrować wysyłaną przez niego do nas wiadomość.



Rysunek 19. Podgląd przykładowego certyfikatu podpisu na ekranie szczegółów [P-Mode]

W przypadku edycji [P-Mode] zmiana certyfikatu jest możliwa po zaznaczeniu pola wyboru „Wczytaj” przy polu certyfikatu. Dzięki temu podczas edycji można zachować poprzednie certyfikaty lub zmienić dowolny z nich.

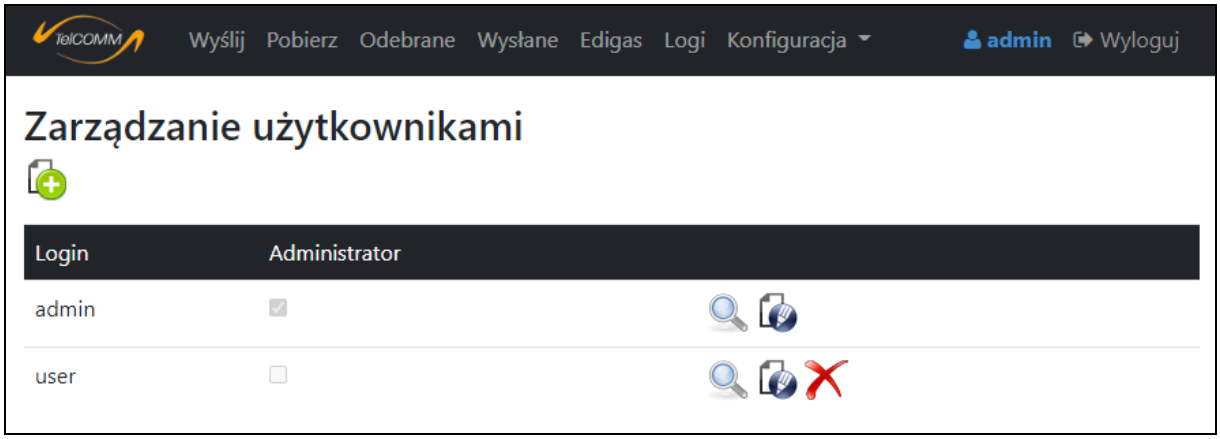


Rysunek 20. Część ekranu edycji [P-Mode]

Edycja certyfikatów w istniejącym [P-Mode] nie jest preferowanym rozwiązaniem – lepiej stworzyć nową wersję dla danego [P-Mode]. Na ogólnym ekranie jak i na ekranie szczegółów istnieje taka możliwość jako jedna z opcji dla [P-Mode] – pojawia się wtedy ekran tworzenia nowej wersji [P-Mode] z wypełnionymi polami aktualnymi wartościami. Korzystanie z takiego podejścia powoduje, że w komunikacji nadal będzie używany aktualny [P-Mode], do czasu gdy przestanie działać – wtedy nastąpi jego dezaktywowanie oraz przełączenie się na korzystanie z nowego [P-Mode] (więcej w rozdziale Automatyczna aktualizacja certyfikatów). Na obu wyżej wspomnianych ekranach istnieje również możliwość wyeksportowania [P-Mode] do XML. Usunąć [P-Mode] można definitywnie lub jedynie dezaktywować tj. [P-Mode] przestaje być używane w aplikacji, ale pozostaje w bazie danych.

EKRAN „UŻYTKOWNICY”

