enova365 dla biznesu systemowy rozwój firmy

enova365

Oprogramowanie ERP do zarządzania.
Wzmacnia firmę i rośnie wraz z nią.
www.enova.pl

Spis treści

enova365.Integrator

Czym jest i do czego służy dodatek enova365.Integrator ?

Dodatek **enova365.Integrator** udostępnia interfejs programistyczny (API) do enova365. Pozwala na pobieranie danych z enova365 oraz ich dodawanie i aktualizację przez zewnętrzne systemy informatyczne. Umożliwia współpracę enova365 z systemami zewnętrznymi niezależnie ich od platformy sprzętowej. Komunikacja pomiędzy systemami następuje poprzez usługę sieciową (Web Service) za pomocą protokołu HTTP i z wykorzystaniem formatu SOAP-XML. Struktura pobieranych lub przekazywanych danych w wiadomości SOAP 1.1 jest w pełni definiowalna (w ramach formatu XML). Definiowanie formatu danych odbywa się za pomocą sprawdzonych mechanizmów zastosowanych w enova365.EDI.

Integracja dwóch systemów polega zazwyczaj na stworzeniu serwisu (aplikacji), który komunikuje się z każdym z tych systemów. enova365.Integrator zapewnia komunikację serwisu integrującego z instalacją enova365.

Dodatek udostępnia tylko dwie ogólne metody: **Get** (do pobierania danych) i **Update** (do dodawania/aktualizacji danych). W parametrach tych metod określamy jaki rodzaj danych chcemy pobrać/aktualizować i która definicja formatu danych XML ma być do tego zastosowana. Integracja z systemem zewnętrznym nie jest więc ograniczona poprzez na sztywno zdefiniowane formaty danych.
enova.Integrator korzysta z tych samych mechanizmów generowania i przetwarzania plików XML, które są zastosowane w enova365.EDI, przy czym nie ma tu żadnego ograniczenia do rodzaju obiektów, które chcemy dodawać/modyfikować. Można więc stosując pewne uproszczenie stwierdzić, że jest to rozszerzone enova365.EDI dostępne przez Web Service.

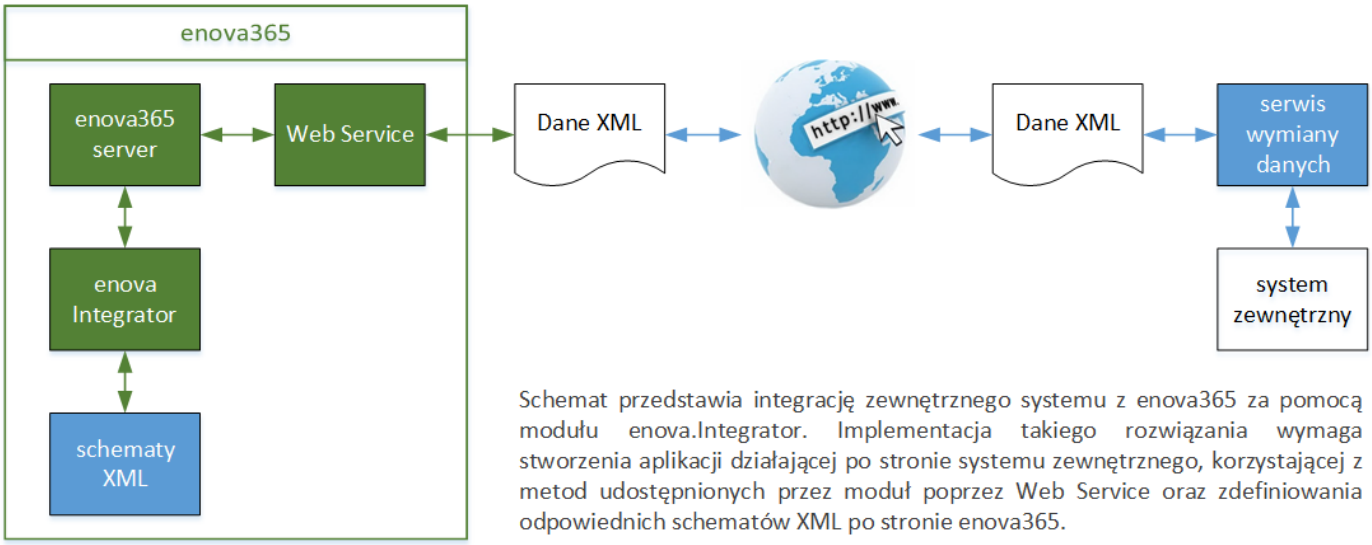
Instalacja

Użycie dodatku jest możliwe w instalacji enova365 multi, zbudowanej do pracy przez interfejs HTML, czyli opartej o enova365 serwer oraz witrynę IIS udostępniającą aplikację. Web Service używany przez enova365.Integrator korzysta z tej samej witryny. Instalacja dodatku polega wyłącznie na wgraniu go do katalogu, z którego będzie ładowany przez enova365 serwer (albo do bazy danych). W funkcjach systemu Windows należy włączyć usługi WCF.
Najprostszym sposobem sprawdzenia, czy instalacja jest gotowa do pracy, jest zalogowanie się do enova365 przez przeglądarkę i przejście na zakładkę \Systemowe\Integrator okna konfiguracji. Jeśli jest widoczna, dodatek jest załadowany.

Budowa dodatku

Dodatek zbudowany jest w oparciu o następujące mechanizmy:

- Usługę sieciową (Web Service) - działającą na serwerze enova365.
- Definicje własnych komunikatów (schematów) XML do wymiany danych. Definiowanie komunikatów według tych samych reguł wykorzystywane jest również w Elektronicznej Wymianie Dokumentów EDI.
- Dwie zdefiniowane metody: Get, Update (wywoływane z odpowiednimi parametrami), służące odpowiednio do pobierania oraz aktualizacji danych.
- Dodatkowe tabele: SystemyZewn (definicje systemów zewnętrznych), ZadaniaZewnSynch (zadania synchronizacji), RelacjeZewn (relacje pomiędzy obiektami lokalnymi i zewnętrznymi). Tabele można wykorzystać, jednak ich użycie nie jest konieczne do działania metod Get i Update.



