|  |
| --- |
| PUCONNECTOR |
| PUConnector |
| Profesjonalna integracja z operatorem płatności internetowych PayU |

|  |
| --- |
| Tomasz Leszczyński  leszczynski@fplorer.com  https://www.linkedin.com/in/tleszczynski |

Spis treści

[1. Wprowadzenie i przegląd komponentów 2](#_Toc419865999)

[2. Dlaczego WCF ? 5](#_Toc419866000)

[3. Aktywne lub pasywne powiadamianie o zmianach stanu transakcji 6](#_Toc419866001)

[4. PUConnector Toolkit – zestaw aplikacji do integracji 8](#_Toc419866002)

[4.1. PUConnector.Api.Cmd 8](#_Toc419866003)

[4.2. PUConnector.Notifications.Service 10](#_Toc419866004)

[4.3. PUConnector.Notifications.Host 11](#_Toc419866005)

[4.4. PUConnector.SoapProxy.Service 12](#_Toc419866006)

[4.5. PUConnector.SoapProxy.Host 13](#_Toc419866007)

[4.6. PUConnector.Api.UI czyli PUConnector Studio 15](#_Toc419866008)

[5. PUConnector Libraries – biblioteki integracyjne dla platformy .NET 17](#_Toc419866009)

[6. Konfiguracja 19](#_Toc419866010)

[6.1. Parametry konfiguracyjne aplikacji i usług 19](#_Toc419866011)

[6.2. Konfiguracja sieci i zabezpieczeń oraz protokołu HTTPS (SSL) 26](#_Toc419866012)

[6.3. Konfiguracja bazy danych 28](#_Toc419866013)

[6.4. Konfiguracja śledzenia (tracing) oraz dziennika logów EventLog 29](#_Toc419866014)

[6.5. Instalacja usług Notifications i SoapProxy na serwerze IIS 30](#_Toc419866015)

[7. Model danych API 33](#_Toc419866016)

[7.1. OrderId i ExtOrderId, RefundId i ExtRefundId 33](#_Toc419866017)

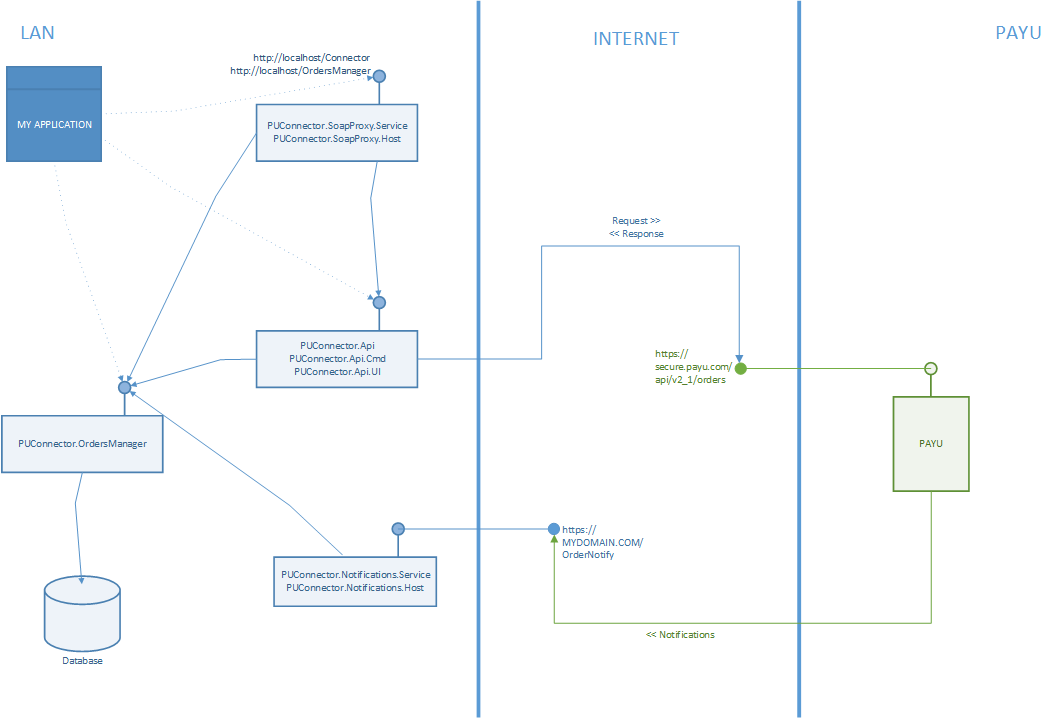
[7.2. PUConnector Toolkit 33](#_Toc419866018)

[7.3. PUConnector Libraries 34](#_Toc419866019)

## Wprowadzenie i przegląd komponentów

PUConnector to pakiet aplikacji oraz bibliotek programistycznych dla platformy .NET umożliwiających szybką, bezpieczną i profesjonalną integrację z wiodącym operatorem płatności internetowych PayU.

Pakiet PUConnector dzieli się na dwie części:

1. **PUConnector Toolkit** – zestaw aplikacji do integracji z PayU.  
     
   PUConnector Toolkit składa się z:  
     
   - **PUConnector.Api.Cmd** – aplikacja konsolowa uruchamiana z systemowego wiersza poleceń. Aplikacja umożliwia wysyłanie żądań (utworzenia nowego zamówienia, anulowania zamówienia, itd.) i odbieranie odpowiedzi na owe żądania od systemu PayU. Żądania jak i odpowiedzi przesyłane są w formacie JSON. Struktura żądań i odpowiedzi opisana została w punkcie [7 – Model danych API](#_Model_danych_API). Aplikacja udostępnia funkcjonalność biblioteki PUConnector.Api.  
     
   - **PUConnector.Notifications.Service** – usługa (WCF) web (hostowana na serwerze web IIS) umożliwiająca odbiór od PayU powiadomień o zmianach stanów/statusów zamówień.  
     
   - **PUConnector.Notifications.Host** – usługa Windows Service umożliwiająca odbiór od PayU powiadomień o zmianach stanów/statusów zamówień. PUConnector.Notifications.Host stanowi alternatywę dla PUConnector.Notifications.Service kiedy nie możemy bądź nie chcemy używać serwera IIS. Aplikacja stanowi środowisko uruchomieniowe dla biblioteki PUConnector.Notifications.Service.  
     
   - **PUConnector.SoapProxy.Service** – usługa (WCF) WebService (hostowana na serwerze web IIS) umożliwiająca korzystanie z całej funkcjonalności PUConnector (a zarazem PayU API) za pomocą protokołu SOAP 1.1 / SOAP 1.2. SoapProxy udostępnia za pomocą usługi WebServices całą funkcjonalność bibliotek PUConnector.Api (wysyłanie żądań i odbieranie odpowiedzi od PayU) oraz PUConnector.OrdersManager (rejestrowanie w bazie danych żądań, odpowiedzi i całej komunikacji z PayU).  
     
   - **PUConnector.SoapProxy.Host** – usługa Windows Service umożliwiająca korzystanie z całej funkcjonalności PUConnectora (a zarazem PayU API) za pomocą protokołu SOAP 1.1 / SOAP 1.2. PUConnector.SoapProxy.Host stanowi alternatywę dla PUConnector.SoapProxy.Service kiedy nie możemy bądź nie chcemy używać serwera IIS. Aplikacja stanowi środowisko uruchomieniowe dla biblioteki PUConnector.SoapProxy.Service.  
     
   - **PUConnector.Api.UI** – aplikacja Windows z graficznym interfejsem użytkownika pozwalająca w prosty i wygodny sposób testować, analizować i optymalizować komunikację z PayU.
2. **PUConnector Libraries** – zestaw bibliotek programistycznych dla platformy .NET umożliwiających integrację z PayU.  
     
   PUConnector Libraries składa się z:  
     
   - **PUConnector.Api** – główna biblioteka implementująca komunikację z PayU. Umożliwia wysyłanie żądań (utworzenia nowego zamówienia, anulowania zamówienia, itd.) i odbieranie odpowiedzi na owe żądania od systemu PayU. Żądania jak i odpowiedzi przekazywane są za pomocą modelu obiektowego opisanego w punkcie [7 – Model danych API](#_Model_danych_API).  
     
   - **PUConnector.Notifications.Service** – biblioteka umożliwiająca odbiór powiadomień o zmianach stanów/statusów transakcji od PayU.  
     
   - **PUConnector.OrdersManager** – biblioteka pozwalająca rejestrować w bazie danych wszystkie zamówienia, zwroty oraz powiadomienia a także wszystkie związane z nimi żądania i odpowiedzi przesłane i odebrane do / z PayU. PUConnector zawiera gotową strukturę bazy danych dla silników baz danych firmy Microsoft: SQL Server oraz SQL CE.  
     
   - **PUConnector.SoapProxy.Service** – biblioteka umożliwiająca korzystanie z całej funkcjonalności PUConnector (a zarazem PayU API) za pomocą protokołu SOAP 1.1 / SOAP 1.2.  
      
   - **PUConnector.Db.Repository** – biblioteka pozwalająca na pełny dostęp do bazy danych zamówień, zwrotów oraz powiadomień za pomocą biblioteki ORM PetaPoco.  
     
   - **PUConnector.Db** – projekt bazy danych przechowującej zamówienia, zwroty oraz powiadomienia dla silników baz danych firmy Microsoft: SQL Server oraz SQL CE.  
     
   - **PUConnector.Commons** – biblioteka wspólna wykorzystywana przez pozostałe biblioteki PUConnector. Zawiera strukturę obiektowego modelu danych API (opisanego szerzej w punkcie [7 – Model danych API](#_Model_danych_API)), funkcjonalność obsługi i logowania błędów oraz kontroli bezpieczeństwa transmisji.  
     
   

## Dlaczego WCF ?

**PUConnector.SoapProxy.Service** oraz **PUConnector.Notifications.Service** to aplikacje typu usługa web zbudowane na bazie technologii WCF (Windows Communication Framework).