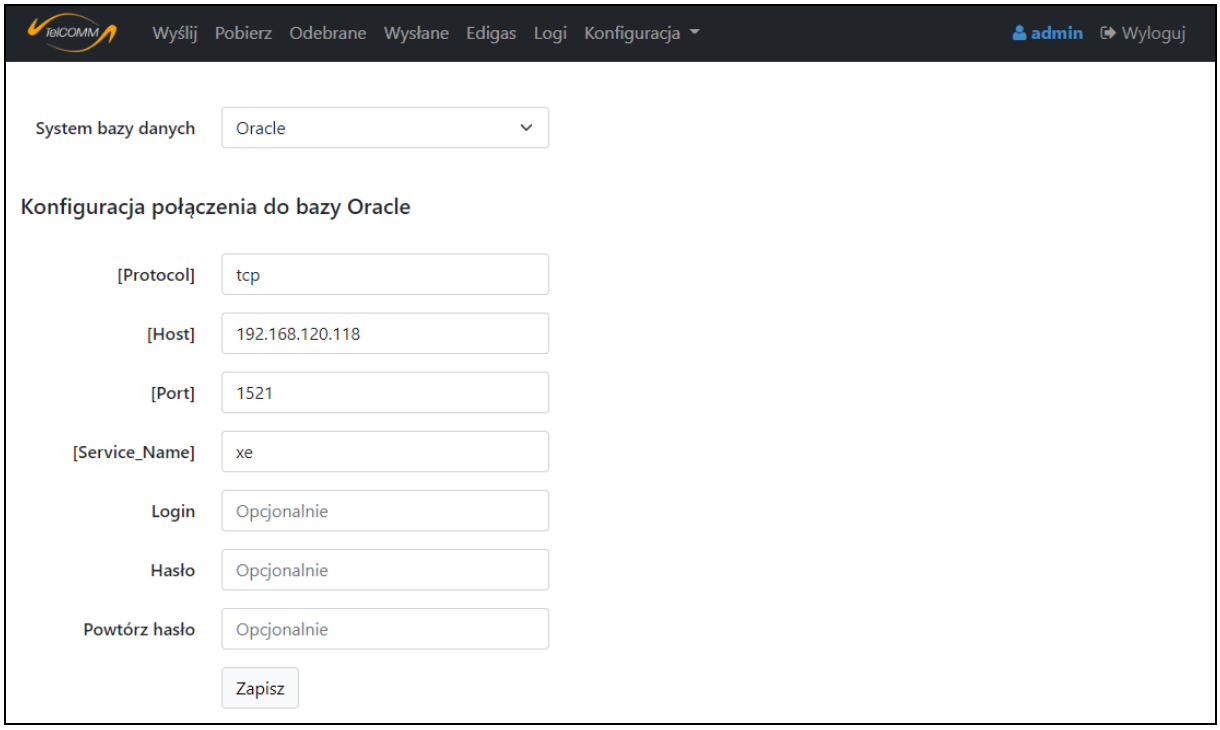
Ekran „Konfiguracja->Użytkownicy” służy do zarządzania użytkownikami. Użytkownik może otrzymać uprawnienia zwykłe lub administratora. W programie istnieje nieusuwalny użytkownik „admin”, którego początkowe hasło również „admin” należy zmienić na ekranie edycji użytkownika po pierwszym logowaniu (i login ew. też).



Rysunek 21. Ekran „Użytkownicy"

EKRAN „BAZA DANYCH"

Na ekranie „Konfiguracja-> Baza danych" istnieje możliwość wyboru systemu bazy danych SQLite lub Oracle i konfiguracji połączenia do bazy w przypadku wyboru systemu Oracle.



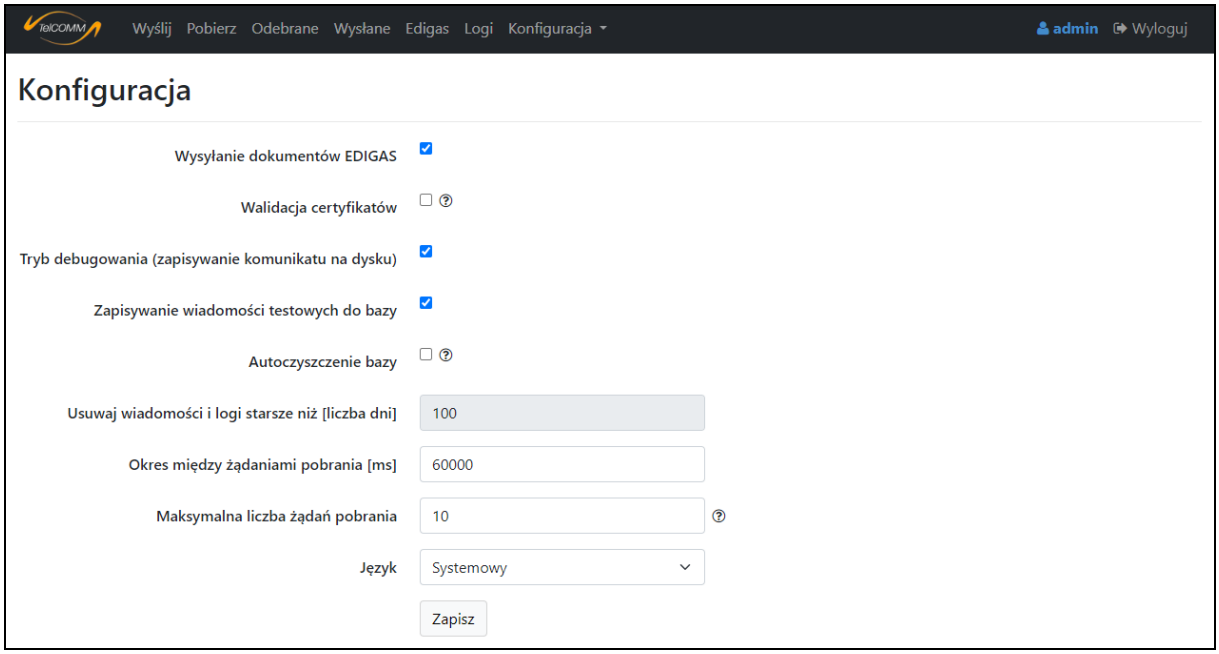
Rysunek 22. Ekran „Baza danych"