Schemat przedstawia integrację zewnętrznego systemu z enova365 za pomocą modułu enova.Integrator. Implementacja takiego rozwiązania wymaga stworzenia aplikacji działającej po stronie systemu zewnętrznego, korzystającej z metod udostępnionych przez moduł poprzez Web Service oraz zdefiniowania odpowiednich schematów XML po stronie enova365.

Web Service

Usługa sieciowa uruchamiana jest na serwerze enova365. Jej adres ma postać: **http://SERVER_NAME:SERVER_PORT/Business/MethodInvokerService.svc** i jest on niezbędny do odsłonięcia interfejsu do komunikacji aplikacji klienckiej z serwerem. Dostępność tej usługi i komunikację z nią można sprawdzić wpisując powyższy adres w przeglądarce.

W środowisku .NET z powyższego adresu możemy wygenerować odpowiednie klasy (tzw. klasy proxy), które zapewnią wywołanie metod udostępnianych przez Web Service.

Powyższy Web Service udostępnia jedną metodę o następującej sygnaturze:

InvokeServiceMethodResult InvokeServiceMethod(ServiceMethodInvokerParams invokerParams)

Do metody tej przekazywany jest parametr typu ServiceMethodInvokerParams, w którym określamy, jakiej bazy dotyczy zapytanie oraz jaka operacja docelowa ma zostać wykonana (Get lub Update). Podajemy też parametry, które będą przekazane do wywoływanej metody docelowej, czyli Get albo Update. Są one podawane w postaci słownika z parami klucz (nazwa parametru) - wartość.

```
public class InternalServiceMethodInvokerParams
{
  public string DatabaseHandle { get; set; } // nazwa bazy danych
  public string Operator { get; set; } // operator (login)
  public string Password { get; set; } // hasło operatora
  public string ServiceName { get; set; } // nazwa wywoływanego serwisu: "enova.Integrator.IIntegrator, enova.Integrator"
  public string MethodName { get; set; } // nazwa wywoływanej metody: "Get" albo "Update"
  public Dictionary<string, object> MethodArgs { get; set; } // parametry przekazywane do wywoływanej metody
                                                             // w postaci par (klucz – wartość)
}
```

Rezultatem działania powyższej metody jest obiekt typu InvokeServiceMethodResult, w którym w właściwości ResultInstance zwracany jest obiekt wywołanej metody serwisu (Get, Update). Obiektem tym będzie plik XML. Będzie on zawierał pobrane dane w zdefiniowanym formacie, albo identyfikatory dodanych/zaktualizowanych danych. Strukturę tego pliku przedstawiono w dalszej części opisu.

Definicje komunikatów XML

Definiowanie komunikatów, które mogą zostać wykorzystane przez Integrator, wykonujemy w enova365 w Konfiguracji na zakładce Systemowe->Integrator. Można skorzystać z 7 predefiniowanych komunikatów (Towary, Kontrahenci, DokumentyHandlowe, DokumentyEwidencji, ZadaniaCRM, RaportyESP, DokumentyDMS) lub tworzyć własne. Standardowe definicje obsługują pobieranie danych, ich aktualizację oraz tworzenie nowych zapisów. Pojedyncza definicja komunikatu określa format XML, w jakim zostaną zwrócone dane przy wywołaniu metody Get lub format, w jakim musimy dostarczyć dane do aktualizacji (wywołując metodę Update). Dla dodatku enova365.Integrator znaczenie mają parametry określające TypDanych oraz nazwę zdefiniowanego komunikatu. Te dwa pola są wymagane przy wywołaniu metod dodatku.

W standardowych definicjach zastosowano następującą logikę: każda z nich zawiera pole Identyfikator, w którym podczas eksportu danych umieszczany jest Guid obiektu. Jeśli w importowanym pliku pole to wystąpi i jest wypełnione, mechanizm importu szuka odpowiedniego obiektu i aktualizuje go na podstawie dostarczonych danych. Jeśli w pliku pole Identyfikator nie wystąpi, dodawany jest nowy obiekt.

Szczegółowe informacje na temat definiowania komunikatów można znaleźć w dokumentacji enova.EDI.

Metody Integratora

Dodatek (serwis) enova.Integrator udostępnia dwie metody :

```
public interface IIntegrator
{
  object Get(GetParams pars);
  object Update(UpdateParams pars);
}
```

Pierwsza z nich służy do pobierania danych z enova365. Druga do aktualizacji/dodawania nowych danych zgodnie ze zdefiniowanym schematem XML.

Pobieranie danych

Do pobierania danych służy metoda Get. Do metody przekazana jest klasa parametrów określających parametry pobierania. Jest to lista par typu klucz (nazwa parametru) - wartość.

```
public class GetParams
{
  public string TableName { get; set; } // nazwa tabeli, z której mają być pobrane dane, np. "DokHandlowe", "Towary"
  public string SchemaName { get; set; } // nazwa schematu (definicji) XML, np. "Towary"
  public string Condition { get; set; } // opcjonalny warunek filtrujący pobierane dane, zbudowany zgodnie z regułami stosowanymi do budowania filtrów dla list, np. "Kod = 'AG001'"
  public int RowCount { get; set; } // określa liczbę rekordów do pobrania, domyślnie 100
```

```
  public DateTime LastDate { get; set; } // data modyfikacji ostatniego rekordu.
                        // Jeżeli jest podana, zostanie zwrócona pierwsza paczka danych zmodyfikowanych po zadanym czasie (zwracana w klasie rezultatów)
  public int LastId { get; set; } // ID rekordu zwróconego dotąd jako ostatni ... konieczne dla pobrania kolejnej paczki danych (zostanie zwrócone w klasie rezultatów)

  // Od wersji 12.3
  public int Search { get; set; } // Sposób wyszukiwania danych, zwracanych z metody Get. Opcje: 1, 2, 3 (opis poniżej)
  public string ExternalSystem { get; set; } // Nazwa systemu zewnętrznego, dla którego zostały utworzone dane do synchronizacji
  public string ActionName { get; set; } //  Nazwa akcji zadania do synchronizacji
  public string SchemaParam { get; set; } //  Parametr przekazywany do schematu XML, gdzie jest dostępny poprzez GetFromDict("EXTERNAL_PARAM_Integrator"),
                        // może sterować sposobem działania schematu (definicji)
  // Od wersji 15.0
  public string ResultDataFormat { get; set; }  // Format danych wynikowych: 1 (domyślnie) - kompletny plik XML jako CDATA w GetResult,  2 - tag wkomponowany w GetResult.
}
```

