UNION SYSTEMS 2000 Sp. z o. o.
00-324 Warszawa ul. Karowa 18a/20
NIP: 525-20-78-955
tel. +48-prefix-22-8283137
e-mail: union@union.com.pl

Warszawa, 2009-08-21

Dokumentacja techniczna modułu dostępu do bazy danych środowiska programu Synapsa: SynapsaRMA. Opis biblioteki SynapsaCtrlX i SynapsaWebService należących do SynapsaRMA.

1 Zastosowanie modułu

Moduł **SynapsaRMA** służy do programowego dostępu do bazy danych środowiska programu serwisowego Synapsa.

Głównym zadaniem modułu jest udostępnienie funkcji na potrzeby środowiska realizującego serwisową procedurę RMA przez Internet.
Innym zadaniem realizowanym za pomocą modułu jest tworzenie niestandardowych raportów i wydruków w zewnętrznym środowisku, np. Acces, Excel, oraz dowolne inne programy obsługujące standard bibliotek programistycznych COM/ActiveX.
Jednym z zadań realizowanych przy pomocy kontrolki jest automatyzacja wymian informacji między środowiskiem programu handlowo-magazynowego a programem Synapsa (serwisowe przydzielenia i wydania towarów i faktur korygujących z magazynu zewnętrznego).
Kontrolka dostępu wykorzystywana jest także przez Moduł Powiadomień do Synapsy.

2 Wybór sposobu korzystania z modułu

Biblioteki modułu RMA pozwalają na programowy wysokopoziomowy oraz niskopoziomowy dostęp do bazy programu Synapsa.

UWAGA: Korzystanie z modułu może spowodować nieoczekiwane rezultaty w bazie danych. Przed skorzystaniem z modułu należy przetestować tworzone procedury w środowisku nieprodukcyjnym. Użycie modułu powinno być ograniczone do sieci w obszarze zaufanym.

2.1 Dostęp wysokopoziomowy przez bibliotekę SynapsaCtrlX

Dostęp wysokopoziomowy realizowany jest przez zestaw wygodnych w użyciu funkcji. W środowiskach programistycznych korzystanie z kontrolki jest ułatwione przez funkcje podpowiadania metod i właściwości klas.

2.2 Dostęp niskopoziomowy przez bibliotekę SynapsaCtrlX

Dostęp niskopoziomowy pozwala na korzystanie z bardzo uproszczonej składni języka podobnego do SQL, operującej na „widokach” bazy danych, poprzez zestawy rekordów pobranych z Synapsy. Widoki te służą do podglądania. Zastosowanie jest mniej wygodne i mniej bezpieczne dla bazy danych. Wymagana jest znajomość nazw kolumn w widokach oraz dokładnej ich interpretacji. Za pomocą dostępu niskopoziomowego realizowana jest też edycja rekordów Synapsy, ale możliwość ta zastrzeżona jest tylko dla potrzeb wewnętrznych firmy Union Systems.

2.3 Dostęp przez SynapsaWebService

SynapsaWebService służy do udostępniania metod biblioteki SynapsaCtrlX w środowisku sieciowym.

UWAGA: Poprzez WebService dostępne są wyłącznie metody wysokopoziomowe.

WebService jest standardem opartym o protokół **SOAP**. Standard ten obsługiwane jest przez wiele środowisk programistycznych i systemów operacyjnych.
Głównym zadaniem WebService'u jest udostępnienie metod w celu realizacji Modułu Internetowego oferowanego przez Union Systems, działającego w środowisku Windows i Linux, pisanego w języku PHP.
WebService może być też wykorzystany przy opracowywaniu własnego modułu internetowego, napisanego w dowolnym języku (PHP, ASP, ASP.NET i innych).

WebService pozwala na scentralizowanie usługi dostępu. Biblioteka jest tylko w jednym miejscu, co ułatwia czynności instalacyjne i administracyjne.
Używanie WebService nie wymaga instalowania bibliotek modułu SynapsaRMA na maszynie, która zdalnie wywołuje metody.

3 Wykorzystanie modułu SynapsaRMA na potrzeby serwisu internetowego z procedurą RMA

Jednym z głównych zadań modułu jest udostępnienie funkcji dla modułu internetowego, w celu realizacji procedury RMA.

Procedura RMA (ang. Return to Manufacturer Assignment) stosowana jest przez dużych dostawców podzespołów i formalizuje proces przyjmowania towarów do serwisu. Proces ten przebiega w następujący sposób:

- Klient zgłasza serwisowi listę uszkodzonych produktów telefonicznie, pocztą elektroniczną lub przez Internet.
- Serwis ręcznie lub automatycznie akceptuje listę i przydziela numer RMA.
- Klient wysyła do serwisu paczkę oznaczoną numerem RMA.
- Serwis przyjmuje paczkę, weryfikując zgodność zawartości ze zgłoszeniem RMA. System wiąże operację przyjęcia ze zgłoszeniem RMA, dzięki czemu otrzymuje wszystkie informacje o towarach, ich uszkodzeniach, zgłaszającym i inne.

Union Systems oferuje kompletny Moduł Internetowy, zrealizowany w języku PHP, działający w środowisku Windows i Linux.

Za pomocą Modułu Dostępu można tez zrealizować własny Moduł Internetowy, jeśli funkcjonalność modułu oferowanego przez Union Systems okaże się niewystarczająca.