EKRAN „OPCJE"

Ekran „Konfiguracja->Opcje" posiada następujące opcje:

- „Wysyłanie dokumentów EDIGAS" – ustawienie określające czy w obrębie aplikacji mają pojawiać się pola odpowiednie dla wysyłania dokumentów Edig@s. W przypadku wybrania tej opcji, na pasku pojawia się zakładka „Edigas" a na ekranach wysyłania wiadomości oraz definiowania [P-Mode] pojawią się elementy związane komunikacją Edig@s.
- „Walidacja certyfikatów" – sprawdzanie poprawności używanych w obrębie aplikacji certyfikatów (również OCSP/CRL, dlatego wymagane jest połączenie z serwerem urzędu certyfikacji) i w przypadku nieprawidłowości blokowanie komunikatu,

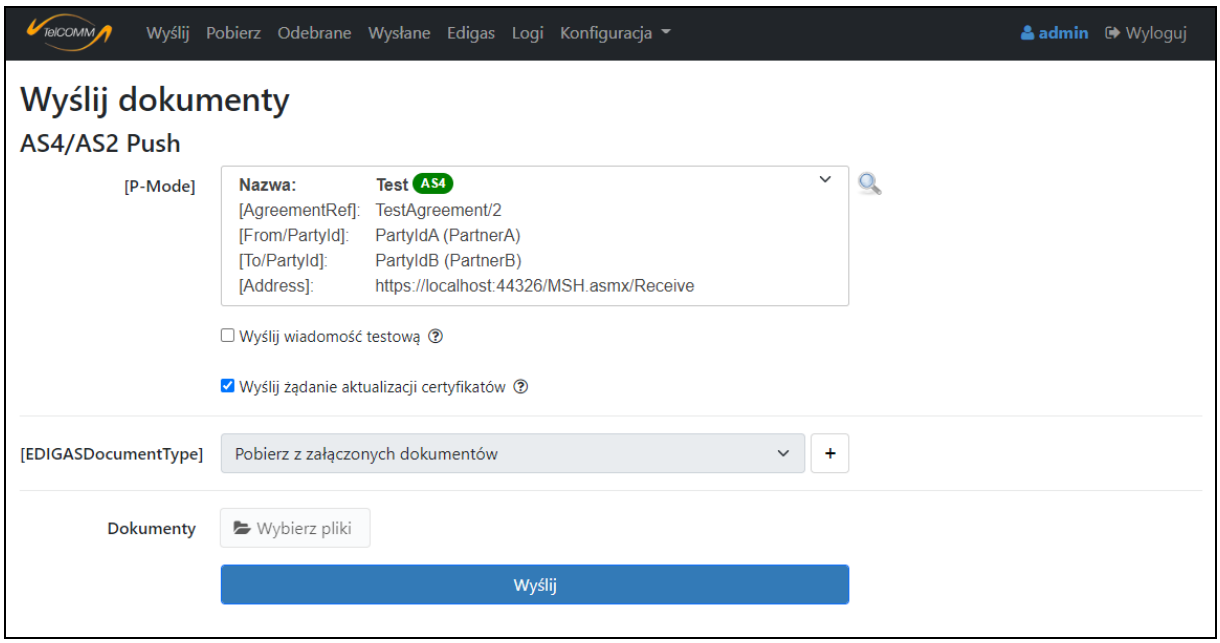
- „Tryb debugowania (zapisywanie komunikatu na dysku)” – aplikacja posiada możliwość pracy w trybie debugowania, który sprowadza się do zapisywania całych żądań i odpowiedzi HTTP wysłanych i odebranych w folderze Debug (przechowywane są komunikaty z ostatnich 100 dni, do podglądu również na ekranie „Komunikaty”).
- „Zapisywanie wiadomości testowych do bazy” – opcja pozwala określić czy wiadomości testowe mają być zapisywane w bazie danych i tym samym widoczne na ekranach „Odebrane” i „Wysłane”.
- „Autoczyszczenie bazy” – opcja ta odpowiada za automatyczne usuwanie z bazy danych starszych wiadomości i wpisów w logu w oparciu o wartość z pola „Usuwaj wiadomości i logi starsze niż [liczba dni]”. Operacja jest wykonywana raz na dobę (dlatego odtwarzanie puli powinno być ustawione nie częściej niż co 24h).
- „Maksymalna liczba żądań pobrania” – parametr ten i następny dotyczą pobierania plików Pull; liczba wskazuje programowi ile razy po wysłaniu żądania ma próbować pobrać plik od partnera.
- „Okres między żądaniami pobrania [ms]” – określa ile czasu ma upłynąć do następnej próby pobrania pliku.
- „Język” – możliwość ustawienia języka polskiego lub angielskiego w obrębie aplikacji.