Metoda Get zwraca rezultat w postaci XML-a zgodnego ze schematem http://www.enova.pl/Schemas/GetResult.xsd. Zawiera on dane dotyczące pobranych obiektów w postaci XML-i zawartych w polach typu CDATA. Przykład pliku przedstawiono w dalszej części opisu.

Sposoby wyszukiwania zwracanych danych

Sposób wyszukiwania danych oraz kolejność, w jakiej zostaną pobrane, określa parametr Search w klasie paremetrów GetParams. Przyjmuje on następujące wartości:

- *Search = 1* - (wartość domyślna) dane są pobierane w kolejności domyślnego klucza dla tabeli określonej parametrem TableName.
- *Search = 2* - dane są pobierane według daty ostatniej zmiany zapisanej w historii zmian obiektu (ChangeInfos), w tym przypadku zastosowanie ma dodatkowy parametr LastDate, który określa datę ostatniej zmiany, od której dane mają być pobrane.
- *Search = 3* - zostaną pobrane tylko te dane, które mają wpisy w tabeli zadań synchronizacji (ZadaniaZewnSynch), w tym przypadku zastosowanie mają dodatkowe parametry ExternalSystem , ActionName.

Search = 1 (klucz podstawowy)

enova365.Integrator wyszukuje i zwraca dane z tabeli określonej w parametrze TableName w kolejności wg klucza podstawowego, czyli takiego, który zdefiniowano jako TableName_Podstawowy. Np. dla parametrów *TableName = "Towary", RowCount = 5, Search = 1* system zwróci 5 pierwszych rekordów bezpośrednio z tabeli Towary, w kolejności wg kodu, gdyż tak właśnie zdefiniowano klucz podstawowy dla tej tabeli (Towary_Podstawowy).

ID	Kod	Nazwa	LastChangeInfo
36	AG001	sprzęgło kpl. Ford Capri AGM	16.04.2018 15:55:21
17	AG002	tarcza sprzęgła Ford Capri 180mm AGM	07.03.2018 08:59:04
6	AG003	docisk sprzęgła Ford Capri 180mm AGM	07.03.2018 09:03:08
13	AG004	łożysko wyciskowe Ford Capri AGM	07.03.2018 09:09:56
22	BONUS	rabat okresowy	07.03.2018 09:06:37
12	BS001	klocki hamulcowe prz. Cadilac CTS 2002-2007 BrakeStar	16.04.2018 16:02:14
10	BS002	klocki hamulcowe prz. Buick LaCrosse 2005-2009 BrakeStar	22.06.2017 15:12:02
19	BS003	tarcze hamulcowe prz. 300mm Cadilac CTS 2002-2007 BrakeStar	07.03.2018 09:06:37
37	BX001	sprzęgło kpl. Ford Capri BXM	07.03.2018 09:08:21
18	BX002	tarcza sprzęgła Ford Capri 180mm BXM	07.03.2018 09:09:58

Search = 2 (historia zmian w ChangeInfos)

enova365.Integrator wyszukuje i zwraca dane typu określonego w parametrze TableName, ale mających wskazanie w tabeli historii zmian obiektu. Np. dla parametrów *TableName = "Towary", RowCount = 5, Search = 2*, LastDate = 2018-03-20 system zwróci 5 pierwszych rekordów typu Towary, które zostały zmodyfikowane po 2018-03-20(mają wpis w tabeli ChangeInfos).

13	AG004	łożysko wyciskowe Ford Capri AGM	07.03.2018 09:09:56
14	BX004	łożysko wyciskowe Ford Capri BXM	07.03.2018 09:09:58
18	BX002	tarcza sprzęgła Ford Capri 180mm BXM	07.03.2018 09:09:58
32	DE001	akumulator żelowy 120 Ah 700A DexoElectro	22.03.2018 10:28:01
34	DE002	akumulator żelowy 80 Ah 500A DexoElectro	22.03.2018 10:28:30
15	PK10	przesyłka kurierska do 10 kg	06.04.2018 14:26:53
16	PK20	przesyłka kurierska do 20 kg	06.04.2018 14:27:09
36	AG001	sprzęgło kpl. Ford Capri AGM	16.04.2018 15:55:21
12	BS001	klocki hamulcowe prz. Cadilac CTS 2002-2007 BrakeStar	16.04.2018 16:02:14
1	VM003	pompa paliwa Aston Martin Lagonda	16.04.2018 16:25:36

Search = 3 (zadania synchronizacji)

enova365.Integrator wyszykuje i zwraca dane typu określonego w parametrze TableName ale, mających wskazanie w tabeli zadań do synchronizacji. Np. dla parametrów *TableName = "Towary", RowCount = 5, Search = 3, ExternalSystem="MA_22", ActionName="Add"* system zwróci 5 pierwszych rekordów typu Towary, które zostały oznaczone do synchronizacji dla systemu MA_22 (mają wpis w tabeli ZadaniaZewnSynch).