Do utworzenia modułu Internetowego konieczna jest znajomość funkcjonalności serwisu RMA oferowanego przez Union Systems:

3.1 Funkcjonalność serwisu internetowego RMA od strony klienta

1) zgłaszanie towarów przez Internet, w tym:
   - sprawdzanie danych o numerze seryjnym i kopiowanie tych danych do zgłoszenia
   - wyszukiwanie numerów seryjnych z podanej faktury zakupu i kopiowanie wybranego numeru seryjnego do zgłoszenia
   - możliwość zgłaszania towarów bez numeru seryjnego, z ręcznym wyborem towaru z listy i wpisywaniem pozostałych danych
   - dodatkowo możliwość zgłaszania towaru o kodzie nie znajdującym się w kartotece towarów
   - automatyczne nadawanie numeru zgłoszenia, numerowanie każdej pozycji zgłoszenia,
   - automatyczne rozpatrzenie zgłoszenia przez system zaraz po zapisaniu zgłoszenia (na podstawie podanych w zgłoszeniu danych) lub pozostawienie zgłoszenia do ręcznego rozpatrzenia przez serwisantów
2) sprawdzanie statusu zgłoszeń, w tym dane o przetwarzaniu serwisowym towaru:
   - wyszukiwanie zgłoszeń wg numeru zgłoszenia, numeru RMA, wg daty i innych cech
   - sprawdzanie, czy zgłoszenie zostało rozpatrzone (stan R)
   - sprawdzanie, czy rozpatrzenie jest pozytywne (zezwolenie na wysyłkę towaru) czy negatywne (odrzucenie zgłoszenia z podaniem przyczyny - stan X)
   - sprawdzanie, czy wysłany przez klienta towar jest już przyjęty przez firmę (pomiędzy pozytywnym rozpatrzeniem a przyjęciem towaru zgłoszenie ma stan C - oCzekiwanie na przyjęcie)
   - sprawdzanie, czy wysłany przez klienta towar jest gotowy do zwrotu (naprawiony lub przydzielony inny towar)
   - sprawdzanie, czy wysłany przez klienta towar jest oddany (wysłany paczką lub odebrany na miejscu)

3.2 Funkcjonalność RMA od strony serwisu

1) rozpatrywanie zgłoszeń przez serwisantów
    - wyszukiwanie nowych zgłoszeń wg różnych cech
    - sprawdzanie danych o zgłaszanym towarze i rozpatrzenie zgłoszenia (nadanie stanu R)
    - nadawanie typu rozpatrzenia na podstawie tych danych - pozytywnego (domyślnie - nadanie statusu C) lub negatywnego (nadanie statusu X)
    - wpisywanie uwag dla klienta - z prośbą o zmianę lub uzupełnienie danych lub podające powód odrzucenia zgłoszenia
    - automatyczne nadawanie numeru rozpatrzenia zgłoszenia (nr RMA) powiązanego do całego lub do części zgłoszenia. Serwis może też wywołać automatyczne pozytywne rozpatrzenie zgłoszenia, jeśli spełnia ono wybrane przez siebie kryteria
2) przyjęcie towaru na podstawie numeru zgłoszenia
    - wyszukiwanie zgłoszeń rozpatrzonych na podstawie numeru RMA
    - przyjęcie towaru ze zgłoszenia, skopiowanie danych ze zgłoszenia do rewersu (status oczekiwania na przyjęcie zostaje usunięty)
3) zwykła procedura serwisowa
    - status rewersu, w tym status naprawy może być teraz przeglądany łącznie ze zgłoszeniami, dzięki ścisłym powiązaniu pomiędzy rewersem a zgłoszeniem.

4 Instalacja komponentu Synapsa RMAX

Komponent umożliwia współpracę Synapsy z zewnętrznym oprogramowaniem przez protokół ActiveX lub SOAP.

Uwaga: Zalecane jest, aby komponent ten zainstalować na tym samym serwerze, na którym znajduje się bazą danych Synapsy.

Do pracy przez protokoł SOAP (wymagany dla modułu internetowego) niezbędna jest instalacja komponentu Microsoft Soap Toolkit 3.0, instalacja serwisu IIS i wystawienie usługi (WebService) przez IIS. Instalator pomaga w tych krokach, choć dla lepszej kontroli zalecana jest ręczna instalacja.

4.1 Instalacja automatyczna

- Wywołać skrypt setup.vbs
- Wybrać zakres instalacji – opcjonalnie można zainstalować wyłącznie biblioteki dostępu SynapsaRMA lub instalację pełną - biblioteki razem z SynapsaWebService (wymagane dla Modułu Internetowego).
- W przypadku wyboru instalacji WebService'u należy upewnić się, że w systemie zainstalowana jest usługa IIS (Internet Information Services). Należy zatwierdzić też instalację Microsoft Soap Toolkit 3.0.
- Jeśli moduł wykorzystujący SynapsaWebService (w tym panel serwisowy WWW) zainstalowany jest na innym komputerze, należy wyedytować plik SynapsaRMAX.WSDL i w linii:
  <soap:address location='http://localhost/SynapsaRMAX/SynapsaRMAX.ASP'/>
  zamienić *localhost* na nazwę hosta (nazwę serwera z zainstalowaym SynapsaWebService).