Wykorzystanie technologii WCF niesie ze sobą szereg korzyści:

- aplikacje wykorzystują sterownik **http.sys** do udostępniania swoich interfejsów pod adresami Url podlegającymi wstępnej rezerwacji. Umożliwia to np. działanie na jednym porcie nasłuchującym TCP(80) kilku aplikacji/usług jednocześnie (identyfikowanych różnymi bazowymi adresami Url).

- możliwość hostowania (uruchomienia) usługi w kontekście różnych procesów: jako aplikacja IIS, Windows Service, aplikacja konsolowa czy owin/katana

- dzięki modularnej budowie i zastosowaniu architektury SOA usługi WCF posiadają ogromne możliwości konfiguracji, rozszerzania i adaptacji

- wsparcie dla mechanizmów skalowania i równoważenia obciążenia (load balancing)

- obsługa wielu protokołów komunikacyjnych, transportowych oraz mechanizmów zabezpieczeń i szyfrowania

- WCF jest technologią silnie wspieraną, rozwijaną i promowaną przez firmę Microsoft i wykorzystywaną jako baza do bardziej wyspecjalizowanych usług sieciowych.

Po więcej szczegółów odsyłamy do:

<http://pl.wikipedia.org/wiki/Windows_Communication_Foundation>

<https://msdn.microsoft.com/pl-pl/library/dd456779(v=VS.110).aspx>

## Aktywne lub pasywne powiadamianie o zmianach stanu transakcji

Usługa PayU umożliwia pozyskiwanie informacji o stanie i statusie transakcji na dwa sposoby.

1. Powiadamianie aktywne (inicjowane przez PayU) - powiadomienia wysyłane przez serwis PayU do sprzedawcy:   
     
   <http://developers.payu.com/pl/restapi.html#notifications>  
     
   Metoda ta wymaga od sprzedawcy uruchomienia usługi web działającej pod publicznie dostępnym adresem Url. Serwis PayU przy każdej zmianie statusu transakcji łączy się ze wspomnianą usługą przekazując jej dane na temat aktualnego stanu transakcji.  
     
   PUConnector umożliwia odbieranie powiadomień kilkoma metodami:  
     
   a.) Za pomocą usługi WCF hostowanej na web serwerze IIS: **PUConnector.Notifications.Service**b.) Za pomocą usługi Windows Service lub aplikacji konsolowej: **PUConnector.Notifications.Host**  
   c.) W celach testowych i analitycznych – za pomocą aplikacji z graficznym interfejsem użytkownika:   
   **PUConnector.Api.UI**d.) Z poziomu kodu, za pomocą biblioteki .NET:  
   **PUConnector.Notifications.Service**
2. Powiadamianie pasywne (inicjowane przez Sprzedawcę) – sprzedawca periodycznie odpytuje serwis PayU o stan transakcji za pomocą żądania typu OrderRetrieve:  
     
   <http://developers.payu.com/pl/restapi.html#retrieving_order_data>  
     
   Metoda ta wymaga od sprzedawcy wysłanie żądania typu OrderRetrieveRequest do PayU Rest Api. W odpowiedzi system PayU zwraca aktualny stan i status transakcji.  
     
   PUConnector umożliwia wysłanie żądania (m.in. OrderRetrieveRequest) do PayU kilkoma metodami:  
     
   a.) Za pomocą aplikacji konsolowej:  
   **PUConnector.Api.Cmd**  
     
   b.) Za pomocą internetowej usługi sieciowej WebService (Soap 1.1/1.2) hostowanej na web serwerze IIS:  
   **PUConnector.SoapProxy.Service**  
   c.) Za pomocą internetowej usługi sieciowej WebService (Soap 1.1/1.2) hostowanej za pomocą usługi Windows Service lub aplikacji konsolowej :  
   **PUConnector.SoapProxy.Host**d.) W celach testowych i analitycznych – za pomocą aplikacji z graficznym interfejsem użytkownika:   
   **PUConnector.Api.UI**e.) Z poziomu kodu, za pomocą biblioteki .NET:  
   **PUConnector.Api**

## PUConnector Toolkit – zestaw aplikacji do integracji

### PUConnector.Api.Cmd

PUConnector.Api.Cmd jest aplikacją konsolową uruchamianą z systemowego wiersza poleceń. Aplikacja może być również uruchamiana z poziomu innego procesu lub aplikacji.   
  
Aplikacja umożliwia wysyłanie żądań (utworzenia nowego zamówienia, anulowania zamówienia, itd.) i odbieranie odpowiedzi na owe żądania od systemu PayU. Żądania jak i odpowiedzi przesyłane są w formacie JSON.   
  
Aplikacja na standardowym wejściu (stdin) przyjmuje typ i treść żądania które zostanie wysłane do PayU a na standardowym wyjściu (stdout) zwraca treść odpowiedzi odebraną od PayU.

Typy i struktura żądań i odpowiedzi opisana została w punkcie [7 – Model danych API.](#_Model_danych_API)

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku PUConnector.Api.Cmd.exe.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing).**

**Sposób użycia:**

Wysłanie żądania do systemu PayU wykonuje się uruchamiając aplikację PUConnector.Api.Cmd z dwoma parametrami:

- pierwszy parametr - typ żądania (requestType). Możliwe wartości to: OrderCreateRequest, OrderCancelRequest, OrderRetrieveRequest, OrderStatusUpdate, RefundCreateRequest. Typy i struktura żądań i odpowiedzi opisana została w punkcie [7 – Model danych API](#_Model_danych_API).

- drugi parametr – treść żądania (jsonRequest). Zawiera treść żądania w formacie Json. Struktura treści żądania musi odpowiadać przekazanemu w parametrze pierwszym typowi żądania. Typy i struktura żądań i odpowiedzi opisana została w punkcie [7 – Model danych API](#_Model_danych_API).  
  
**UWAGA! Cała treść żądania przekazywanego w parametrze drugim musi znaleźć się pomiędzy znakami cudzysłowu.   
  
UWAGA! Wartości parametrów tekstowych w treści żądania muszą znaleźć się pomiędzy znakami akcentu (‘).**

**Przykład użycia:**

|  |
| --- |
| PuConnector.Api.Cmd.exe OrderCreateRequest "{'merchantPosId':'145227', 'notifyUrl':'http://MyDomain.com/OrderNotify/Receive','customerIp':'90.1.1.1', 'description':'test order','currencyCode':'PLN','totalAmount':1000,'products':[{'name':'test product','unitPrice':1000,'quantity':1}]}" |

Powyższy przykład przedstawia wysłanie żądania utworzenia nowego zamówienia (OrderCreateRequest) dla punktu płatności nr 145227. Żądanie informuje, że adres Url pod którym nastąpi odbiór powiadomień od PayU to <http://MyDomain.com/OrderNotify/Receive> (więcej o odbieraniu powiadomień w punkcie [4.2 - PUConnector.Notifications.Service](#_PUConnector.Notifications.Service)), adres IP osoby dokonującej zakupu to 90.1.1.1, opis zamówienia to „test order”, waluta rozliczeniowa to PLN, pełna kwota zamówienia wynosi 10PLN (kwoty przekazywane są w groszach). Zamówienie zawiera jeden produkt o nazwie „test product” którego cena wynosi 10PLN.

**Przykładowa odpowiedź PayU:**

|  |
| --- |
| {"redirectUri":"https://secure.payu.com/pl/standard/co/summary?sessionId=fd31dc3  1-0afe-47c3-93fc-9c78ef2d8f6a&merchantPosId=145227&timeStamp=1429272452640&showL  oginDialog=false&apiToken=99741f609baec2df91629610e11f17f77626db50e291590565bf30  973f0d0368","orderId":"TPDDX9KQ1R150417GUEST000P01","extOrderId":"fd31dc31-0afe-  47c3-93fc-9c78ef2d8f6a","status":{"statusCode":"SUCCESS"}} |

Powyższa odpowiedź oznacza, iż zamówienie zostało poprawnie utworzone (statusCode = SUCCESS), zamówieniu nadany został numer zamówienia TPDDX9KQ1R150417GUEST000P01 a link pozwalający klientowi dokonać właściwej płatności to:

<https://secure.payu.com/pl/standard/co/summary?sessionId=fd31dc31-0afe-47c3-93fc-9c78ef2d8f6a&merchantPosId=145227&timeStamp=1429272452640&showLoginDialog=false&apiToken=99741f609baec2df91629610e11f17f77626db50e291590565bf30973f0d0368>

*Funkcjonalność aplikacji PUConnector.Api.Cmd realizuje biblioteka .NET o nazwie PUConnector.Api. Funkcjonalność rejestrowania transakcji, zwrotów, żądań i odpowiedzi realizuje biblioteka .NET o nazwie PUConnector.OrdersManager.*

### PUConnector.Notifications.Service

PUConnector.Notifications.Service jest aplikacją typu usługa WCF służącą do odbierania powiadomień o zmianie stanu/statusu transakcji od PayU.  
 **UWAGA! Ta aplikacja przeznaczona jest do uruchomienia pod kontrolą serwera web IIS.   
Jeśli chcesz uruchomić odbieranie powiadomień bez użycia IIS skorzystaj z aplikacji PUConnector.Notifications.Host. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing), w sposób zależny od konfiguracji udostępniać usługę web.**

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku Web.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**.**

**Instalacja:**

**UWAGA! Uruchomienie aplikacji na serwerze web IIS opisane zostało w punkcie** [**6.5 – Instalacja usług Notifications i SoapProxy na serwerze IIS**](#_Instalacja_usług_Notifications)**.**

Po poprawnym przeprowadzeniu procesu konfiguracji oraz instalacji usługa jest gotowa do odbierania i rejestrowania powiadomień o zmianach stanów/statusów transakcji od PayU.