Rysunek 23. Ekran „Opcje”

Automatyczna aktualizacja certyfikatów

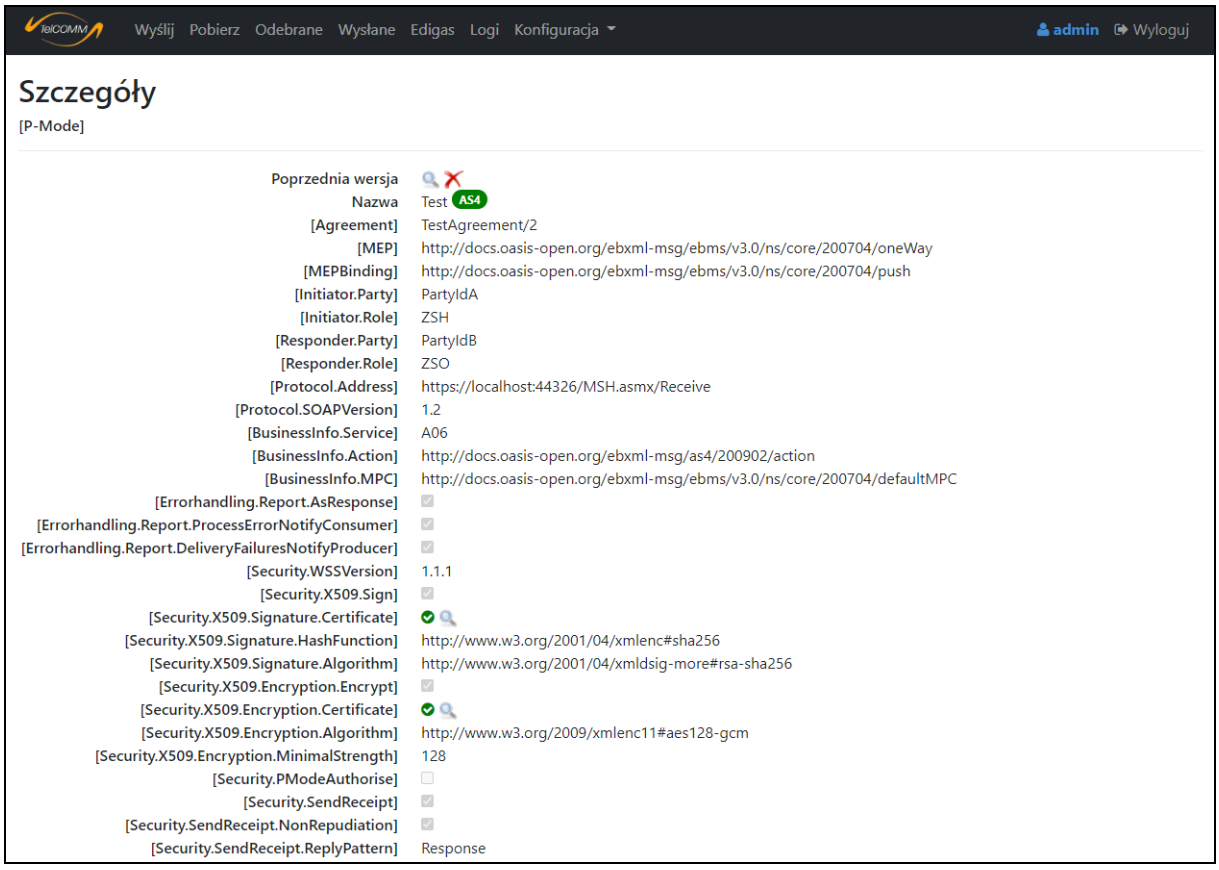
Funkcjonalność ta jest zaimplementowana dla protokołu AS4 i ma na celu zautomatyzowanie procesu aktualizacji certyfikatów podpisu i szyfrowania u partnerów komunikacji, który wynika z ograniczonej ważności certyfikatów. Aby można z niej skorzystać oprogramowanie po obu stronach komunikacji musi wspierać to rozwiązanie. Aplikacja TelCOMM przyjmuje żądania aktualizacji certyfikatów od partnerów (odbiorców), gdy chcą oni poinformować o aktualizacji swojego certyfikatu. Jeśli chodzi o ten kierunek aktualizacji nie są wymagane żadne działania ze strony użytkownika – wszystkie potrzebne operacje wykonywane są w sposób automatyczny, a ich wynik odnotowany w logu (podczas przełączenia na nowy certyfikat pojawi się wpis z błędem dotyczący odbioru, a następnie informacja o zakończeniu używania dotychczasowego [P-Mode]). Sytuacja ma się inaczej gdy aktualizacji wymaga certyfikat partnera (nadawcy) po stronie TelCOMM. Należy wtedy dla [P-Mode], w którym występuje dany certyfikat, dodać nie nowy [P-Mode], a nową wersję dla danego [P-Mode] z nowym certyfikatem. Jeśli [P-Mode] posiadających zmieniany certyfikat jest więcej, należy zaznaczyć przy dodawaniu nowego certyfikatu opcję „aktualizacja dla wszystkich [P-Mode]" – wtedy dla odpowiednich [P-Mode] zostanie również utworzona nowa wersja z nowym certyfikatem. Przy selekcji odpowiednich [P-Mode] brany jest pod uwagę jedynie certyfikat tego samego rodzaju np. podpisu, dlatego jeśli certyfikat, który ma zostać zmieniony jest używany także jako np. szyfrowania w [P-Mode] w kierunku odwrotnym, przy definiowaniu [P-Mode] w obu kierunkach, to również należy dla tego [P-Mode] dodać nową wersję i skorzystać z opcji „aktualizacja dla wszystkich [P-Mode]". Teoretycznie w nowej wersji [P-Mode] powinna pojawić się nowa wartość dla [AgreementRef], ale jest dopuszczone zachowanie aktualnej wartości, chociaż wiąże się z tym ograniczenie. Na ekranie wysyłania (lub pobierania) dla każdej nowej wersji [P-Mode] (pod warunkiem, że w nowej wersji jest nowa wartość [AgreementRef]) pojawi się opcja „Wyślij żądanie aktualizacji certyfikatu", którą należy zaznaczyć i wysłać komunikat.