Instalator przegra i zarejestruje biblioteki modułu do katalogu C:\Windows\System32, w katalogu systemowym utworzy też plik client.config z adresem serwera Synapsy.
Przy (opcjonalnej) instalacji WebService'u zainstalowana zostanie usługa w IIS o nazwie SynapsaRMAX, gotowa do wykorzystania przez SOAP.

Uwaga: instalacja automatyczna SynapsaWebService może nie powieść się w niektórych środowiskach. Należy wtedy wykonać instalację ręczną.

4.2 Instalacja ręczna bibliotek modułu SynapsaRMA

- Skopiować client.config, zawierający adres serwera Synapsy (zwykle localhost), do katalogu C:\Windows\System32.
- Instalacja komponentu ActiveX: Przekopiować plik SynapsaCtrl.dll do C:\Windows\system32. Przekopiować plik SynapsaCtrlX.dll do C:\Windows\system32 i zarejestrować poleceniem:
  regsvr32.exe „C:\Windows\SynapsaCtrlX.dll”

Uwaga: SynapsaCtrl.dll (biblioteka zwykła) musi być w katalogu systemowym, aby była dostępna dla WebService. SynapsaCtrlX.dll (COM/ActiveX) może być skopiowana i zarejstrowana w innym katalogu (np. C:\Program Files\Union Systems).

Komponent Synapsa RMA powinien już działać jako komponent ActiveX. Należy go przetestować za pomocą skryptu *example_FindReverses.vbs* (wybrać opcję COM).

4.3 Instalacja ręczna modułu SynapsaWebService (moduł wymagany dla aplikacji internetowych)

- Upewnić się, że została wykonana instalacja bibliotek modułu SynapsaRMA (wg poprzednio opisanych kroków) i przetestowano ich działanie.
- Instalacja IIS: W instalatorze Windows dodać składnik „Internet Information Services” (o ile nie był wcześniej zainstalowany).
- Zainstalować Soap Toolkit 3.0 (soapsdk.exe), dołączony do instalacji.
- Wystawienie usługi WWW: Wywołać narzędzie IIS Manager z panelu zarządzania i dodać nowy katalog wirtualny o nazwie SynapsaRMAX wskazujący na katalog soap (dołączony do instalacji). Katalog „soap” można skopiować do C:\inetpub\wwwroot\SynapsaRMAX lub innego katalogu.
- Należy przetestować, czy serwis działa, wywołując w przeglądarce URL:
  http://localhost/SynapsaRMAX/SynapsaRMAX.WSDL
  Powinna pojawić się zawartość pliku w formacie XML.
  W przypadku Windows Server 2003 może zajść potrzeba dodatkowych zabiegów konfiguracyjnych. Należy zezwolić na uruchamianie skryptów ASP w zakładce IIS Web Service Extensions oraz dodać rozszerzenia .wsdl i .wsml do rozpoznawanych MIME types. Poniżej opis tej ostatniej czynności dla wersji angielskiej:
  1. Kliknąć prawym przyciskiem myszy na default web site i wybrać Properties.
  2. Kliknąć zakładkę HTTP Headers.
  3. Nacisnąć MIME Types na dole strony.
  4. Kliknąć New.
  5. Wpisać .wsml w polu Extension.
  6. Wpisać text/xml w polu MIME Type.
  7. Kliknąć OK. Analogiczne kroki wykonać dla rozszerzenia .wsdl.
- Jeśli Panel serwisowy WWW zainstalowany jest na innym komputerze, należy wyedytować plik SynapsaRMAX.WSDL i w linii:
  <soap:address location='http://localhost/SynapsaRMAX/SynapsaRMAX.ASP'/>
  zamienić localhost na nazwę hosta (nazwę serwera z zainstalowaym SynapsaWebService).
- Instalację można przetestować za pomocą skryptu example_FindReverses.vbs (opcja SOAP).

5 Korzystanie z modułu dostępu SynapsaRMA

5.1 Metoda wysokopoziomowa, dostęp z poziomu środowisk programistycznych

W metodzie tej należy

- Utworzyć referencję do biblioteki SynapsaCtrlX np. o nazwie synapsaRMA.
- Utworzyć obiekt klasy SynapsaCtrlX.SynapsaRMA (obiekt, z którego wywoływane będą funkcje).

- Utworzyć obiekt klasy zawierającej parametry funkcji i wypełnić te parametry (np. utworzyć kolekcję SynapsaCtrlX.FindReversesParams z wypełnionym numerem rewersu)
- Wywołać wybraną z wcześniej utworzonego obiektu klasy SynapsaCtrlX.SynapsaRMA metodę z parametrem (np. FindReverses)
- Jeśli wyniki są kolekcją (np. Rewerses) można je iterować przez standardową składnię danego języka (np. foreach), wykonując na wynikach określoną operację (czytanie wartości, zmiana wartości)
- Zwrócone wyniki mogą służyć jako parametr dla innych funkcji.