### PUConnector.Notifications.Host

PUConnector.Notifications.Host jest aplikacją typu usługa Windows (Windows Service) / aplikacja konsolowa służącą do odbierania powiadomień o zmianie stanu/statusu transakcji od PayU.

Pod względem budowy PUConnector.Notifications.Host jest aplikacją hostującą czyli stanowiącą środowisko uruchomieniowe dla usługi WCF PUConnector.Notifications.Service (opisanej w poprzednim punkcie) służącej do odbierania powiadomień o zmianie stanu/statusu transakcji od PayU.

**UWAGA! Ta aplikacja przeznaczona jest do instalacji jako usługa Windows (Windows Service) lub uruchamiania jako aplikacja konsolowa z wiersza poleceń. Jeśli chcesz uruchomić odbieranie powiadomień przy użyciu web serwera IIS skorzystaj z aplikacji PUConnector.Notifications.Service.**

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku PUConnector.Notifications.Host.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing), w sposób zależny od konfiguracji udostępniać usługę web.**

**Instalacja:**

Aplikacja może działać w dwóch trybach: usługi Windows (Windows Service) lub aplikacji konsolowej.

Uruchomienie aplikacji z wiersza poleceń spowoduje jej uruchomienie w trybie aplikacji konsolowej działającej do czasu naciśnięcia przycisku Enter.

Zainstalowanie aplikacji jako usługi Windows (Windows Service) wymaga użycia z poziomu wiersza poleceń aplikacji InstallUtil.exe (będącej częścią platformy .Net Framework). Uruchamiając aplikację InstallUtil.exe należy przekazać dwa parametry: pierwszy parametr to ścieżka do aplikacji PUConnector.Notifications.Host.exe, drugi parametr to przełącznik /i.

|  |
| --- |
| C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe PUConnector.Notifications.Host.exe /i |

Aby odinstalować usługę należy użyć przełącznika /u.

Po poprawnym przeprowadzeniu procesu konfiguracji oraz instalacji usługa jest gotowa do odbierania i rejestrowania powiadomień o zmianach stanów/statusów transakcji od PayU.

### PUConnector.SoapProxy.Service

PUConnector.SoapProxy.Service jest aplikacją typu usługa WebService (WCF) umożliwiająca korzystanie z całej funkcjonalności PUConnector (a zarazem PayU API) za pomocą protokołu WebService SOAP 1.1 / SOAP 1.2.

**UWAGA! Ta aplikacja przeznaczona jest do uruchomienia pod kontrolą serwera web IIS.   
Jeśli chcesz uruchomić SoapProxy bez użycia IIS skorzystaj z aplikacji PUConnector.SoapProxy.Host.**

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku Web.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing), w sposób zależny od konfiguracji udostępniać usługę web.**

**Instalacja:**

**UWAGA! Uruchomienie aplikacji na serwerze web IIS opisane zostało w punkcie** [**6.5 – Instalacja usług Notifications i SoapProxy na serwerze IIS**](#_Instalacja_usług_Notifications)**.**

Po poprawnym przeprowadzeniu procesu konfiguracji oraz instalacji usługa udostępnia całą funkcjonalność bibliotek PUConnector.Api (wysyłanie żądań i odbieranie odpowiedzi od PayU) oraz PUConnector.OrdersManager (rejestrowanie wszystkich żądań, odpowiedzi i komunikacji z PayU w bazie dennych) za pomocą protokołu SOAP 1.1/1.2 (WebServices).

Uruchamia dwie usługi WebService pod adresami Url zdefiniowanymi w konfiguracji aplikacji. Pierwsza usługa udostępnia funkcjonalność biblioteki PUConnector.Api. Druga usługa udostępnia funkcjonalność biblioteki PUConnector.OrdersManager. Szczegółowy opis bibliotek znajduje się w punkcie [5 – PUConnector Libraries – biblioteki integracyjne dla platformy .NET](#_PUConnector_Libraries_–).

### PUConnector.SoapProxy.Host

PUConnector.SoapProxy.Host jest aplikacją typu usługa Windows (Windows Service) / aplikacja konsolowa umożliwiająca korzystanie z całej funkcjonalności PUConnectora (a zarazem PayU API) za pomocą protokołu WebService SOAP 1.1 / SOAP 1.2.

Pod względem budowy PUConnector.SoapProxy.Host jest aplikacją hostującą czyli stanowiącą środowisko uruchomieniowe dla usługi WCF PUConnector.SoapProxy.Service (opisaną w poprzednim punkcie).

**UWAGA! Ta aplikacja przeznaczona jest do instalacji jako usługa Windows (Windows Service) lub uruchamiania jako aplikacja konsolowa z wiersza poleceń. Jeśli chcesz uruchomić SoapProxy przy użyciu web serwera IIS skorzystaj z aplikacji PUConnector.SoapProxy.Service.**

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku PUConnector.SoapProxy.Host.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing), w sposób zależny od konfiguracji udostępniać usługę web.**

**Instalacja:**

Aplikacja może działać w dwóch trybach: usługi Windows (Windows Service) lub aplikacji konsolowej.

Uruchomienie aplikacji z wiersza poleceń spowoduje jej uruchomienie w trybie aplikacji konsolowej działającej do czasu naciśnięcia przycisku Enter.

Zainstalowanie aplikacji jako usługi Windows (Windows Service) wymaga użycia z poziomu wiersza poleceń aplikacji InstallUtil.exe (będącej częścią platformy .Net Framework). Uruchamiając aplikację InstallUtil.exe należy przekazać dwa parametry: pierwszy parametr to ścieżka do aplikacji PUConnector.Notifications.Host.exe, drugi parametr to przełącznik /i.

|  |
| --- |
| C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe PUConnector.SoapProxy.Host.exe /i |

Aby odinstalować usługę należy użyć przełącznika /u.

Po poprawnym przeprowadzeniu procesu konfiguracji oraz instalacji usługa udostępnia całą funkcjonalność bibliotek PUConnector.Api (wysyłanie żądań i odbieranie odpowiedzi od PayU) oraz PUConnector.OrdersManager (rejestrowanie wszystkich żądań, odpowiedzi i komunikacji z PayU w bazie dennych) za pomocą protokołu SOAP 1.1/1.2 (WebServices).

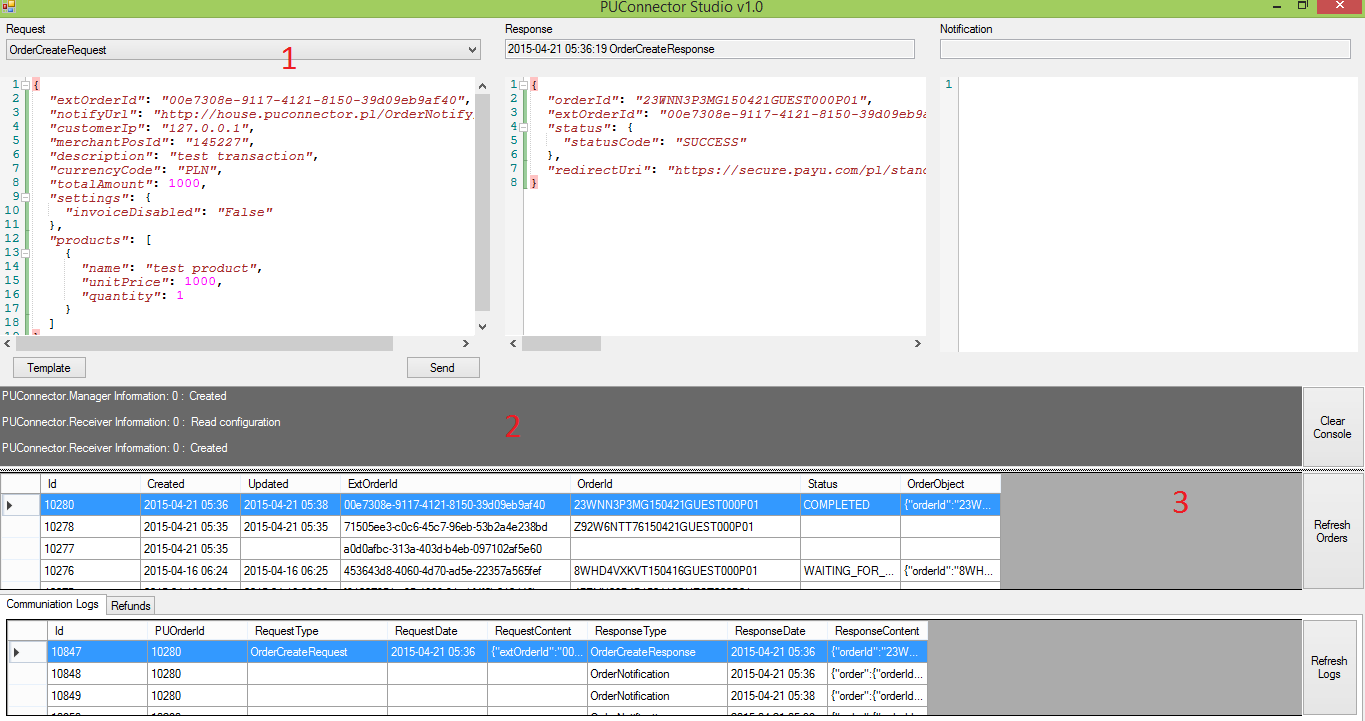
Aplikacja uruchamia dwie usługi WebService pod adresami Url zdefiniowanymi w konfiguracji aplikacji. Pierwsza usługa udostępnia funkcjonalność biblioteki PUConnector.Api. Druga usługa udostępnia funkcjonalność biblioteki PUConnector.OrdersManager. Szczegółowy opis bibliotek znajduje się w punkcie [5 – PUConnector Libraries – biblioteki integracyjne dla platformy .NET](#_PUConnector_Libraries_–).