Rysunek 24. Ekran „Wyślij" – wysyłanie żądania aktualizacji certyfikatu

Wysłanie żądania aktualizacji certyfikatu jest specjalnie spreparowanym komunikatem AS4 działającym na tych samych zasadach co pozostałe komunikaty AS4. Po wysłaniu reszta procesu polegająca na dezaktywowaniu aktualnej wersji [P-Mode] i aktywowaniu nowej odbywa się automatycznie w momencie gdy partner zaktualizuje konfigurację po swojej stronie, a szczegóły tej operacji zostają zapisane w logu (podczas przełączenia na nowy certyfikat pojawi się wpis z błędem dotyczący wysłania, a następnie informacja o zakończeniu używania dotychczasowego [P-Mode]).

W przypadku gdy chcemy już w danym momencie rozpocząć korzystanie z nowej wersji [P-Mode] należy na górze ekranu szczegółów dla nowej wersji [P-Mode] dezaktywować poprzednią wersję. Wtedy dla danego [P-Mode] będzie używany od tego momentu nowy certyfikat.



Rysunek 25. Ekran szczegółów [P-Mode] – dezaktywowanie poprzedniej wersji

Interfejs do współpracy z aplikacją zewnętrzną

Aplikacja TelCOMM może spełniać zadanie bramki do komunikacji B2B, przez którą inna aplikacja może automatycznie wysyłać i odbierać wiadomości kanałem AS4 lub AS2, ponieważ udostępnia odpowiednie metody w usłudze internetowej /WebServices/**Gateway.asmx**. Aby móc korzystać z tych metod należy najpierw zalogować się w aplikacji TelCOMM. Można to uczynić korzystając z metod HTTP:

a) /Login/LogOnExternal – metoda typu POST służąca do logowania aplikacji zewnętrznej w aplikacji TelCOMM. Login i hasło należy przekazać w parametrach „login" i „password". W przypadku poprawnego zalogowania w odpowiedzi HTTP umieszczone zostaną ciastka (cookie), które należy załączyć do ciastek żądania HTTP adresowanego do metody z Web Service Gateway.asmx,

b) /Login/LogOffExternal – metoda typu GET wylogowująca aplikację zewnętrzną z aplikacji TelCOMM.