5.1.1 Przykład – wykorzystanie modułu w środowisku Excel, środowisko VBA.

Działanie – wyszukanie nierozliczonych rewersów przyjętych po dwutysięcznym roku.

```
'Procedura wyszukująca rekordy z Synapsy i wypełniająca arkusz
'procedura wykorzystuje wysokopoziomową metodę dostępu (obiekt SynapsaRMA)

'stwórz główny obiekt dostępu do Synapsy
Dim SynapsaRMA As New SynapsaCtrlX.SynapsaRMA

'ustaw parametry
Dim frParams As New SynapsaCtrlX.FindReversesParams
frParams.dataprzyjeciaod = ”2000.01.01”
frParams.oddane = 0

'wyszukaj
Dim rews As SynapsaCtrlX.Rewersy
Set rews = SynapsaRMA.FindReverses(frParams)

'wypełnij arkusz programu Excel
Dim rew As SynapsaCtrlX.Rewers
Dim row: row = 1
Cells(row, 1) = "Nr rewersu"
Cells(row, 2) = "SN"
Cells(row, 3) = "Kod towaru":
For Each rew In rews
    row = row + 1
    Cells(row, 1) = rew.nr_rew
    Cells(row, 2) = rew.sn
    Cells(row, 3) = rew.kod_towaru
Next

'koniec
MsgBox "OK"
```

5.2 Metoda wysokopoziomowa, dostęp z poziomu języków skryptowych

Wykorzystanie modułu z poziomu skryptowego różni się od wykorzystania w środowisku programistycznym. W środowisku tym nie dodaje się referencji do głównego obiektu i nie używa mocnego typowania.

Główną klasą wywołań metod dostępu do Synapsy nie jest SynapsaRMA, zamiast niej wykorzystuje się obiekt klasy SynapsaRMAV, która zawiera metody przyjmujące parametry typu Variant.

5.2.1 Przykład – wyszukiwanie pierwszego nierozliczonego rewersu, w środowisku skryptowym VBS

(zawartość pliku o nazwie np. z roszerzeniem .vbs)

```
'stwórz główny obiekt dostępu do Synapsy
```

```
Dim SynapsaRMA
Set SynapsaRMA = CreateObject("SynapsaCtrlX.SynapsaRMAV")

'ustaw parametry
set frParams=CreateObject("SynapsaCtrlX.FindReversesParams")
frParams.oddane=0

'wyszukaj i pokaż wynik
set rews = SynapsaRMA.FindReverses(frParams,"")
If rews.Count<1 Then
    msgbox "Nie znaleziono rewersu o podanym numerze"
else
    dim rew: set rew=rews(1)
    txt="Znaleziono rewers nr: " & rew.nr_rew & ", status: " & rew.status_naprawy
    MsgBox txt
end if

'koniec
MsgBox "OK"
```

5.3 Metoda wysokopoziomowa, wykorzystanie WebService przez SOAP

Poprzez WebService można wywoływać metody zdalnie, bez potrzeby instalowania bibliotek SynapsaRMA na kliencie.

Wyjątkiem jest środowisko klienckie Microsoft Soap Toolkit 3.0, które wymaga obecności i zarejestrowania biblioteki SynapsaCtrlX.dll.

Sposób wykorzystania WebService jest inny niż w przypadku korzystania z biblioteki ActiveX, zależny od środowiska wywołującego metody. Same metody pozostają jednak te same.

5.3.1 Przykład – korzystanie z SynapsaWebService ze zdalnego klienta, z poziomu środowiska skryptowego vbs

Przykład wymaga zainstalowania Microsoft Soap Toolkit 3.0 na kliencie. W podkatalogu soapX powinien znaleźć się plik SynapsaRMAX.wsml (ten sam, który znajduje się w części serwerowej) lub nieco krótszy plik SynapsaRMAXClient.wsml (również załączony do pliku instalacyjnego). Poniższy przykład wymaga też obecności i rejestracji biblioteki SynapsaCtrlX.dll na kliencie. (zawartość pliku o nazwie np. z roszerzeniem .vbs).

```
'stwórz główny obiekt dostępu do Synapsy
'w adresie do wsdl wpisz nazwę serwera lub pozostaw localhost
Set SynapsaRMA = CreateObject("MSSOAP.SoapClient30")
Dim wsdl: wsdl="http://localhost/SynapsaRMAX/SynapsaRMAX.wsdl"
SynapsaRMA.MSSoapInit wsdl, , , "soapX\SynapsaRMAX.wsml"
SynapsaRMA.ConnectorProperty("Timeout") = 10000

'ustaw parametry
set frParams=CreateObject("SynapsaCtrlX.FindReversesParams")
frParams.oddane=0

'wyszukaj i pokaż wynik
set rews = SynapsaRMA.FindReverses(frParams,"")
If rews.Count<1 Then
    msgbox "Nie znaleziono rewersu o podanym numerze"
else
    dim rew: set rew=rews(1)
    txt="Znaleziono rewers nr: " & rew.nr_rew & ", status: " & rew.status_naprawy
```

```
    MsgBox txt
end if

'koniec
MsgBox "OK"
```

5.3.2 Przykład – korzystanie SynapsaWebService z poziomu .NET (w tym ASP.NET)

Aby wykorzystać WebService należy w środowisku Visual Studio dodać referencję do serwisu poprzez opcję

Add Service Reference
(Windows Communication Foundation - WCF)
lub w trybie kompatybilności z .net 2.0:
Add Service Reference/Advanced.../Add Web Service.
(XML Web Reference)

W obu przypadkach należy podać adres WSDL, np:
http://localhost/SynapsaRMAX.WSDL
Referencję należy nazwać np. **RMAServer.**
Do pliku należy dodać odpowiedni namespace (**using *TwojNamespace*.RMAServer;**).