### PUConnector.Api.UI czyli PUConnector Studio

PUConnector.Api.UI (nazwa kodowa PUConnector Studio) to aplikacja Windows z graficznym interfejsem użytkownika pozwalająca w prosty i wygodny sposób testować, analizować i optymalizować komunikację z PayU.

**Konfiguracja:**

**UWAGA! Przed rozpoczęciem korzystania z aplikacji należy ją skonfigurować poprzez modyfikację pliku PUConnector.SoapProxy.Host.config. Parametry konfiguracyjne opisane zostały w punkcie** [**6.1 – Parametry konfiguracyjne aplikacji i usług**](#_Parametry_konfiguracyjne_aplikacji)**. Aplikacja w zależności od konfiguracji może: rejestrować w bazie danych stany zamówień (transakcji) oraz komunikację z PayU , logować błędy w systemowym dzienniku EventLog, logować występujące zdarzenia i stan aplikacji (tracing).**



**Sposób użytkowania:**

Interfejs użytkownika aplikacji możemy podzielić na trzy obszary:

1. Obszar żądania, odpowiedzi i powiadomień. Obszar ten w trzech kolumnach przestawia treść żądania wysłanego bądź planowanego do wysłania do PayU (Request), treść odpowiedzi otrzymanej od PayU (Response) oraz w ostatniej kolumnie: treść ostatnio odebranego powiadomienia (Notifications) o zmianie stanu transakcji bądź refundacji.   
     
   W kolumnie Request użytkownik posiada możliwość wyboru typu żądania za pomocą rozwijanego pola, wpisania treści żądania w formacie JSON a następnie wysłanie owego żądania do systemu PayU za pomocą przycisku „Send”.  
     
   Wybór typu żądania (za pomocą rozwijalnego pola) powoduje wypełnienie treści pola żądania domyślnym szablonem. Szablon ten można również przywołać za pomocą przycisku „Template”.
2. Obszar konsoli – jest to obszar logowania w którym wyświetlane są komunikaty błędów oraz zdarzenia występujące w poszczególnych komponentach pakietu PUConnector i/lub platformy .Net Framework. Szczegółowość prezentowanych informacji zależna jest od konfiguracji aplikacji opisanej w punkcie [6.4 – Konfiguracja śledzenia](#_Konfiguracja_śledzenia_(tracing)). Przycisk „Clear Logs” umożliwia wyczyszczenie treści okna.
3. Obszar bazy danych – jest to obszar umożliwiający przeglądanie wszystkich zarejestrowanych a bazie danych transakcji, zwrotów oraz żądań i odpowiedzi do / z PayU.   
     
   Obszar jest widoczny pod warunkiem włączenia w konfiguracji aplikacji opcji połączenia z bazą danych (OrdersManager) – szczegóły w punkcie [6.1 – Parametry konfiguracyjne aplikacji i usług](#_Parametry_konfiguracyjne_aplikacji).   
     
   Pierwsza z tabel prezentuje listę zamówień (transakcji). Tabela poniżej reprezentuje listę żądań i odpowiedzi (zakładka Communication Logs) oraz zwrotów (zakładka Refund) wysłanych w kontekście wybranego w pierwszej tabeli zamówienia.  
     
   Wybór dowolnego wiersza w drugiej tabeli (żądań i odpowiedzi) powoduje wyświetlenie treści wysłanego żądania i odebranej odpowiedzi w obszarze pierwszym w kolumnach Request i Response. Wiersze zawierające treści odebranych od PayU powiadomień (notyfikacji) zawierają wypełnioną jedynie treść odpowiedzi (Response).  
     
   Wszystkie tabele można odświeżać ręcznie za pomocą przycisków „Refresh”.

**Odbieranie powiadomień**

Odebrane w trakcie działania aplikacji powiadomienia automatycznie wyświetlone zostaną w obszarze pierwszym obszarze w kolumnie Notifications oraz odświeżą widok transakcji wyświetlanych w trzecim obszarze. Poprawne działanie zależne jest od prawidłowej konfiguracji adresu Url dla usługi OrderNotify (szczegóły w punkcie [6.1 – Parametry konfiguracyjne aplikacji i usług](#_Parametry_konfiguracyjne_aplikacji)) oraz prawidłowego adresu Url przekazanego w polu „notifyUrl” żądania typu OrderCreateRequest (szczegóły w punkcie [7 – Model danych API](#_Model_danych_API)).

## PUConnector Libraries – biblioteki integracyjne dla platformy .NET

**PUConnector Libraries** to zestaw bibliotek programistycznych dla platformy .NET umożliwiających integrację z PayU.

Z punktu widzenia programisty rdzeń stanowią cztery biblioteki.  
Każda z nich zawiera dedykowaną klasę pozwalającą na dostęp do jej funkcjonalności:

* 1. Biblioteka **PUConnector.Api** i klasa **Connector***komunikacja z PayU, wysyłanie żądań i odbieranie odpowiedzi*
  2. Biblioteka **PUConnector.OrdersManager** i klasa **Manager***warstwa dostępu do bazy danych*
  3. Biblioteka **PUConnector.Notifications.Service** i klasa **Receiver***odbieranie powiadomień od PayU*
  4. Biblioteka **PUConnector.SoapProxy.Service** i klasa **WSProxy***webservice (soap) proxy dla klas Connector i Manager*

W przeciwieństwie do aplikacji z PUConnector Toolkit które wymagają deklaratywnej konfiguracji w plikach typu .config - PUConnector Libraries pozwalają na przekazanie wszystkich parametrów konfiguracyjnych w runtime czyli w kodzie aplikacji.   
  
Mamy zatem możliwość użycia bibliotek w dwóch trybach:

- z konfiguracją zapisaną deklaratywnie w plikach .config. W tym wypadku instancje wyżej wymienionych klas tworzymy za pomocą domyślnego bezparametrowego konstruktora.  
  
- z konfiguracją przekazaną w runtime jako parametry w konstruktorach klas bibliotek PUConnector. W tym wypadku instancję wyżej wymienionych klas tworzymy za pomocą sparametryzowanych konstruktorów pozwalających na przekazanie wszystkich parametrów konfiguracyjnych.

**Przykład konfiguracji przekazywanej w runtime przy wykorzystaniu wszystkich czterech bibliotek:**

Manager manager = new Manager(

"Data Source=(local);Initial Catalog=PUConnector.Db;User Id=user;Password=password",

"System.Data.SqlClient"

);

Connector connector = new Connector(

"145227",

"13a980d4f851f3d9a1cfc792fb1f5e50",

manager

);

Receiver receiver = new Receiver(

"http://localhost:90/OrderNotify",

"145227",

"13a980d4f851f3d9a1cfc792fb1f5e50",

manager

);

receiver.Start();

WSProxy wsProxy = new WSProxy(

connector,

"http://localhost:90/Connector",

manager,

"http://localhost:90/OrdersManager"

);

wsProxy.Start();

**Przykład utworzenia nowego zamówienia za pomocą klasy Connector:**

ProductRecord product1 = new ProductRecord(

"Produkt 1", 500, 1);

ProductRecord product2 = new ProductRecord(

"Produkt 2", 500, 1);

Connector connector = new Connector();

OrderCreateResponse order = connector.OrderCreate(

new OrderCreateRequest(

"https://mojsklep.pl/OrderNotify/Receive",

Request.UserHostAddress,

"Zakupy w sklepie mojsklep.pl",

"PLN",

1000,

product1,

product2));

if (order.Status.StatusCode != ResponseStatuses.SUCCESS)

throw new Exception("Błąd przetwarzania płatności:" +

order.Status.StatusDesc + " " +

order.Status.CodeLiteral

);

return Redirect(order.RedirectUri);