lub bezpośrednio w Web Service Gateway.asmx, za pomocą analogicznych metod:

a) LogOn

b) LogOff

W Web Service **Gateway.asmx** poza tym udostępniono metody, z których najważniejsze to:

a) SendByPModeName – metoda służąca do wysyłania za pomocą protokołu AS4, jako argumenty przyjmuje pliki (nazwa pliku i binarna treść) oraz nazwę zdefiniowanego [P-Mode]. Poza tym ewentualnie podawana jest informacja o typie wysyłanego dokumentu Edig@s i o wysyłaniu plików w osobnych wiadomościach, a zwracany jest obiekt z wynikiem analizy odpowiedzi od odbiorcy; SendAS2ByPModeName – analogiczna metoda dla protokołu AS2,

b) GetSendOptions – metoda pomocnicza zwracająca możliwe wartości dla niektórych parametrów, z których można skorzystać w powyższych metodach,

c) GetUnreadMessagesIds – zwraca listę wartości tekstowych „MessageId" wiadomości odebranych, które nie są oznaczone jako przeczytane; GetUnreadPullMessagesIds i GetUnreadPushMessagesIds – jw. z tą różnicą, że wyniki dotyczą odpowiedniego wzorca komunikacji,

d) GetMessage – pobiera szczegóły wiadomości odebranej podając jako argument „MessageId"; GetMessages – pobiera szczegóły wielu wiadomości odebranych podając jako argument listę wartości „MessageId"; GetMessageSent - pobiera szczegóły wiadomości wysłanej,

e) MarkAsRead - oznacza wiadomości jako przeczytane, przyjmując jako argument listę wartości „MessageId",

f) AddPullRequestByPModeName – bezpośrednie pobranie korzystając z wzorca One-Way/Pull,

g) AddPullResponseByReceivedMessageId – metoda wykorzystywana przy udostępnianiu danych z użyciem wzorca Two-Way/Push-Pull.

Poniższa tabela przedstawia jakich metod usługi internetowej Gateway.asmx należy użyć aby wysłać do lub odebrać od partnera komunikat dla danego wzorca komunikacji AS4 przez aplikację zewnętrzną:

Wzorzec komunikacji AS4	Kierunek	Metody Web Service Gateway.asmx
One-Way/Push	Wysłanie	1. SendByPModeName lub SendAS2ByPModeName
	Odebranie	1. GetUnreadPushMessagesIds 2. GetMessage lub GetMessages 3. MarkAsRead
One-Way/Pull	Wysłanie	1. AddPullRequestByPModeName
Two-Way/Push-Pull	Wysłanie	1. SendByPModeName
	Odebranie	1. GetUnreadPullMessagesIds 2. GetMessage lub GetMessages 3. MarkAsRead 4. AddPullResponseByReceivedMessageId (w przypadku udostępniania danych)

Tab. 1. Metody Web Service Gateway używane przy wysyłaniu i odbieraniu dla danego wzorca komunikacji

Na następnym rysunku przedstawiony został przykład synchronicznego wykorzystania metod Web Service Gateway.asmx w środowisku .NET Framework w języku C#. W podanym przykładzie skorzystano z wcześniej dodanego Web Reference o nazwie „localhost".

```
localhost.Gateway gate = new localhost.Gateway
{
    CookieContainer = new CookieContainer()
};

string logOnResult = gate.LogOn("admin", "admin");

string[] unreadMessagesIds = gate.GetUnreadPushMessagesIds();

var receivedMessages = gate.GetMessages(unreadMessagesIds);

bool markAsReadResult = gate.MarkAsRead(unreadMessagesIds);

gate.LogOff();
```

Rysunek 26. Przykład wykorzystania metod Web Service Gateway [.NET Framework C#, logowanie w Web Service, Web Reference, synchronicznie]