Podstawowy obiekt klasy **SynapsaRMAX** tworzony jest następująco:

```
RMAServer.SynapsaRMASoapPortClient synapsaRMA =
        new SynapsaRMASoapPortClient();
```

W trybie kompatybilności:

```
SynapsaRMAX synapsaRMA = new SynapsaRMAX().
```

Z poziomu obiektu synapsaRMA wywoływane są wszystkie główne metody.

W środowisku .Net nie będą wykorzystywane klasy kolekcji (Klienci, Produkty, Rewersy itp). Zamiast nich posługuje się tablicami, które dostępne są w obiektach kolekcji pod zmienną Items. Wykorzystuje się klasy z podkreśleniem.

Np. aby znaleźć towar o danym kodzie należy użyć:

```
_Produkt[] produkty=synapsaRMA.FindProducts
        (new FindProductsRequest(kod_produktu , 0, "")).Result.Items;
```

W trybie kompatybilności:

```
_Produkt[] produkty=synapsaRMA.FindProducts
        (kod_produktu,0,"").Items;
```

Uwaga: parametry tablicowe w trybie kompatybilności przekazuje się jako referencję

5.3.3 Przykład 2 – wyszukiwanie rewersów z poziomu .NET z zapisem informacji o wystawionej fakturze korygującej

Przykład wykorzystuje tryb kompatybilności z .Net 2.0 (*XML Web Reference*).

```
using TwojNamespace.RMAServer;
//parametry wejściowe
int nrrewersu=XXXXXX; //nr rewersu, do któreg będzie wykonany zapis
string dok_przydz="FA_TEST"; //numer faktury korygującej, do zapisania do rewersu
DateTime data_dok_przydz=DateTime.Now; //data faktury korygującej
string kod_towaru_wyd="TOW_TEST"; //kod towaru do wydania, który będzie zapisany do
    rewersu

//inicjalizacja
string config="";
SynapsaRMAX synapsaRMA = new SynapsaRMAX();

//****************************************
```

```
//WYSZUKAJ REWERSY
//*****************************************
_FindReversesParams fndRewPar = new _FindReversesParams();
fndRewPar.nrrewersu = nrrewersu.ToString();    //tu wypełnić, jeśli szukany jest konkretny nr
      rewersu
_Rewers[] arr_rew = synapsaRMA.FindReverses(fndRewPar, config).Items;
if (arr_rew.Length != 1)
      RaiseError ("Nie znaleziono rewersu");

ConsoleWrite ("Sukces - Znaleziono rewersy, ilość wyników: " + arr_rew.Length);
_Rewers rew=arr_rew[0];
ConsoleWrite ("nr rewersu: " + rew.nr_rew);
ConsoleWrite ("SN: " + rew.sn);
ConsoleWrite ("kod towaru: " + rew.kod_towaru);
ConsoleWrite ("nazwa towaru: " + rew.nazwa_towaru);
ConsoleWrite ("data wydania: " + rew.data_wydania);
ConsoleWrite ("SN towaru wydawanego: " + rew.wydano_sn);
ConsoleWrite ("kod towaru wydawanego: " + rew.kod_towaru_wyd);
ConsoleWrite ("nazwa towaru wydawanego: " + rew.nazwa_towaru_wyd);

//*****************************************
//SPRAWDŹ DANE REWERSU PRZED ZAPISANIEM
//*****************************************
if (rew.dok_przydz!="")
      RaiseError ("Rewers ma już zapisane dane");

//*****************************************
//USTAW DANE DO ZAPISU W ZNALEZIONYM REWERSIE
//*****************************************
rew.dok_przydz=dok_przydz;
rew.data_dok_przydz= data_dok_przydz.ToString("yyyy.MM.dd");//dataprzyjeciaod w formacie
      RRRR.MM.DD
rew.kod_towaru_wyd=kod_towaru_wyd;
ConsoleWrite ("Ustawiono dane do zapisania");

//*****************************************
//ZAPISZ REWERS
//*****************************************
_Rewersy col_rew = new _Rewersy();
col_rew.Items=arr_rew;
synapsaRMA.SaveReversesAssign(ref col_rew, config);
ConsoleWrite ("Zapisano dane do rewersu");
```

5.4 Metoda niskopoziomowa – dostęp przez uproszczony SQL

Metoda niskopoziomowa umożliwia bardziej bezpośredni dostęp do środowiska bazodanowego programu Synapsa. Wykorzystuje się w niej klasy przypominające *recordsety* znane z ADO i DAO oraz składnię uproszczonego języka SQL. Operacje wykonywane są na „widokach” (tabele z powiązanymi polami pochodzącymi z innych tabel). Metoda niskopoziomowa umożliwia edycję rekordów.

Metoda ta wykorzystywana jest wewnętrznie przez programistów Union Systems i nie jest opisana w dokumentacji.