**UWAGA !**

**SZCZEGÓŁOWA DOKUMENTACJA TECHCZNICZNA BIBLIOTEK .NET PUCONNECTOR LIBRARIES ZNAJDUJE SIĘ W PLIKU:**

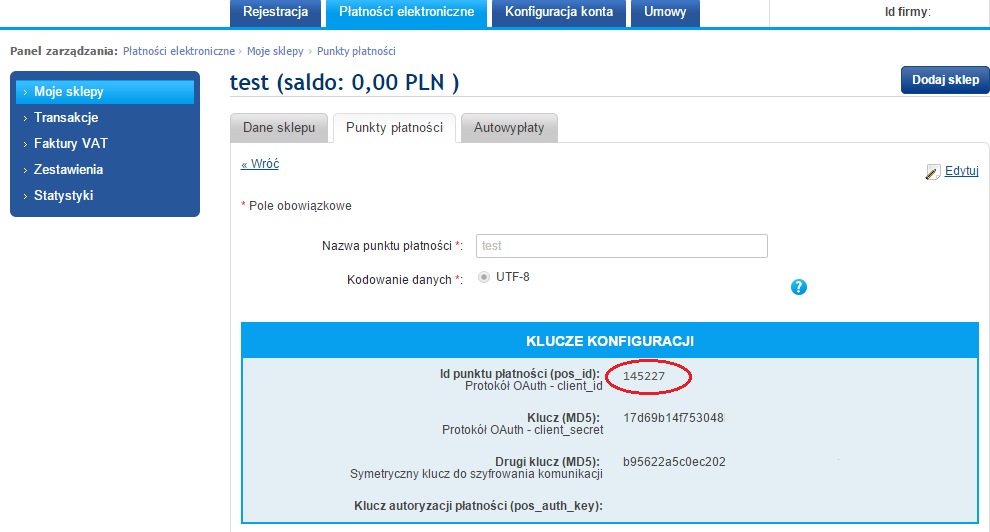
**PUConnector.Libraries.chm**

## Konfiguracja

### Parametry konfiguracyjne aplikacji i usług

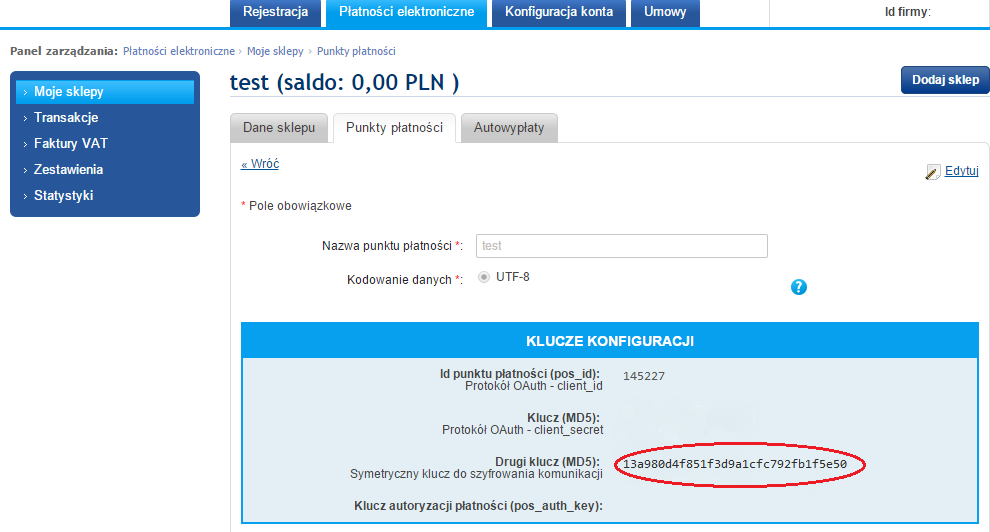
Każda z aplikacji wchodząca w skład PUConnector Toolkit wymaga przeprowadzenia konfiguracji. Aplikację konfigurujemy poprzez odpowiednie ustawienie wartości parametrów zapisanych w plikach o rozszerzeniu .config.

#### *6.1.1 MerchantPosId*

Parametr określający numer identyfikacyjny punktu płatności. Wartość tego parametru znajdziemy w panelu administracyjnym PayU (Moje Sklepy->Punkty płatności->Nazwa punktu płatności) pod nazwą „Id punktu płatności (pos\_id)”.  
  


#### *6.1.2 SecondSecurityKey*

Parametr określający klucz do szyfrowania danych podczas komunikacji z PayU. Wartość tego parametru znajdziemy w panelu administracyjnym PayU (Moje sklepy->Punkty płatności->Nazwa punktu płatności) pod nazwą „Drugi klucz (MD5)”.



#### 6.1.3 WriteExceptionsToEventLog

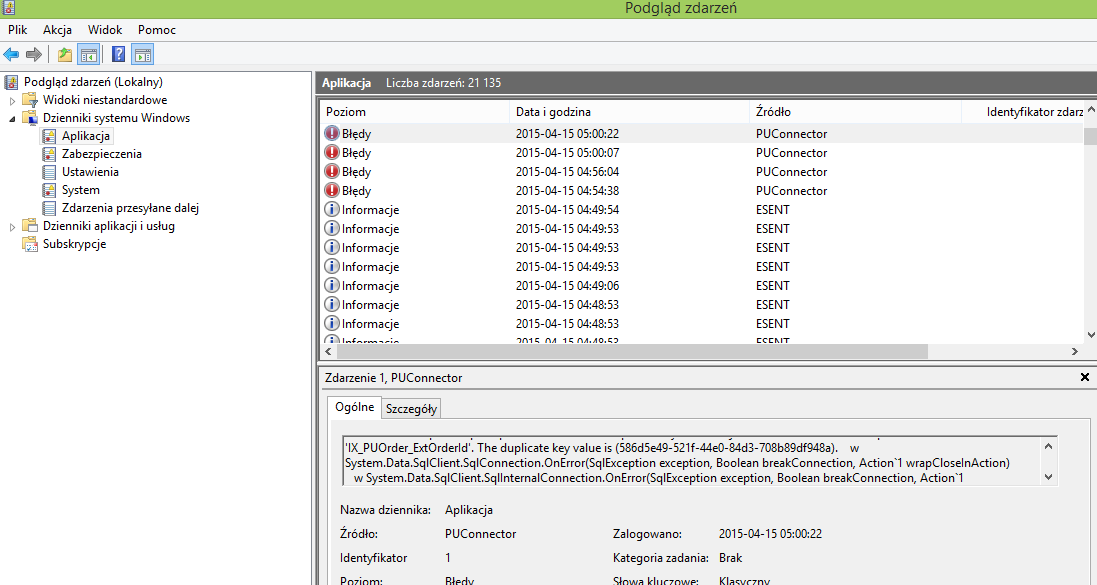
Parametr określający logowanie treści błędów które wystąpiły w modułach PUConnector do dziennika zdarzeń (EventLog) systemu Windows.

Parametr może przyjąć jedną z dwóch wartości:  
true – logowanie włączone  
false – logowanie wyłączone

**Uwaga! Przed włączeniem opcji logowania należy zarejestrować w systemie operacyjnym zdarzenie EventLog przypisane do PUConnector. Rejestrację zdarzenia należy dokonać z poziomu wiersza poleceń uruchomionego z uprawnieniami administracyjnymi:**

|  |
| --- |
| eventcreate /T ERROR /ID 1 /L APPLICATION /SO PUConnector /D "Unhandled exception" |

Podgląd dziennika zdarzeń możliwy jest za pomocą aplikacji „Podgląd zdarzeń” znajdującej się w systemowym Panelu sterowania w katalogu Narzędzia administracyjne.



#### 6.1.4 IgnoreIncorrectNotifications

Parametr dotyczący aplikacji / bibliotek odpowiedzialnych za odbieranie powiadomień (notyfikacji) o zmianach stanu transakcji od PayU.  
  
Parametr ten definiuje sposób zachowania bibliotek PUConnector w przypadku odebrania błędnego powiadomienia.  
  
Błędne powiadomienie ma miejsce w następujących przypadkach:  
  
- brak lub błędny podpis kryptograficzny podczas komunikacji kanałem nieszyfrowanym (http)  
- brak lub błędna autoryzacja podczas komunikacji kanałem szyfrowanym (https)  
- błędna treść powiadomienia  
- powiadomienie odnosi się do zamówienia nie zarejestrowanego w bazie danych (w przypadku włączonego i skonfigurowanego połączenia z bazą danych – parametr OrdersManager)  
  
Parametr może przyjąć jedną z dwóch wartości:  
  
false – w przypadku otrzymania błędnego powiadomienia PUConnector loguje fakt jego wystąpienia i odpowiada kodem błędu http. Zachowanie takie wymusza na systemie PayU ponowienie próby dostarczenia owego powiadomienia w późniejszym czasie.  
  
true – w przypadku otrzymania błędnego powiadomienia PUConnector loguje fakt jego wystąpienia i odpowiada kodem 200 (OK) http oznaczającym poprawne odebranie powiadomienia. Zachowanie takie powoduje, że system PayU nie będzie podejmował więcej próby dostarczenia owego powiadomienia w późniejszym czasie.

Domyślna i zalecana wartość parametru to true.

#### 6.1.5 OrdersLimit

Parametr ten dotyczy jedynie aplikacji PUConnector.Api.UI (PUConnector Studio) i jest wartością liczbową określającą maksymalną liczbę wyświetlanych zamówień / transakcji.

Lista zamówień zarejestrowanych w bazie danych (w przypadku włączonego i skonfigurowanego połączenia z bazą danych – parametr OrdersManager) wyświetlana jest w obszarze bazy danych aplikacji PUConnector.Api.UI opisanym szerzej w punkcie [4.6 – PUConnector.Api.UI](#_PUConnector.Api.UI).

#### 6.1.6 OrdersManager

Parametr występujący w sekcji **<connectionStrings>** plików konfiguracyjnych.

Parametr ten definiuje połączenie z bazą danych PUConnector w której rejestrowane będą wszystkie zamówienia / transakcję oraz komunikacja z PayU.

Parametr ten jest opcjonalny ale zaleca się jego ustawienie w celu skorzystania z pełnych możliwości bibliotek PUConnector.

PUConnector dostarczany jest wraz ze skryptem oraz projektem bazy danych możliwym do uruchomienia na silniku bazy danych Microsoft SQL Server. Dodatkowo PUConnector dostarczany jest wraz ze skryptem oraz przykładowym plikiem bazy danych dla silnika bazy danych Microsoft SQL CE. Więcej informacji na ten temat znajdziesz w punkcie [6.3 – Konfiguracja bazy danych](#_Konfiguracja_bazy_danych).

Dla bazy danych SQL Server atrybut „providerName” powinien przyjąć wartość „System.Data.SqlClient”, dla bazy danych SQL CE powinien przyjąć wartość „System.Data.SqlServerCe.4.0”.

Przykład dla SQL Server:

<connectionStrings>

<add name="OrdersManager" connectionString="Data Source=(local);Initial Catalog=PUConnector.Db;User Id=user;Password=password" providerName="System.Data.SqlClient"/>

</connectionStrings>

Przykład dla SQL CE:

<connectionStrings>

<add name="OrdersManager" connectionString="Data Source=d:\some\path\puconnector.sdf" providerName="System.Data.SqlServerCe.4.0"/>

</connectionStrings>

Więcej informacji na temat składni parametrów typu connectionString znajdziesz pod adresem:

<https://msdn.microsoft.com/en-us/library/windows/desktop/ms722656%28v=vs.85%29.aspx>

#### 6.1.7 Konfiguracja usług WCF

**PUConnector.Notifications.Service** oraz **PUConnector.SoapProxy.Service** są aplikacjami typu usługa WCF których konfiguracja zawarta jest w plikach Web.config lub App.config (App.config w przypadku aplikacji hostujących owe usługi poza serwerem IIS: PUConnector.Notifications.Host, PUConnector.SoapProxy.Host, PUConnector.Api.UI).

Ogromne możliwości konfiguracyjne usług WCF znacznie wykraczają poza ramy tego dokumentu. Przedstawione zostaną jedynie ustawienia kluczowe dla uruchomienia i działania usług.