ID	Akcja	Zapis tabela	System zewn	Data	Zapis
1	Add	Towary	eSklep - MA_22	16.04.2018 16:38:35	001da03b-6aeb-4c83-a7f6-68b66dde0dd7
3	Add	Towary	eSklep - MA_22	16.04.2018 16:38:41	3999a53e-e399-4f95-8a3e-f8769f1e5ec7
5	Add	Towary	eSklep - MA_22	16.04.2018 16:38:46	902d7b14-0ccc-43e7-ab2c-e6e83cdc88a0
7	Add	Towary	eSklep - MA_22	16.04.2018 16:38:53	d85d9473-a731-4ddb-843d-e4e210c108c5
9	Add	Towary	eSklep - MA_22	16.04.2018 16:38:58	97fd9da4-f15d-40e7-9ed9-03bf720b3321
11	Add	Towary	eSklep - MA_22	16.04.2018 16:39:04	9fe6f27d-ac82-4920-85f4-cb75c4fcb1be
13	Delete	Towary	eSklep - MA_22	16.04.2018 16:40:13	0caf46b9-2af3-4a62-ab03-1a7a9d2d201c
15	Add	Towary	eSklep - MA_22	16.04.2018 16:40:29	14fb9799-266c-4496-b2ac-5e7d6fc6535b
17	Update	Towary	eSklep - PR_01	16.04.2018 16:41:26	9eb0fbd7-3b7a-4799-9244-a4e7dd4b04dd
19	Add	Towary	eSklep - MA_22	16.04.2018 16:41:33	bede8458-a7e7-4aea-b7eb-221c5ac40f1a

Usuwanie zadań synchronizacji

Jeśli pobieramy dane według zadań synchronizacji, to po ich pobraniu należy usunąć odpowiednie zadania synchronizacji. Powinien to zrobić ten sam serwis, który pobiera dane. Można to zrobić za pomocą metody Update wykonanej dla tabeli ZadaniaZewnSynch ze wskazaniem służącego do tego schematu XML PoSynchronizacji. Działanie tej definicji polega na usunięciu zadań zidentyfikowanych na podstawie guidów obiektów, których dotyczyły oraz na utworzeniu relacji pomiędzy zsynchronizowanymi obiektami, czyli utworzeniu zapisów w tabeli RelacjeZewn.

Przykłady

Poniżej przedstawiono przykład użycia metody Get do pobrania danych.

```
using (MethodInvokerServiceClient myClient_ = new MethodInvokerServiceClient())
{
  var Params = new ServiceMethodInvokerParams
```

```
    {
      DatabaseHandle = "enova_demo",
      Operator = "Integrator",
      Password = "enova",
      ServiceName = "enova.Integrator.IIntegrator, enova.Integrator",
      MethodName = "Get",
      MethodArgs = new Dictionary<string, object>
      {
        { "TableName", "Towary" },
        { "SchemaName", "Towary" },
        { "Search", 1 },
        { "Condition", "Kod='AG001'" /*nie musimy podawać*/},
        { "RowCount", 20  /*nie musimy podawać, domyślnie = 100*/ },
        { "LastDate", "" /*potrzebna, gdy pobieramy zmienione dane paczkami*/ },
        { "LastId", 0  /*potrzebne, gdy pobieramy dane paczkami*/ }
      }
    };

    InvokeServiceMethodResult result = myClient_.InvokeServiceMethod(Params); // rezultat (obiekt), z danymi o wyjątkach itp ...
    var xmlResult = result.ResultInstance; // to już jest konkretny rezultat (XML) zwrócony przez wywołaną metodę Get;
 }
```

Metoda zwróci pobrane dane w postaci pliku XML (zawartego w result.ResultInstance):

```
<?xml version="1.0" encoding="utf-16"?>
<GetResult>
  <LastRow>
    <Id>22</Id>
    <Date>0001-01-01T00:00:00</Date>
    <End>false</End>
  </LastRow>
  <Rows>
    <Row>
    <Xml><![CDATA[<?xml version="1.0" encoding="utf-8"?>
     <Towar>
      <Identyfikator>9eb0fbd7-3b7a-4799-9244-a4e7dd4b04dd</Identyfikator>
      <Typ>Produkt</Typ>
      <Kod>AG001</Kod>
      <Nazwa>sprzęgło kpl. Ford Capri AGM</Nazwa>
      <Jednostka>szt</Jednostka>
      (... kolejne pola...)
     </Towar>]]>
    </Xml>
    </Row>
    <Row>
    <Xml><![CDATA[<?xml version="1.0" encoding="utf-8"?>
     <Towar>
       (... pola...)
     </Towar>]]>
    </Xml>
    </Row>
  </Rows>
</GetResult>
```

Sekcja LastRow zawiera dane o ostatnim zwróconym rekordzie, sekcja Rows zawiera dane pobranych obiektów.

Poniżej przedstawiono przykład pobrania wszystkich towarów w pakietach po 20. W pętli sprawdzamy ID ostatniego rekordu, który został zwrócony (LastRow/ID) oraz parametr mówiący o tym, czy pobrano już wszystkie rekordy (LastRow/End).

```
var client = new Service.MethodInvokerServiceClient();
var methodParams = new Service.ServiceMethodInvokerParams()
{
  DatabaseHandle = "enova_demo",
  Operator = "Integrator",
  Password = "enova",
  ServiceName = "enova.Integrator.IIntegrator, enova.Integrator",
  MethodName = "Get",
  MethodArgs = new Dictionary<string, object> {
    {"TableName", "Towary" },
    { "SchemaName", "Towary" },
    { "Search", 1 },
    { "Condition", "" },
    { "RowCount", 20 },
    { "LastId", 0 }
  }
};

var end = false;
do
{
  var methodResult = client.InvokeServiceMethod(methodParams);
  var result = methodResult.ResultInstance as string;       // tutaj mamy rezultat, który możemy przetworzyć
  XmlDocument xd = new XmlDocument();
  xd.LoadXml(result);
  methodParams.MethodArgs["LastId"] = int.Parse(xd.SelectSingleNode("GetResult//LastRow//Id").InnerText);
  end = bool.Parse(xd.SelectSingleNode("GetResult//LastRow//End").InnerText);
}
while (!end);
```