6 Uwagi do opisu klas i metod modułu SynapsaRMA

- W większości przypadków parametry przekazywane są przez wartość. Jeżeli w opisie nie podano słowa kluczowego ByRef, parametr jest przekazywany przez wartość (nie jest możliwa jego zwrotna modyfikacja).
- W przypadku niektórych funkcji parametrami są obiekty, które zostają **zaktualizowane** po wykonaniu metody (przekazywanie przez referencję). Dotyczy to np. zapisu zgłoszeń, w których po zapisaniu pojawia się numer zgłoszenia, pełna nazwa towaru (do zgłoszenia podaje się sam kod towaru lub PN czyli EAN) i inne dane.
- Jeżeli w opisie parametrów i składników klas nie podano inaczej, zmienna jest typu **tekstowego**.
- Jeśli klasa zwraca kolekcję (np. Rewersy), pod opisem metody podano opis klasy (np. Rewers) a nie kolekcji. Opis wykorzystywania kolekcji jest wspólny dla wszystkich klas. W przypadku używania kolekcji przez SOAP (np. w środowisku .Net), z kolekcji korzysta się pośrednio za pomocą tablic, do których odwołuje się przez zmienną **Items**.

6.1 Wspólne parametry metod – parametr config, uprawnienia do metod

Parametr config w większości metod może być ustawiony na pusty (a w VB może zostać pominięty), co oznacza operacje na wszystkich filiach z domyślnymi uprawnieniami.

Interpretacja uprawnień, gdy nie podano jawnie użytkownika i hasła w parametrze *config*, zależy od zdefiniowania i uprawnień **użytkownika domyślnego dla funkcji zewnętrznych**.
Użytkownikiem domyślnym dla funkcji zewnętrznych wybierany jest w zaawansowanych opcjach programu Synapsa, w panelu Opcje (pole „użytkownik dla operacji zewnętrznych").
Są trzy możliwości:

- Jeżeli użytkownik domyślny nie jest zdefiniowany i nie wpisano w config loginu i hasła, funkcje otrzymują pełny dostęp do metod.
- Jeżeli użytkownik domyślny jest zdefiniowany i nie wpisano w config loginu i hasła, zadziałają wyłącznie funkcje zgodne z uprawnieniami użytkownika domyślnego. **Jeżeli użytkownik domyślny został zdefiniowany, ale ma wyłączone wszystkie uprawnienia, funkcje zapisu zakończą się niepowodzeniem.**
- Podanie użytkownika i poprawnego hasła pozwala na dostęp do tych funkcji, do których podany w *config* użytkownik ma uprawnienia (użytkownik domyślny nie ma wtedy znaczenia).

Jeżeli zależy na dostępie bez jawnego podawania loginu i hasła do niskopoziomowych funkcji zapisu, wymagane jest uprawnienie pełnej edycji rewersów dla użytkownika domyślnego (lub brak użytkownika domyślnego w Synapsie).

Składnia parametru config jest następująca:
„parametr1=wartość1; [parametr2=wartość2]; [...]; [parametrN=wartośćN];"

Dopuszczalne parametry konfiguracji:

konfiguracja – dwuliterowa nazwa filii (opcjonalna)
filtr – filtr wyszukiwań dla filii
uzytkownik – nazwa użytkownika dla metod wymagających uwierzytelniania
haslo – hasło użytkownika dla metod wymagających uwierzytelniania

7 Opis klas dostępu wysokopoziomowego

Biblioteka SynapsaCtrlX zawiera struktury obiektów podstawowych Synapsy, kolekcje oraz jedną główną klasę SynapsaRMA, z której wywoływane są wszystkie wysokopoziomowe funkcje.

7.1 Opis metod kolekcji (dotyczy klas Klienci, Produkty, SNInfos, PozycjeZgloszen, Rewersy, SzczegolyNaprawy)

```
Item(vntIndex As Long) As Class
```
Działanie: Pobiera obiekt kolekcji o podanym indeksie.
```
Count() As Long
```
Działanie: Pobiera ilość obiektów w kolekcji.
```
Add(objNewMember As Class) As Long
```
Działanie: Dodaje obiekt do kolekcji.
```
Remove(vntIndex As Long)
```
Działanie: Usuwa obiekt do kolekcji.
```
Items()
```
Działanie: Oferuje dostęp do kolekcji w postaci tablicy.
W przypadku udostępnienia modułu przez zdalną referencję do webSerwisu (np. w środowisku .Net), klasy będą wykorzystywać zmienną Items do przeglądania, dodawania i usuwania zawartości kolekcji.

Class jest klasą typu Klient, Produkt, SNInfo, PozycjaZgloszenia, Rewers.

Obiekty kolekcji mogą być enumerowane za pomocą składni For Each - Next w VB, VBA, VBS.

7.2 Metoda Test – funkcja testowa

```
Test(a As Long, b As Long) As Long
```
Działanie: Metoda testowa, do sprawdzania działania serwisu IIS/SOAP (dodawanie a+b)

7.3 Metoda FindSN – szukaj numeru seryjnego

```
FindSN(fp As FindSNParams, Config as String) As SNInfos
      Klasa FindSNParams:
            kodklienta As String
            nipklienta As String
            sn As String
            faktura As String
            umowa As String
            kodtowaru As String
            pntowaru As String
            localonly As String
```

Działanie: szuka numeru seryjnego, ewentualnie pozycje faktury lub umowy, tworząc listę opisującą historię egzemplarza. Egzemplarze są szukane w:

- rewersach Synapsy (w historii wszystkich przyjęć serwisowych),
- dokumentach wydań programu Magnat
- w module produkcji Magnata
- w innych zewnętrznych źródłach danych

Wymagane jest podanie przynajmniej jednego parametru ograniczającego ilość danych:

- numer seryjny
- faktura
- umowa

Wynik:

kolekcja obiektów klasy **SNInfo (opis niżej)** - klasa zawiera dane o egzemplarzach (opis niżej), pobrane z Synapsy i Magnata.