Po kompletny opis możliwości konfiguracyjnych usług WCF odsyłamy do dokumentacji firmy Microsoft:

<https://msdn.microsoft.com/en-us/library/ms733932%28v=vs.110%29.aspx>

<https://msdn.microsoft.com/en-us/library/ms731734(v=vs.110).aspx>

**<webHttpBinding>**

Gałąź konfiguracyjna wykorzystywana przez usługę odbierania powiadomień (Notifications) od PayU i zawarta w aplikacjach:  
  
- PUConnector.Notifications.Service  
**-** PUConnector.Notifications.Host  
- PUConnector.Api.UI  
  
<webHttpBinding>  
 <binding contentTypeMapper="PUConnector.Notifications.Service.WebContentTypeMapperExt, PUConnector.Notifications.Service">  
 <security mode="None"></security>  
 </binding>  
</webHttpBinding>

Gałąź *<webHttpBinding>* zawiera element *<security>* którego atrybut *mode* nalży ustawić na wartość:

**None** – w przypadku odbierania powiadomień protokołem HTTP

**Transport** – w przypadku odbierania powiadomień bezpiecznym protokołem HTTPS

**Uwaga! Więcej na ten temat konfiguracji protokołu HTTPS/SSL znajdziesz w punkcie** [**6.2 – Konfiguracja sieci i zabezpieczeń oraz protokołu HTTPS (SSL)**](#_Konfiguracja_sieci_i)**.**

**<basicHttpBinding>**

Gałąź konfiguracyjna wykorzystywana przez usługę WebService SoapProxy i zawarta w aplikacjach:  
  
- PUConnector.SoapProxy.Service  
- PUConnector.SoapProxy.Host

<basicHttpBinding>

<binding>

<security mode="None"></security>

</binding>

</basicHttpBinding>

Gałąź *<basicHttpBinding>* zawiera element *<security>* którego atrybut *mode* nalży ustawić na wartość:

**None** – w przypadku dostępu do webservice za pomocą protokołu HTTP

**Transport** – w przypadku dostępu do webservice bezpiecznym protokołem HTTPS

**Uwaga! Więcej na ten temat konfiguracji protokołu HTTPS/SSL znajdziesz w punkcie 6.2 – Konfiguracja sieci i zabezpieczeń oraz protokołu HTTP/SSL.**

**<baseAddress>**

Gałąź konfiguracyjna wykorzystywana zarówno przez usługę odbierania powiadomień Notifications jak i przez usługę WebService SoapProxy. Hostowane przez aplikacje:  
  
**-** PUConnector.Notifications.Host  
- PUConnector.Api.UI   
- PUConnector.SoapProxy.Host  
  
Umożliwia zdefiniowanie adresu punktu końcowego (endpoint) czyli adresu Url pod którym zostanie udostępniona dana usługa sieciowa.

**NIE DOTYCZY WERSJI DLA WEB SERWERA IIS** (PUConnector.SoapProxy.Service, PUConnector.Notifications.Service). W obrębie web serwera IIS konfigurację adresu punktu końcowego przeprowadzamy podczas operacji dodawania witryny w aplikacji „Menedżer internetowych usług informacyjnych (IIS)” – więcej w punkcie [6.5 - Instalacja usług Notifications i SoapProxy na serwerze IIS](#_Instalacja_usług_Notifications).

Usługa odbierania powiadomień **Notifications** posiada gałąź konfiguracyjną <**baseAddress**> zapisaną w konfiguracjach aplikacji:  
  
**-** PUConnector.Notifications.Host  
- PUConnector.Api.UI

<service name="PUConnector.Notifications.Service.OrderNotify">

<endpoint binding="webHttpBinding" contract="PUConnector.Notifications.Service.IOrderNotify"></endpoint>

<host>

<baseAddresses>

<add baseAddress="http://localhost:90/OrderNotify" />

</baseAddresses>

</host>

</service>

**Powyższy adres Url należy zarejestrować w systemowym module http za pomocą polecenia nestsh z poziomu wiersza poleceń uruchomionego z uprawnieniami administracyjnymi:**

|  |
| --- |
| netsh http add urlacl url=http://+:90/OrderNotify/ user=nazwa\_uzytkownika |

**Uwaga! Powyższy adres Url może zostać skonfigurowany dla protokołu http lub https. Więcej na ten temat konfiguracji HTTPS/SSL znajdziesz w punkcie** [**6.2 - Konfiguracja sieci i zabezpieczeń oraz protokołu HTTPS (SSL)**](#_Konfiguracja_sieci_i)**.**

**Uwaga! Usługa odbierania powiadomień udostępnia metodę Test dostępną pod ścieżką /Test zdefiniowanego adresu Url. Dla powyższego przypadku byłby to adres** [**http://localhost:90/OrderNotify/Test**](http://localhost:90/OrderNotify/Test)**. Adres otwarty z poziomu przeglądarki powinien wyświetlić napis „OK”. Metoda ta umożliwia szybkie przetestowanie dostępności usługi.**

Usługa WebService **SoapProxy** posiada dwa wystąpienia gałęzi konfiguracyjnej <**baseAddress**> zapisane w konfiguracji aplikacji:  
  
- PUConnector.SoapProxy.Host  
  
Pierwsze wystąpienie (znajdujące się w gałęzi **<service name="PUConnector.Api.Connector">**) dotyczy usługi webservice udostępniającej funkcjonalność biblioteki PUConnector.Api (wysyłanie żądań i odbieranie odpowiedzi od PayU).  
  
Drugie wystąpienie (znajdujące się w gałęzi **<service name="PUConnector.OrdersManager.Manager">**) dotyczy usługi webservice udostępniającej funkcjonalność biblioteki PUConnector.OrdersManager (rejestrowanie w bazie danych żądań, odpowiedzi i całej komunikacji z PayU).

<services>

<service name="PUConnector.Api.Connector">

<endpoint binding="basicHttpBinding" contract="PUConnector.Api.IConnector"></endpoint>

<host>

<baseAddresses>

<add baseAddress="http://localhost:90/Connector" />

</baseAddresses>

</host>

</service>

<service name="PUConnector.OrdersManager.Manager">

<endpoint binding="basicHttpBinding" contract="PUConnector.OrdersManager.IManager"></endpoint>

<host>

<baseAddresses>

<add baseAddress="http://localhost:90/Manager" />

</baseAddresses>

</host>

</service>

</services>

**Powyższe adresy Url należy zarejestrować w systemowym module http za pomocą polecenia nestsh z poziomu wiersza poleceń uruchomionego z uprawnieniami administracyjnymi:**

|  |
| --- |
| netsh http add urlacl url=http://+:90/Connector/ user=nazwa\_uzytkownika |

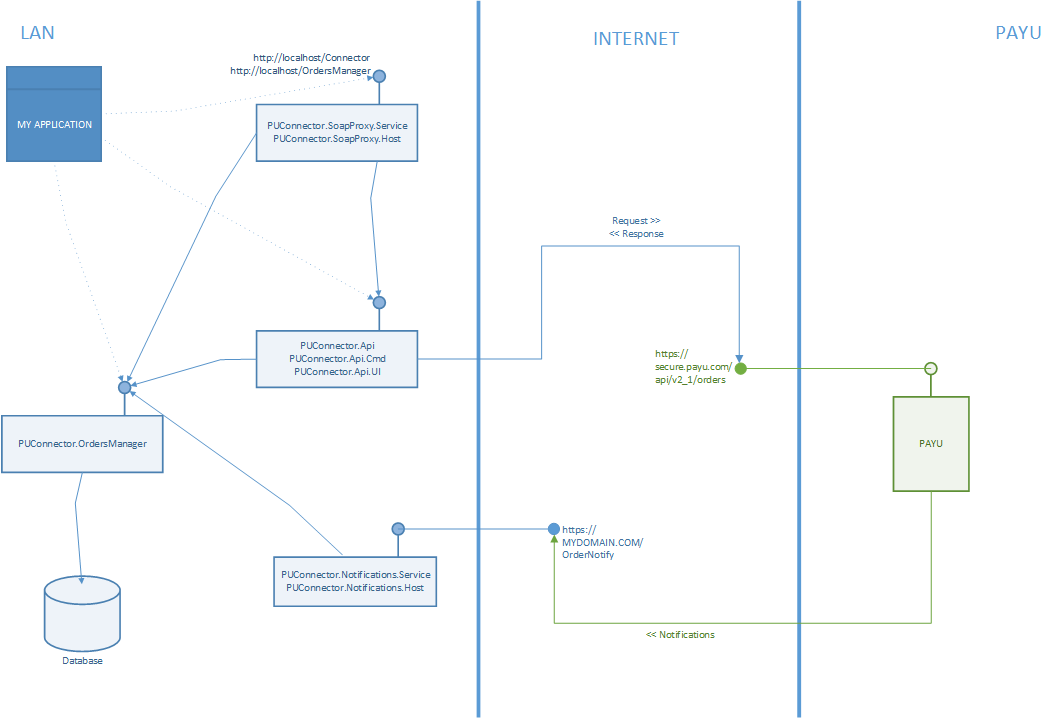
|  |
| --- |
| netsh http add urlacl url=http://+:90/Manager/ user=nazwa\_uzytkownika |

**Uwaga! Powyższe adresy Url mogą zostać skonfigurowane dla protokołu http lub https. Więcej na ten temat konfiguracji HTTPS/SSL znajdziesz w punkcie** [**6.2 - Konfiguracja sieci i zabezpieczeń oraz protokołu HTTPS (SSL)**](#_Konfiguracja_sieci_i)**.**

### Konfiguracja sieci i zabezpieczeń oraz protokołu HTTPS (SSL)

Komunikacja z Rest PayU Api następuje bezpiecznym kanałem HTTPS zapewniając poufność i integralność wymiany danych.  
  