W podobny sposób można pobrać wszystkie towary zmienione w ostatnim miesiącu. Zamiast LastId użyjemy parametru LastDate.

```
  // pierwsza wartość
```

```
  MethodArgs = new Dictionary<string, object> {
    ...
    { "LastDate", DateTime.Today.AddMonths(-1) }
  }

  // zmiana w pętli
  methodParams.MethodArgs["LastDate"] = int.Parse(xd.SelectSingleNode("GetResult//LastRow//Date").InnerText);
```

Aktualizacja/Dodawanie danych

Do dodawania/aktualizacji danych służy metoda Update. Do metody przekazywana jest klasa parametrów zawierająca plik XML z danymi do przetworzenia.

```
public class UpdateParams
{
  public string TableName { get; set; } // nazwa tabeli
  public string SchemaName { get; set; } // nazwa schematu XML
  public string Data { get; set; } // dane w postaci XML, które zostaną przetworzone zdefiniowanym komunikatem (schematem)

  // Od wersji 12.3
  public string ExternalSystem { get; set; } // Parametr przekazywany do schematu XML, gdzie jest dostępny poprzez GetFromDict("EXTERNAL_PARAM_ExternalSystem"),
                                             // przekazuje informację, którego systemu zewnętrznego dotyczy żądanie
  public string SchemaParam { get; set; }    // Parametr przekazywany do schematu XML, gdzie jest dostępny poprzez GetFromDict("EXTERNAL_PARAM_Integrator"),
                                             // może sterować sposobem działania schematu (definicji)

  // Od wersji 15.0
  public string ResultSchemaName { get; set; } // Parametr przekazywany do schematu XML. Jeśli jest podany, wynik zwracany przez metodę Update będzie zawierał dane utworzonego
                                               // lub zmienionego obiektu, wygenerowane za pomocą wskazanego schematu. Będą one umieszczone w tagu <Xml> wynikowego pliku.
                                               // Podanie parametru pozwala na pobranie danych utworzonego obiektu wraz z informacją o wykonaniu metody Update
}
```

Parametr Data (dane do aktualizacji) przesyłany jest w postaci XML-a zgodnego ze schematem http://www.enova.pl/Schemas/UpdateParamsData.xsd.
Poniżej przedstawiono przykład dodania 2 towarów z wypełnieniem kodu i nazwy.

```
using (MethodInvokerServiceClient myClient_ = new MethodInvokerServiceClient())
{
  var Params = new ServiceMethodInvokerParams
  {
    DatabaseHandle = "enova_demo",
    Operator = "Integrator",
    Password = "enova",
    ServiceName = "enova.Integrator.IIntegrator, enova.Integrator",
    MethodName = "Update",
    MethodArgs = new Dictionary<string, object>
    {
      { "TableName", "Towary" },
      { "SchemaName", "Towary" },
      { "Data", @"<?xml version=""1.0"" encoding=""utf-16""?>
      <UpdateParamsData>
        <Rows>
          <Row>
            <Xml><![CDATA[<?xml version=""1.0"" encoding=""utf-8""?>
              <Towar>
                <Kod>AG100</Kod>
                <Nazwa>sprzęgło kpl. Jaguar Sovereign AGM</Nazwa>
                <Jednostka>szt</Jednostka>
              </Towar>]]></Xml>
          </Row>
          <Row>
            <Xml>(dane drugiego towaru...)</Xml>
          </Row>
        </Rows>
      </UpdateParamsData>"  }
    }
  };
  InvokeServiceMethodResult result = myClient_.InvokeServiceMethod(Params); // rezultat (obiekt), z danymi o wyjątkach itp ...
  var xmlResult = result.ResultInstance; // to już jest konkretny rezultat (XML) zwrócony przez wywołaną metodę Update;
}
```

Metoda Update zwraca rezultat w postaci XML-a zgodnego ze schematem http://www.enova.pl/Schemas/UpdateResult.xsd. Poniższy przykład przedstawia wynik po przetworzeniu dwóch obiektów, z pustym tagiem Xml, czyli w przypadku, gdy w parametrach metody nie podano ResultSchemaName.

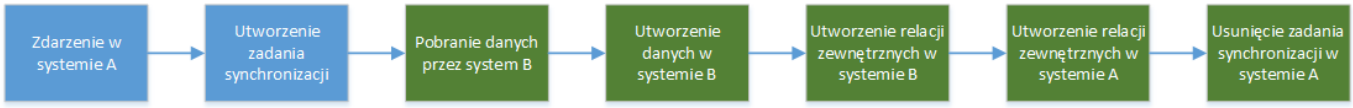
```
<?xml version="1.0" encoding="utf-16"?>
<UpdateResult>
 <Rows>
  <Row>
   <No>1</No>
   <ID>24</ID>
   <Guid>f26aa399-b520-4865-8fbc-f132b20921e1</Guid>
   <Xml><![CDATA[]]></Xml>
   <Message ></Message>
   <MessageType>Info</MessageType>
  </Row>
  <Row>
   <No>2</No>
   <ID>25</ID>
   <Guid>40c97751-2a36-486d-a795-0f8b76534eaf</Guid>
   <Xml><![CDATA[]]></Xml>
   <Message ></Message>
   <MessageType>Info</MessageType>
  </Row>
```

```
 </Rows>
</UpdateResult>
```

Sekcja Row zawiera numer kolejny dodanego/zaktualizowanego obiektu (zgodny z kolejnością w pliku XML przesłanym jako parametr metody Update), jego Guid, ID oraz komunikat o błędzie, jeśli taki wystąpił.