Parametry:

kodklienta - kod zalogowanego klienta, wymagany, jeśli nie podano nipu
nipklienta - nip kontrahenta, wymagany, jeśli nie podano kodu
sn - szukany numer seryjny
faktura - szukana faktura sprzedaży
umowa – szukany numer umowy
kodtowaru - szukany kod towaru, nie wymagany
pntowaru - szukany kod kreskowy towaru, nie wymagany
localonly - ustawić na "" (pole dotyczy internetowych źródeł numerów seryjnych)
config - stały ciąg konfiguracji, opis na początku rozdziału

Uwagi:
Procedura służy do wyświetlenia listy z opisem danego egzemplarza przy tworzeniu zgłoszenia. Użytkownik klika na wybrany (najczęściej ostatni) rekord w liście, wtedy dane z tego rekordu zostają skopiowane do zgłoszenia. W zgłoszeniu podaje dodatkowe parametry (np. opis usterki) i w ten sposób tworzy jeden rekord zgłoszenia. Klient powinien kliknąć na ostatni rekord w wyświetlonej liście obrazującej historię SN, lub na dowolnie wybrany rekord gdy wyszukiwane są pozycje faktury lub umowy.
Innym trybem działania przy wyszukiwaniu SN może być wyświetlanie w liście tylko ostatniego rekordu lub automatyczne kopiowanie danych do zgłoszenia bez pokazywania listy, albo uzależnienie powyższego od liczby znalezionych rekordów.
Podany SN może dotyczyć różnych typów towarów. Dobrą praktyką jest umożliwienie w takiej sytuacji wybrania konkretnej pozycji z listy, niekoniecznie ostatniej.
Rekordy pobrane z Synapsy (czyli wydane z serwisu) mają niezerową wartość w polu oznaczającym numer rewersu.
Rekordy pobrane z bazy sprzedaży (czyli zakupione i wydane z magazynu) mają w tym miejscu wartość 0.

Do zgłoszenia powinny być kopiowane następujące pola z wybranego przez użytkownika rekordu (czyli z rekordu opartym na klasie SNInfo):
sn, kod_towaru, faktura_sprzedazy, data_sprzedazy, oraz numer umowy, jeśli jest wymagany dla serwisu
Nie są kopiowane, ale powinny być wyświetlane w liście historii SN pola:
nazwa_towaru, gwarancja, typ_gwarancji, data_gwarancji, nr_rew
Nie musi być wyświetlane pole
PN_towaru
Skojarzenie towarów, podobnie jak klientów, będzie na podstawie kodu, choć zostało zaimplementowane dodatkowo wyszukiwanie towaru wg PN a klienta wg pola NIP (podanego ze znaczącym ustawieniem kresek).
W liście nie należy wyświetlać czasu i daty gwarancji, jeśli typ gwarancji jest typu "" (nieokreślona), "B" (brak), "W" (wieczysta). W pozostałych przypadkach - "G" (gwarancja), "R" (rękojmia) należy wyświetlać datę i czas gwarancji. Czas podany jest w miesiącach.

7.4 Opis klasy SNInfo (składnik kolekcji SNInfos zwracanej przez FindSN)

sn - numer seryjny
kod_klienta - kod klienta
nip_klienta - NIP klienta
kod_towaru - kod towaru
nazwa_towaru - nazwa towaru
pn_towaru - PN (product number) - kod EAN towaru
gwarancja - czas gwarancji lub rękojmi podany w miesiącach.
typ_gwarancji - typ gwarancji: "" (nieokreślona), "B" (brak), "G" (gwarancja), "R" (rękojmia), "W" (wieczysta)
data_gwarancji - data w formacie RRRR.MM.DD oznaczająca moment upływu gwarancji lub rękojmii.
Czas ten może być odnowiony w serwisie, dlatego w rewersie Synapsy może być inny niż wynika to z faktury sprzedaży.
faktura_sprzedazy - faktura sprzedaży (czyli zakupu przez klienta - tak powinno się to pole wyświetlać klientowi).

umowa - numer umowy (np. kontraktu na długookresową obsługę gwarancyjną).
data_sprzedazy - data sprzedaży
nr_rew - numer rewersu
ilosc_sn - ilość numerów (dla numerów partii zamiast numerów seryjnych)

7.5 Metoda SavePack – zapisz paczkę zgłoszeń

```
SavePack(ByRef PozycjeZgloszen As PozycjeZgloszen, config As String)
```

Działanie: Zapisuje rekordy zgłoszeniowe, stanowiących jedno zgłoszenie.
Parametry:
pozycje - parametr wejściowo - wyjściowy - kolekcja obiektów PozycjaZgloszenia
config - stały ciąg konfiguracji, opis na początku rozdziału