Komunikacja zwrotna czyli powiadomienia (Notifications) o zmianach stanu transakcji otrzymywane od PayU wymagają od sprzedawcy uruchomienia usługi web działającej pod publicznie dostępnym adresem Url. Serwis PayU przy każdej zmianie statusu/stanu transakcji łączy się ze wspomnianą usługą przekazując jej dane na temat aktualnego stanu transakcji. Funkcjonalność odbierania powiadomień realizuje usługa PUConnector.Notifications.Service którą można uruchomić na web serwerze IIS (patrz punkt [6.5 - Instalacja usług Notifications i SoapProxy na serwerze IIS](#_Instalacja_usług_Notifications)) lub jako usługę Windows Service / aplikację konsolową (patrz punkt [4.3 – PUConnector.Notifications.Host](#_PUConnector.Notifications.Host)).

Po przeprowadzeniu konfiguracji usługi PUConnector.Notifications.Service (lub aplikacji PUConnector.Notifications.Host w przypadku hostowania usługi jako Windows Service / aplikacja konsolowa) [wg punktów [6.1.7 - Konfiguracja usług WCF](#_6.1.7_Konfiguracja_usług) i ewentualnie [6.5 - Instalacja usług Notifications i SoapProxy na serwerze IIS](#_Instalacja_usług_Notifications)] należy na routerze brzegowym (łączącym sieć LAN z Internetem) udostępnić (zazwyczaj za pomocą opcji port forwarding) port tcp/ip na którym uruchomiliśmy usługę odbierania powiadomień (we wszystkich naszych przykładach portem tym jest port 90). Alternatywną opcją jest instalacja usługi PUConnector.Notifications.Service lub aplikacji PUConnector.Notifications.Host na komputerze znajdującym się w tzw. strefie DMZ (Demilitarized Zone) posiadającej bezpośrednie połączenie z siecią Internet.



**Uwaga! Upewnij się, że Twój punkt płatności w PayU został utworzony w wariancie współpracy "Checkout - Express Payment”. W przeciwnym wypadku korzystanie z usługi PayU Rest API nie będzie możliwe.**

**Konfiguracja protokołu HTTPS (SSL) dla usługi odbierana powiadomień.**

Konfiguracja bezpiecznego kanału HTTPS (SSL) dla usługi odbierania powiadomień (PUConnector.Notifications.Service / PUConnector.Notifications.Host) wymaga wykonania następujących czynności:

1. W konfiguracji usługi odbierania powiadomień ustawić atrybut mode elementu *<secure>* na wartość: Transport – patrz punkt [6.1.7 - Konfiguracja usług WCF](#_6.1.7_Konfiguracja_usług)
2. W konfiguracji usługi odbierania powiadomień ustawić adresy Url elementu *<baseAddress>* na wartość zaczynającą się od frazy https:// – patrz punkt [6.1.7 - Konfiguracja usług WCF](#_6.1.7_Konfiguracja_usług)
3. Zainstalować certyfikat w magazynie komputera lokalnego (Local Machine Store)
4. Odczytać z treści certyfikatu nr odcisku palca (tzw. thumbprint)
5. Powiązać port na którym uruchomiliśmy usługę odbierania powiadomień z zainstalowanym certyfikatem. Powiązania dokonujemy za pomocą polecenia netsh uruchamianego z poziomu wiersza poleceń uruchomionego z uprawnieniami administratora:

|  |
| --- |
| netsh http add sslcert ipport=0.0.0.0:90 odcisk\_palca\_z\_certyfikatu appid={5b904e2a-e620-4a57-a059-5970f8e54977} |

1. Na routerze brzegowym (łączącym sieć LAN z siecią Internet) udostępnić naszą usługę na standardowym porcie 443 (HTTPS)
2. Wysyłając do PayU żądanie utworzenia nowego zamówienia / transakcji OrderCreateRequest należy ustawić pole notifyUrl na adres domenowy powiązany z certyfikatem

**Usługa SoapProxy**

Komunikacja z usługą webservice SoapProxy (PUConnector.SoapProxy.Service / PUConnector.SoapProxy.Host powinna odbywać się w obrębie sieci lokalnej – w tym wypadku nie więc ma konieczności zestawiania bezpiecznego kanału (https) szyfrowanego za pomocą SSL. Jeśli jednak chcielibyśmy to zrobić – należy przeprowadzić analogiczną konfigurację jak opisana powyżej dla usługi odbierania powiadomień.

### Konfiguracja bazy danych

W folderze PUConnector.DB umieszczony został plik PUConnector.sql zawierający skrypt SQL dla silnika bazy danych Microsoft SQL Server tworzący w bazie strukturę pozwalającą na rejestrowanie transakcji przez PUConnector. Struktura składa się z trzech powiązanych tabel:   
  
- PUOrder – zamówienia / transakcje  
- PUCommLog – logi komunikacyjne, treści wszystkich wysłanych żądań i odebranych odpowiedzi do/z PayU  
- PURefund –zwroty (refunds)

Skrypt może zostać uruchomiony zarówno na nowej bazie danych dedykowanej dla PUConnector lub istniejącej bazie danych którą będziemy chcieli zintegrować z powyższą strukturą PUConnector.

Dodatkowo w folderze PUConnector.DB umieszczony został plik puconnector.sqlce zawierający skrypt SQL dla silnika bazy danych Microsoft SQL CE tworzący powyżej opisaną strukturę oraz plik puconnector.sdf stanowiący gotowy do pracy plik bazy danych Microsoft SQL CE. Silnik Microsoft SQL CE w przeciwieństwie do Microsoft SQL Server nie wymaga instalacji żadnego dodatkowego oprogramowania.

**Uwaga! Włączenie obsługi bazy danych wymaga ustawienia parametru <connectionString> w plikach konfiguracyjnych PUConnector. Więcej informacji na ten temat znajdziesz w punktach:**[**6.1 – Parametry konfiguracyjne aplikacji**](#_Parametry_konfiguracyjne_aplikacji)[**6.1.6 – Orders Manager**](#_6.1.6_OrdersManager)

### Konfiguracja śledzenia (tracing) oraz dziennika logów EventLog

Każda z aplikacji PUConnector Toolkit posiada możliwość śledzenia zdarzeń występujących w trakcie działania aplikacji.

Włączenie śledzenia zdarzeń dokonuje się za pomocą gałęzi <system.diagnostics> w pliku konfiguracyjnym (.config).

PUConnector definiuje cztery własne tzw. źródła zdarzeń (Trace Source):

PUConnector – ogólne zdarzenia zgłoszone przez PUConnector

PUConnector.Connector – zdarzenia zgłoszone przez warstwę komunikacyjną

PUConnector.Manager – zdarzenia zgłoszone przez warstwę dostępu do bazy danych

PUConnector.Receiver – zdarzenia zgłoszone przez warstwę odbierania powiadomień PayU

Jednocześnie możesz śledzić szereg zdarzeń występujących w różnych komponentach platformy .NET Framework.

Przykład włączonego mechanizmu śledzenia znajdziesz w konfiguracji aplikacji PUConnector.Api.UI. Skonfigurowane tam źródła zdarzeń domyślnie logują do okna logów aplikacji PUConnector.Api.UI za co odpowiada dedykowany tzw. listener: PUUIListener.

Włączając sekcje znajdujące się w komentarzach możesz np. dodać logowanie zdarzeń sieciowych występujących na poziomie platformy .NET (System.Net, System.ServiceModel) i/lub włączyć zapisywanie zdarzeń do plików logów (XmlTraceFile, TraceFile).

Pełny opis możliwości mechanizmów śledzenia znacząco wykracza poza ramy niniejszego dokumentu. Po szczegóły odsyłamy do źródeł Microsoft.

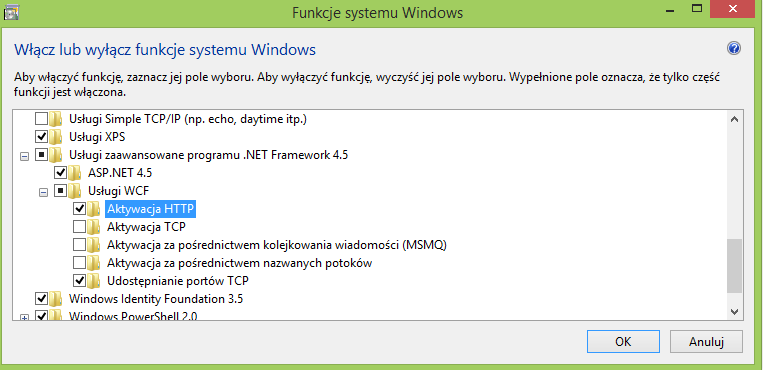
<https://msdn.microsoft.com/pl-pl/library/76dt1k3h(v=vs.110).aspx>

**Uwaga! Opis konfiguracji dziennika EventLog dla PUConnector znajdziesz w punkcie** [**6.1.3 - WriteExceptionsToEventLog**](#_6.1.3_WriteExceptionsToEventLog)

### Instalacja usług Notifications i SoapProxy na serwerze IIS

Aplikacje **PUConnector.Notifications.Service** oraz **PUConnector.SoapProxy.Service** to aplikacje typu **usługa WCF** przeznaczone do uruchomienia na web serwerze IIS.