Kolejny przykład w .NET Framework C#, tym razem zalogowania do aplikacji z użyciem metody HTTP, przekazania ciastka i synchronicznego wykorzystania metody z Web Service **Gateway.asmx** z użyciem Web Reference przedstawiony został poniżej. Natomiast na następnym rysunku obecny jest przykład wysłania komunikatu AS4 w oparciu o zdefiniowany [P-Mode] o nazwie „nomint". Wysyłane pliki zostały przekazane w argumencie „files" funkcji „SendByPModeName", który jest listą obiektów o dwóch właściwościach: dane binarne i nazwa pliku. Dodatkowo określony zostaje

typ wysyłanego dokumentu Edig@s i każdy plik zostaje wysłany w osobnej wiadomości (opcja ta jest brana pod uwagę przy wysyłaniu więcej niż jednego pliku).

```
public void Example()
{
    CookieCollection cookieCollection = Login("https://localhost:44326/Login/LogOnExternal", "login=admin&password=admin");

    localhost.Gateway gateway = new localhost.Gateway();
    gateway.CookieContainer = new CookieContainer();
    gateway.CookieContainer.Add(cookieCollection);
    string[] unreadMessagesIds = gateway.GetUnreadPushMessagesIds();

    Logoff("https://localhost:44326/Login/LogOffExternal", cookieCollection);
}

1 reference | MSiatkowski, 57 days ago | 1 author, 2 changes
public CookieCollection Login(string uri, string parameters)
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
    request.ContentType = "application/x-www-form-urlencoded";
    request.Method = "POST";
    byte[] bytes = System.Text.Encoding.ASCII.GetBytes(parameters);
    request.ContentLength = bytes.Length;
    request.CookieContainer = new CookieContainer();
    using (System.IO.Stream requestStream = request.GetRequestStream())
    {
        requestStream.Write(bytes, 0, bytes.Length);
        using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
        {
            return response.Cookies;
        }
    }
}

1 reference | MSiatkowski, 57 days ago | 1 author, 2 changes
public void Logoff(string uri, CookieCollection cookieCollection)
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
    request.Method = "GET";
    request.CookieContainer = new CookieContainer();
    request.CookieContainer.Add(cookieCollection);
    using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
    {
    }
}
```

Rysunek 27. Przykład wykorzystania metody Web Service Gateway [.NET Framework C#, logowanie HTTP, Web Reference, synchronicznie]

```
localhost.Gateway gateway = new localhost.Gateway();
gateway.CookieContainer = new CookieContainer();
gateway.CookieContainer.Add(cookieCollection);

localhost.AS4SendModelPModeName sendModelPModeName = new localhost.AS4SendModelPModeName()
{
    PModeName = "nomint",
    EDIGASDocumentTypeCode = "01G",
    AttachmentsApart = true
};

localhost.SendResult result = gateway.SendByPModeName(files, sendModelPModeName);
```

Rysunek 28. Przykład wysłania komunikatu AS4 [.NET Framework C#, logowanie HTTP, Web Reference, synchronicznie]

Analogiczny przykład w .NET C# z użyciem Service Reference i metod asynchronicznych dla wysyłania i odbierania:

```
2 references | MSiatkowski, 89 days ago | 1 author, 1 change
public async Task<SendResult> SendData(List<FileModel> files, AS4SendModelPModeName sendModel)
{
    GatewaySoapClient gate = await PrepareSoapClient();

    SendResult sendResult = await gate.SendByPModeNameAsync([.. files], sendModel);

    return sendResult;
}

2 references | MSiatkowski, 89 days ago | 1 author, 1 change
public async Task<List<MessageModelReceived>> ReadData()
{
    GatewaySoapClient gate = await PrepareSoapClient();

    string[] unreadMessagesIds = await gate.GetUnreadPushMessagesIdsAsync();

    MessageModelReceived[] receivedMessages = await gate.GetMessagesAsync(unreadMessagesIds);

    bool markAsReadResult = await gate.MarkAsReadAsync(unreadMessagesIds);

    return [.. receivedMessages];
}

2 references | MSiatkowski, 89 days ago | 1 author, 1 change
private async Task<GatewaySoapClient> PrepareSoapClient()
{
    IEnumerable<Cookie> cookies = await Login(as4Options);

    var binding = new BasicHttpsBinding
    {
        AllowCookies = true
    };

    var address = new EndpointAddress(as4Options.Url);

    var gate = new GatewaySoapClient(binding, address);

    var cookieManager = gate.InnerChannel.GetProperty<IHttpCookieContainerManager>();
    foreach (Cookie cookie in cookies)
        cookieManager.CookieContainer.Add(cookie);

    return gate;
}
```