Pełny proces pobrania i zapisania danych

Proces pobierania danych najlepiej zorganizować z wykorzystaniem definicji systemów zewnętrznych, zadań synchronizacji i relacji zewnętrznych. Będzie on wtedy przebiegać według następującego schematu:



- Zdarzenie w źródłowym systemie (np. wystawienie dokumentu) powoduje powstanie zadania synchronizacji. Tworzenie zadania należy oprogramować we własnym zakresie, np. za pomocą tasków.
- System zewnętrzny pobiera dane objęte zadaniami synchronizacji za pomocą wykonanej zdalnie metody Get.
- System zewnętrzny wykonuje zapis pobranych danych w swojej bazie. Jeśli jest to instalacja enova365, wykonuje to za pomocą wykonanej lokalnie metody Update, do której przekazuje pobrane dane w parametrze Data.
- System zewnętrzny tworzy w swojej bazie zapisy wiążące obiekty w obu systemach. Jeśli jest to enova365, wykonuje zapisy w tabeli RelacjeZewn.
- System zewnętrzny usuwa zadania synchronizacji w instalacji źródłowej, wykonując zdalnie metodę Update. Usuwanie zadań tworzy jednocześnie zapisy w tabeli RelacjeZewn w systemie źródłowym.

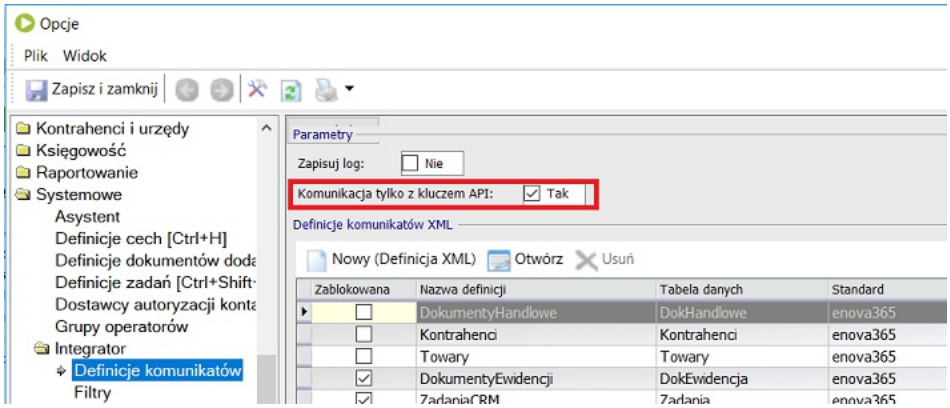
Według opisanego powyżej modelu działa standardowa czynność **Systemy zewnętrzne/Pobieranie danych**, dostępna od wersji 15.3 w menu głównym systemu z aktywnym modułem enova365.Integrator, służąca do pobierania danych z innej instalacji enova365.

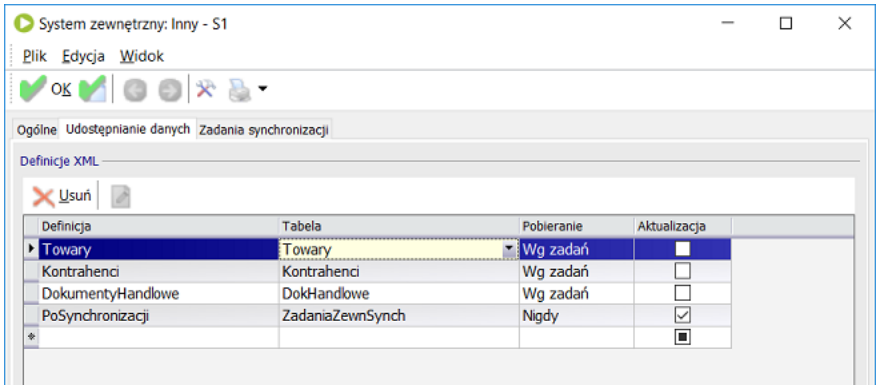
Ograniczenie dostępu do danych

Uruchomienie enova365.Serwer z modułem Integrator i zdefiniowaną witryną sieci Web oznacza udostępnienie danych dla aplikacji zewnętrznych, które mogą je pobierać za pomocą metody Get. Oczywiście pobranie danych z zewnątrz jest możliwe wtedy, gdy aplikacji zewnętrznej udostępnimy operatora (podając login i hasło), na którego może się ona logować do bazy. Drugim warunkiem koniecznym do pobrania danych jest istnienie odblokowanej definicji XML, która takie dane wygeneruje. Dostęp do danych jest standardowo chroniony prawami operatora, na którego loguje się aplikacja zewnętrzna.

Od wersji 15.3 enova365 udostępnia dodatkowe narzędzia, które pozwalają w przejrzysty sposób skonfigurować dostęp do danych za pomocą enova365.Integrator. Wykorzystywana jest do tego definicja systemu zewnętrznego. Konfigurację wykonujemy w następujący sposób:

- Włączamy parametr **Komunikacja tylko z kluczem API**. Teraz w każdym żądaniu pobrania/aktualizacji danych w parametrze *ExternalSystem* musi zostać przekazany klucz API, dostarczony przez stronę udostępniającą dane. Przed wykonaniem żądania system zweryfikuje poprawność tego klucza w sposób opisany dalej. Przy takim ustawieniu możliwe będzie pobieranie wyłącznie tych danych, które jawnie określono w konfiguracji systemu udostępniającego dane.
- Definiujemy system zewnętrzny, który odpowiada instalacji pobierającej dane za pomocą enova365.Integrator. Włączamy parametr **Udostępnianie danych**, co aktywuje zakładkę o takiej samej nazwie. Po włączeniu parametru staje się widoczne pole **Identyfikator własny** (tylko do odczytu), w którym wyświetlany jest guid zdefiniowanego systemu. To on właśnie jest kluczem API, który udostępniamy stronie pobierającej dane. Klucz jest więc indywidualny dla każdej zewnętrznej instalacji, której odpowiada osobna definicja systemu zewnętrznego.
- Konfigurujemy dostęp do danych na zakładce **Udostępnianie danych**. Wiersz na tej zakładce udostępnia dane z określonej tabeli, w postaci generowanej wskazanym schematem. Określa też dozwolony tryb pobierania danych, odpowiadający wartości parametru *Search*, czyli **Zawsze** (*Search* = 1,2 3), albo tylko **Wg zadań** (*Search* = 3). Inaczej mówiąc, wiersz zawiera zestaw dozwolonych parametrów metody *Get*.