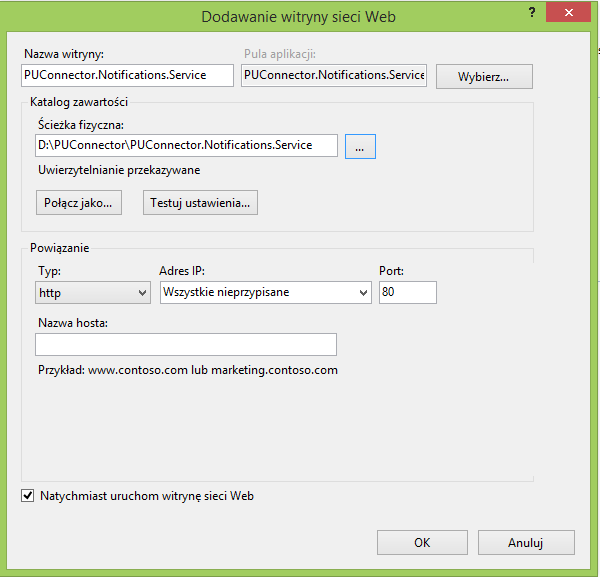
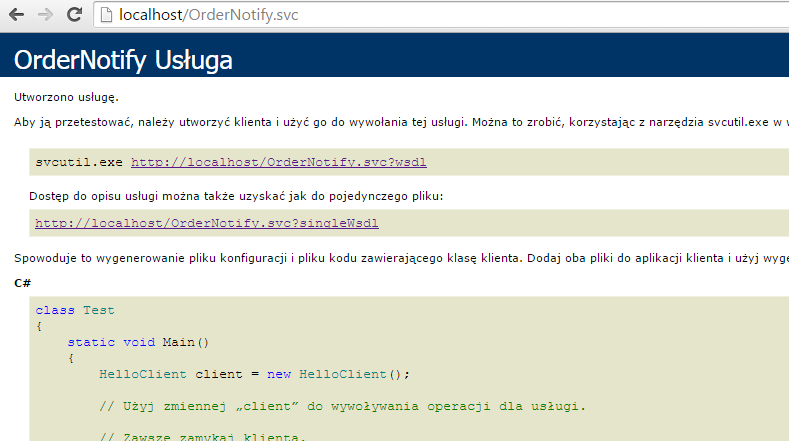
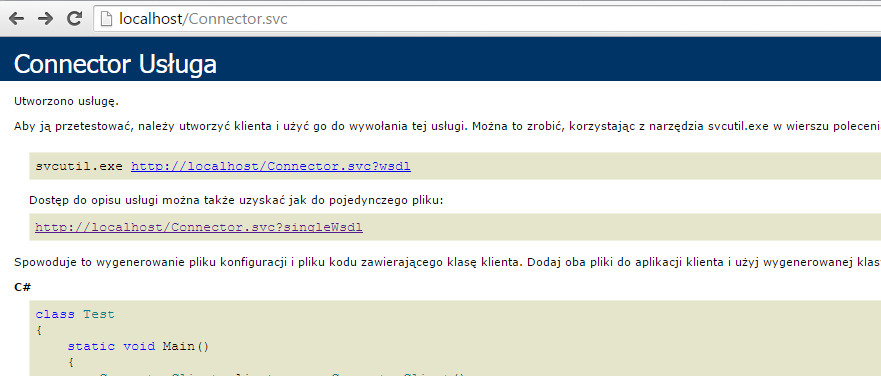
Uruchomienie usług na serwerze IIS wymaga zainstalowanego mechanizmu aktywacji HTTP dla usług WCF:

1. Przejdź do: „Panel Sterowania -> Programy i funkcje -> Włączy lub wyłącz funkcje systemu Windows”. Sprawdź czy zainstalowany został składnik „Aktywacja HTTP” dla usług WCF. Jeśli wspomniany składnik systemu nie jest zainstalowany – zaznacz go i zainstaluj.  
     
   
2. Uruchom wiersz polecenia z uprawnieniami administracyjnymi a następnie wykonaj następująco po sobie dwie komendy:

|  |
| --- |
| "C:\Windows\Microsoft.NET\Framework\v3.0\Windows Communication Foundation\servicemodelreg" -i |

|  |
| --- |
| iisreset |

Aby zainstalować usługi na serwerze IIS należy wykonać następujące czynności:

1. Uruchom „Menedżer internetowych usług informacyjnych (IIS)”.
2. Dodaj nową witrynę sieci web ustawiając ścieżkę fizyczną na folder w którym znajduje się aplikacja **PUConnector.Notifications.Service** (i plik OrderNotify.svc) lub aplikacja **PUConnector.SoapProxy.Service** (i pliki Connector.svc oraz Manager.svc).   
     
   
3. Upewnij się również, że pula aplikacji posiada ustawioną wersję 4.0 środowiska .NET CLR zarządzaną w trybie „Zintegrowany”.
4. Sprawdź poprawność instalacji otwierając witrynę w przeglądarce internetowej.
5. Dla aplikacji **PUConnector.Notifications.Service** witryna powinna być dostępna pod ścieżką /OrderNotify.svc adresu url. Pełny adres url będzie oczywiście zależny od powiązania jakie zdefiniowałeś dodając witrynę (punkt 3). Jeśli nie modyfikowałeś domyślnych parametrów wiązania to usługa dostępna będzie pod adresem:   
     
   <http://localhost/OrderNotify.svc>  
     
   **Uwaga! Usługa odbierania powiadomień udostępnia metodę Test dostępną pod ścieżką /Test zdefiniowanego adresu Url. Dla powyższego przypadku byłby to adres http://localhost/OrderNotify.svc/Test. Adres otwarty z poziomu przeglądarki powinien wyświetlić napis „OK”. Metoda ma na celu możliwość szybkiego przetestowania dostępności usługi.**  
     
     
   
6. Dla aplikacji **PUConnector.SoapProxy.Service** witryna powinna być dostępna pod ścieżkami /Connector.svc oraz /Manager.svc adresu url. Pełny adres url będzie oczywiście zależny od powiązania jakie zdefiniowałeś dodając witrynę (punkt 3). Jeśli nie modyfikowałeś domyślnych parametrów wiązania to usługa dostępna będzie pod adresami:   
     
   <http://localhost/Connector.svc>  
     
   <http://localhost/Manager.svc>  
     
   
7. Jeśli posiadasz certyfikat SSL i chcesz aby powiadomienia od PayU odbierane były bezpiecznym kanałem HTTPS - powiąż w standardowy sposób witrynę z protokołem HTTPS i zainstalowanym certyfikatem.   
   <http://www.iis.net/learn/manage/configuring-security/how-to-set-up-ssl-on-iis>

## Model danych API

Głównymi strukturami danych na których opiera się komunikacja z PayU są struktury żądań i odpowiedzi na owe żądania.

Zdefiniowanych zostało pięć par żądanie-odpowiedź:

1. **OrderCreateRequest** i **OrderCreateResponse** – żądanie utworzenia zamówienia/transakcji i odpowiedź na nie
2. **OrderCancelRequest** i **OrderCancelResponse** – żądanie anulowania zamówienia/transakcji i odpowiedź na nie
3. **OrderRetrieveRequest** i **OrderRetrieveResponse** – żądanie pobrania szczegółów zamówienia/transakcji i odpowiedź na nie
4. **OrderStatusUpdateRequest** i **OrderStatusUpdateResponse** – żądanie modyfikacji statusu zamówienia/transakcji i odpowiedź na nie
5. **RefundCreateRequest** i **RefundCreateResponse** – żądanie dokonania zwrotu dla zamówienia/transakcji i odpowiedź na nie

### OrderId i ExtOrderId, RefundId i ExtRefundId

Szczególnymi polami struktur danych na których opiera się komunikacja z PayU są pole identyfikujące zamówienie/transakcję oraz zwrot środków (tzw. refund).

Identyfikatory zamówień/transakcji oraz zwrotów możemy podzielić na dwie kategorie:

1. Identyfikatory wewnętrzne – nadawane przez operatora PayU:  
   **OrderId** = identyfikator zamówienia/transakcji  
   **RefundId** = identyfikator zwrotu
2. Identyfikatory zewnętrzne – nadawane przez sprzedawcę:  
   **ExtOrderId** = identyfikator zamówienia/transakcji  
   **ExtRefundId** = identyfikator zwrotu

Identyfikatory wewnętrzne są identyfikatorami obligatoryjnymi nadawanymi automatycznie przez PayU. Identyfikatory zewnętrzne są identyfikatorami opcjonalnymi nadawanymi przez sprzedawcę.  
  
PUConnector automatycznie nadaje identyfikatory zewnętrzne wszystkim nowym transakcjom oraz zwrotom. Mechanizm ten zapewnia dodatkową kontrolę integralności danych. Użytkownik może jawnie przekazać własny nadany przez siebie identyfikator ExtOrderId / ExtRefundId – w tym wypadku PUConnector nie wygeneruje go automatycznie.

### PUConnector Toolkit

Komunikacja z PayU odbywa się za pomocą komunikatów w formacie JSON.

Korzystając z aplikacji PUConnector.Api.Cmd oraz PUConnector.Api.UI przekazujesz dane również w tym formacie (w PUConnector.Api.Cmd w sposób szczególny sformatowane ze względu na sposób ich przekazywania z poziomu wiersza poleceń – szczegóły w punkcie 4.1 – PUConnector.Api.Cmd) .

Dokumentacje struktury komunikatów znajdziesz na stronie PayU:

<http://developers.payu.com/pl/restapi.html#references_api_parameters>

oraz w pliku:

**PUConnector.JSON.txt**

### PUConnector Libraries

Korzystając z bibliotek .NET z PUConnector Libraries nie musisz przejmować się parsowaniem oraz serializacją danych z/do formatu JSON.

Wszystkie struktury danych używane podczas komunikacji z PayU zostały opakowane w obiektowy model danych który zapewnia łatwe i wygodne korzystanie z PUConnector Libraries.

Implementacja obiektowego modelu danych znajduje się w bibliotece **PUConnector.Commons** w przestrzeni nazw **PUConnector.Commons.ApiModel**. Każda ze struktur danych wyrażona została w postaci odrębnej klasy. W każdej z nich przekazanie wartości pól WYMAGANYCH wymuszają konstruktory tych klas.

**Pełny opis obiektowego modelu danych znajdziesz w pliku:**

**PUConnector.Libraries.chm**w rozdziale:  
PUConnector.Commons.ApiModel