Rysunek 29. Wysyłanie i odbieranie [.NET C#, logowanie HTTP, Service Reference, asynchronicznie] – część 1

```
1 reference | MSiatkowski, 89 days ago | 1 author, 3 changes
private static async Task<IEnumerable<Cookie>> Login(AS4Options as4Options)
{
    var cookies = new CookieContainer();
    var handler = new HttpClientHandler
    {
        CookieContainer = cookies
    };

    using var httpClient = new HttpClient(handler);

    string loginUrl = GetGateUrl(as4Options.Url, "Login", "LogOnExternal");

    var data = new Dictionary<string, string>
    {
        {"login", as4Options.User},
        {"password", as4Options.Pass}
    };

    var res = await httpClient.PostAsync(loginUrl, new FormUrlEncodedContent(data));

    IEnumerable<Cookie> responseCookies = cookies.GetCookies(new Uri(loginUrl)).Cast<Cookie>();

    return responseCookies;
}

1 reference | MSiatkowski, 89 days ago | 1 author, 2 changes
private static string GetGateUrl(string wsUrl, string controller, string action)
{
    string gatewayUrl = wsUrl;
    int index = gatewayUrl.IndexOf("webservice", 0, gatewayUrl.Length, StringComparison.OrdinalIgnoreCase);
    if (index > 0)
    {
        gatewayUrl = gatewayUrl.Remove(index);
        gatewayUrl = gatewayUrl + controller + "/" + action;
    }
    return gatewayUrl;
}

5 references | MSiatkowski, 94 days ago | 1 author, 1 change
public class AS4Options
{
    2 references | MSiatkowski, 94 days ago | 1 author, 1 change
    public string Url { get; set; } = string.Empty;
    4 references | MSiatkowski, 94 days ago | 1 author, 1 change
    public string User { get; set; } = string.Empty;
    4 references | MSiatkowski, 94 days ago | 1 author, 1 change
    public string Pass { get; set; } = string.Empty;
}
```

Rysunek 30. Wysyłanie i odbieranie [.NET C#, logowanie HTTP, Service Reference, asynchronicznie] – część 2

Usługa umożliwiająca odbiór dokumentów od partnera

Integralną częścią aplikacji TelCOMM jest usługa do odbierania komunikatów AS4 i AS2. Służy do tego metoda **Receive** w Web Service **MSH.asmx**. Adres względny odbierania to: **/MSH.asmx/Receive.**

Po odebraniu wiadomości wysyłana jest odpowiedź (synchronicznie lub w osobnym komunikacie) spełniająca wymaganie niezaprzeczalności odbioru (pod warunkiem, że wiadomość była podpisana i aplikacja dysponuje certyfikatem podpisu dla nadawcy odpowiedzi) lub z potwierdzeniem odbioru. Moduł odbierania wyposażony jest również w funkcjonalność wykrywania duplikatów wiadomości (duplicate detection) oraz obsługę błędów (error handling).

TEL-STER Sp. z o.o.
ul. Stefana Stefańskiego 23
62-002 Suchy Las
Tel. +48 61 628 97 50
Fax. +48 61 639 37 11

Udostępnianie danych

W przypadku korzystania z aplikacji TelCOMM w celu udostępniania danych za pomocą wzorca komunikacji **Two-Way/Push-Pull** analiza żądań o dane i zwracanie wyników do aplikacji odbywa się poza aplikacją TelCOMM z wykorzystaniem odpowiednich metod Web Service Gateway.asmx podanych w tabeli 1.

Załączniki

TelCOMM – Wymagania i architektura

TelCOMM - Procedura instalacji

TelCOMM – Konfiguracja AS4 – Szablon 1

TelCOMM – Konfiguracja AS4 – Szablon 2

Materiały źródłowe

Aplikacja TelCOMM została wykonana w oparciu o poniższe dokumenty, w których znajdują się szczegółowe informacje na temat komunikacji z użyciem protokołów AS4 i AS2.

[ENTSOG] AS4 Usage profile

[OASIS] AS4 Profile of ebMS 3.0 Version 1.0

[OASIS] ebCore Agreement Update Specification Version 1.0

[GAZ-SYSTEM] Instrukcja w zakresie wymiany danych protokołem AS4

[GAZ-SYSTEM] Techniczny opis rozwiązania dla wymiany komunikatów Edig@s z wykorzystaniem standardu AS4

[GAZ-SYSTEM] Techniczny opis rozwiązania dla udostępniania danych pomiarowych i zagregowanych z wykorzystaniem standardu AS4

[IETF] AS2 Specification

[IETF] AS2 Compression