Przy takiej konfiguracji logika przetwarzania żądania pobrania danych będzie następująca:

- System sprawdza, czy zdefiniowano system zewnętrzny o guidzie zgodnym z kluczem API przesłanym w parametrze *ExternalSystem*.
- Jeśli nie, odrzuca żądanie jako nieuprawnione.
- Jeśli tak, system analizuje parametry żądania: nazwę tabeli, nazwę schematu i tryb pobierania. Sprawdza w odszukanej po kluczu API definicji systemu zewnętrznego, czy

taki zestaw parametrów jest dozwolony, to znaczy, czy ma swoją reprezentację w postaci wiersza na zakładce **Udostępnianie danych**.
- Jeśli zestaw parametrów jest dozwolony, żądanie zostaje wykonane.

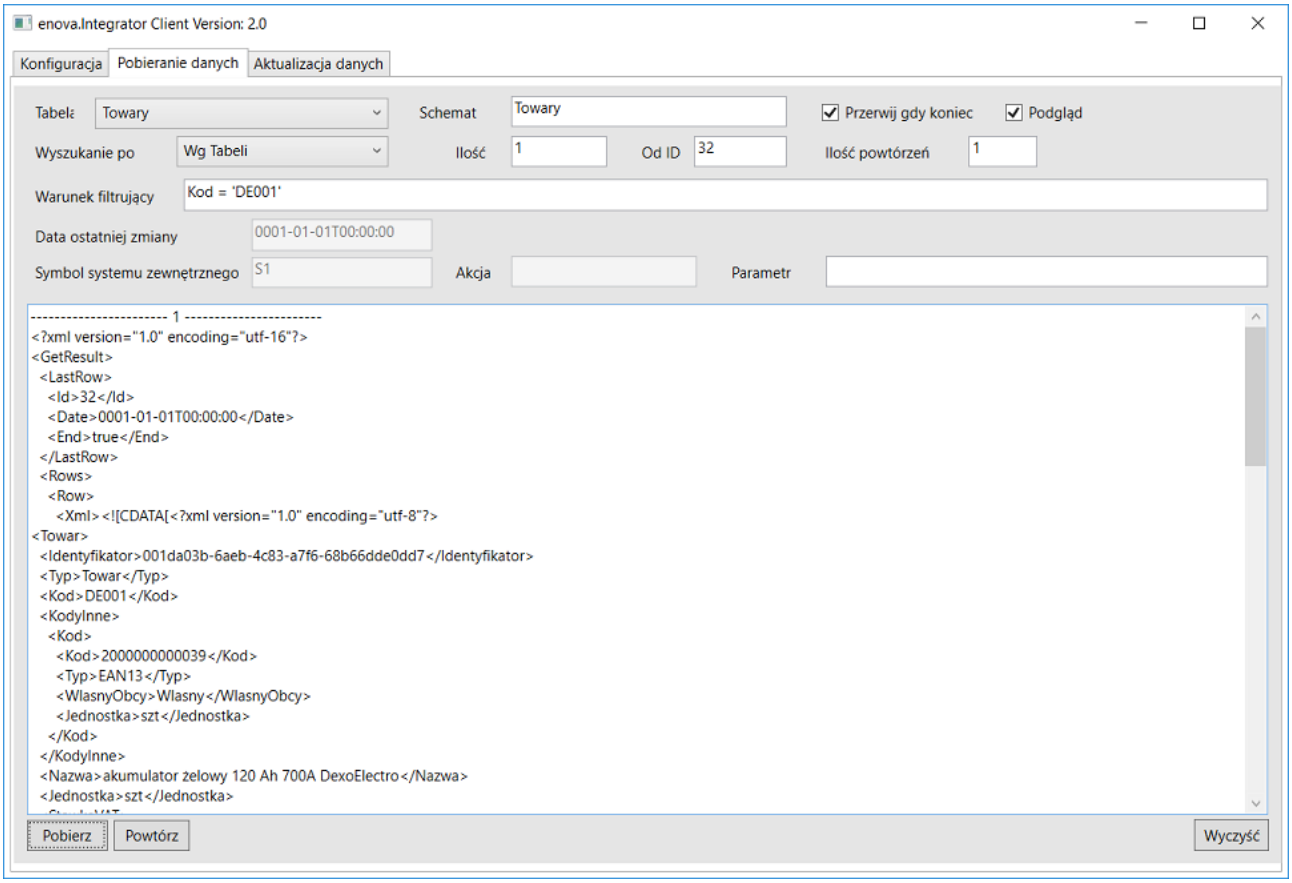
Przykład SOAP POST

Poniżej przedstawiono przykład pełnej struktury wysyłanej za pomocą SOAP 1.1.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <InvokeServiceMethod xmlns="http://tempuri.org/">
   <invokerParams xmlns:a="http://schemas.datacontract.org/2004/07/Soneta.Net.Types" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <ClientAddress i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types" ></ClientAddress>
    <ContextHandle i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types" ></ContextHandle>
    <DatabaseHandle xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types">Moto_150</DatabaseHandle>
    <IpFilters i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types" ></IpFilters>
    <MethodArgs xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types" xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
     <b:KeyValueOfstringanyType>
      <b:Key>Search</b:Key>
      <b:Value i:type="c:int" xmlns:c="http://www.w3.org/2001/XMLSchema">1</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>TableName</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">Towary</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>SchemaName</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">Towary</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>Condition</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema" ></b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>RowCount</b:Key>
      <b:Value i:type="c:int" xmlns:c="http://www.w3.org/2001/XMLSchema">10</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>ActionName</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema" ></b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>ExternalSystem</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema" ></b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>LocalRequest</b:Key>
      <b:Value i:type="c:boolean" xmlns:c="http://www.w3.org/2001/XMLSchema">false</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>LastDate</b:Key>
      <b:Value i:type="c:dateTime" xmlns:c="http://www.w3.org/2001/XMLSchema">0001-01-01T00:00:00</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>LastId</b:Key>
      <b:Value i:type="c:int" xmlns:c="http://www.w3.org/2001/XMLSchema">18</b:Value>
     </b:KeyValueOfstringanyType>
     <b:KeyValueOfstringanyType>
      <b:Key>SchemaParam</b:Key>
      <b:Value i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema" ></b:Value>
     </b:KeyValueOfstringanyType>
    </MethodArgs>
    <MethodName xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types">Get</MethodName>
    <Operator xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types">Integrator</Operator>
    <Password xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types">enova</Password>
    <ServiceName xmlns="http://schemas.datacontract.org/2004/07/Soneta.Types">enova.Integrator.IIntegrator, enova.Integrator</ServiceName>
    <a:ConnectionInfo i:nil="true" ></a:ConnectionInfo>
   </invokerParams>
  </InvokeServiceMethod>
 </s:Body>
</s:Envelope>
```

Program testowy (klient)

Działanie opisanych metod można testować za pomocą aplikacji enova.Integrator.Client zawartej w pakiecie. Umożliwia ona pobieranie i aktualizację danych, prezentuje tworzone pliki XML i umożliwia ich edycję.



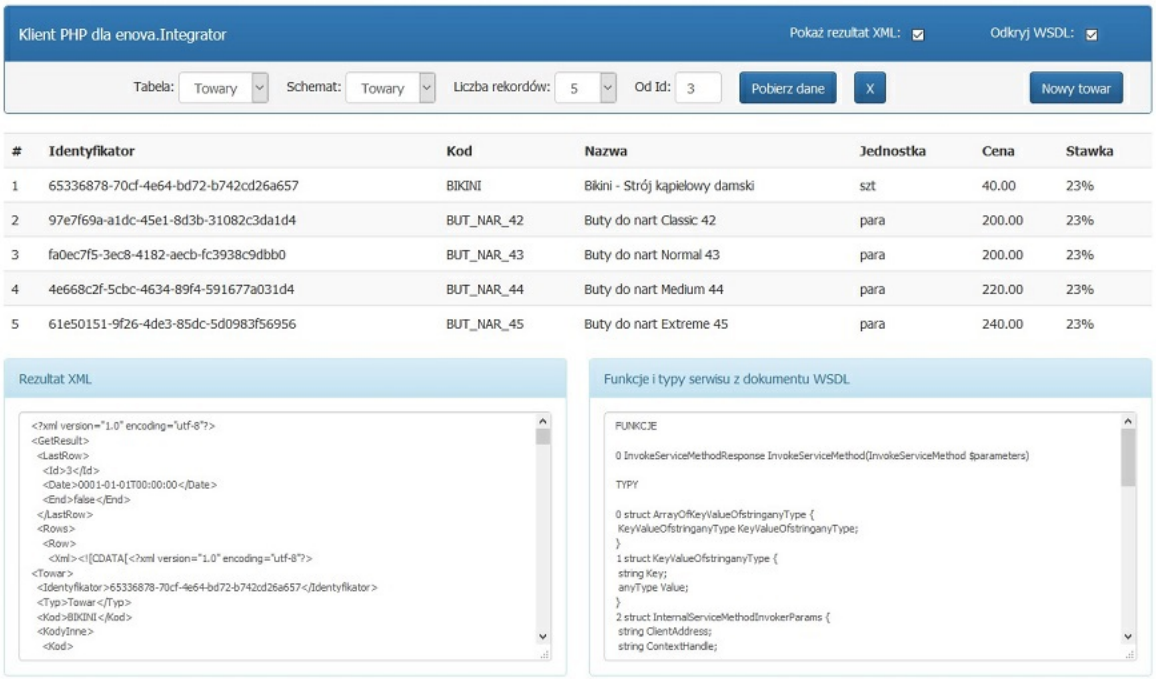
Przykładowe aplikacje

Przykład Java

Przykład aplikacji w języku Java korzystającej z enova365.Integrator można znaleźć tutaj. Jej działanie polega na pobraniu jednego towaru za pomocą metody Get, a następnie na zmianie jego nazwy za pomocą metody Update.

Przykład PHP

Przykład aplikacji w języku PHP korzystającej z enova365.Integrator można znaleźć tutaj. Jej działanie polega na pobraniu wskazanej ilości towarów za pomocą metody Get i wyświetleniu pobranych danych na liście w oknie aplikacji.



Przykład tworzenia aplikacji

Przykład tworzenia aplikacji korzystającej z enova365.Integrator przedstawiono na III Konferencji Deweloperskiej Soneta 2016. Nagranie z prezentacji można znaleźć tu: https://vimeo.com/172720412/79e504f950 .

Licencja

Użycie enova365.Integrator wymaga licencji na ten dodatek. Instalacja musi też mieć jakąkolwiek licencję multi, dającą dostęp do serwera aplikacji poprzez witrynę IIS. enova365.Integrator nie pobiera licencji modułowych. W związku z tym do logowania się do enova365 najlepiej zdefiniować operatora z przypisaną rolą Harmonogram zadań, czyli bez przypisanych modułów.

Dokumentacja w języku angielskim

pobierz >

Pomoc techniczna - 12 34 92 810, techniczne@enova.pl
Pomoc Kadry Płace - 12 34 92 820, place@enova.pl
Pomoc Księgowość - 12 34 92 830, ksiegowosc@enova.pl
Pomoc Handel - 12 34 92 840, handel@enova.pl
Pomoc CRM - 12 34 92 850, crm@enova.pl
Pomoc Workflow & DMS - 12 34 92 860, workflow@enova.pl
Pomoc BI - 12 34 92 865, BI@enova.pl

Soneta Sp.z o.o.
ul. Wadowicka 8A, 30-415 Kraków,
tel. 12 34 92 800,
e-mail: info@enova.pl,
www.enova.pl