Przy zapisie brane są pod uwagę tylko niektóre pola klasy PozycjaZgloszenia.
Po zapisie pozostałe pola klasy są automatycznie uzupełnione, np. uzupełnia się numer rewersu zgłoszeniowego, numer zgłoszenia zbiorczego dla całej "paczki", data wysłania itp.
Ten sam rekord po zapisaniu może posłużyć do jego edycji (dzięki pamiętaniu jego identyfikatora, czyli numeru rewersu zgłoszeniowego).
Przy zapisie paczki wymagane jest, żeby każdy z rekordów miał zapisany kod klienta.
Rekord można edytować, dopóki nie jest rozpatrzony. Edycja rozpatrzonego towaru da w wyniku błąd (zastosowano standardowe generowanie błędów przez "raise error").
Uwagi:
Została zaimplementowana synchronizacja klientów i towarów pomiędzy Synapsą a modułami zewnętrznymi w momencie zapisywania pozycji zgłoszenia.

7.6 Opis klasy PozycjaZgloszenia (składnik kolekcji PozycjeZgloszen używanej przez FindPack, SavePack, CancelPack)

poniższe pola wypełnia aplikacja związana z logowaniem
kod_klienta - kod klienta
nip_klienta - nip klienta
auto_rma - Long – Oznacza różne metody rozpatrywania zgłoszenia.
- Wartość „0” oznacza brak automatycznej akceptacji RMA (do rozpatrzenia przez serwisantów).
- Wartość „1” określa, że dany rekord zgłoszenia ma być automatycznie rozpatrzony i automatycznie nadany zostanie numer RMA. Przy ustawieniu tego pola, aplikacja wywołująca metodę SavePack może sprawdzić za pomocą metody FindSN, czy towar został odnaleziony i czy jest jeszcze na gwarancji. Programiści serwisu internetowego powinni sami utworzyć procedurę do ustawiania tego pola odpowiednio do zapotrzebowania serwisu.
- Wartość „2” oznacza, że moduł RMA sam wykona procedurę sprawdzania zgłoszenia. Procedura FindSN będzie wywołana wewnętrznie, dla określenia zgodności sn, faktury i daty zakupu, kodu towaru lub pn, oraz typu i czasu obowiązywania gwarancji.

poniższe pola wypełnia klient (większość przez kopiowanie z SNInfo, jesli towar ma SN)
nr_rew_zewn - oznaczenie klienta (zwykle numer własnego rewersu klienta/oddziału)
sn - numer seryjny
kod_towaru - kod towaru - pole jest wymagane. Zwykle pole będzie automatycznie wypełnione przez kopiowanie z listy SN, ale dla towarów, które nie mają SN będzie określane "ręcznie" przez użytkownika, tzn wybierana z listy. Lista powinna być wyświetlona jako okienko pop-up.
pn_towaru - Product Number (kod kreskowy EAN)
data_sprzedazy - data sprzedaży (dla klienta widoczne jako data zakupu) - w formacie RRRR.MM.DD
faktura_zakupu - faktura zakupu
umowa - numer umowy
wyposazenie - 79 znaków - wyposażenie do wysyłanego towaru (np. kabel zasilający itp.).
opis_uszkodzenia - opis uszkodzenia. Bez ograniczenia ilości znaków.

transport - sposób dostarczenia towaru. Typy: "W" (własny), "F" (firma przewozowa), "P" (poczta), "I" (inny). Domyślnie powinno być F.
odbior - deklarowany odbiór towaru. Typy jak wyżej.
nr_listu - nr listu przewozowego. Pole nie wymagane.
uwagi_klienta - dodatkowe uwagi klienta.
kod_magazynu - dział serwisowy. Wartość powinna być ustalona dzięki metodzie FindStorages.

pola wypełniane automatycznie przy zgloszeniu (zwracane i zapisywane przy edycji). Przy tworzeniu zgłoszenia należy zostawić pola puste (nie wypełniać)

nr_rew_zgl - Long - nadany unikalny numer pojedynczego rekordu zgłoszeniowego. Klient może się nim posługiwać przy zapytaniach o towar, może też używać numer rewersu (czyli numeru związanego z przyjęciem towaru) ewentualnie numeru zgłoszenia (czyli całej paczki) lub numeru RMA (czyli numeru zgody na wysłanie paczki). Jednak numer zgłoszenia i numer RMA nie identyfikuje jednoznacznie towaru, jeśli w paczce był więcej niż jeden towar.
data_wyslania
nr_zgloszenia
nazwa_towaru

wypełniane przy nadawaniu RMA przez serwis (zwykle w ciągu dwóch dni od wysłania)
Przy tworzeniu zgłoszenia należy zostawić pola puste (nie wypełniać).

nr_RMA_zbiorczego - numer RMA czyli numer zgody na wysłania paczki lub części paczki.
stanR - Long - 0 lub 1 - "rozpatrzony" oznacza, czy potwierdzenie RMA zostało dokonane (jeśli 1). Ustawiane też przy przyjęciu serwisowym bez tworzenia RMA.
stanC - Long - 0 lub 1 - "oczekujacy" - stan oczekiwania na paczkę po nadaniu numeru RMA
stanX - Long - 0 lub 1 - "odrzucony" - ujemny wynik przy rozpatrywaniu RMA, klient nie powinien wysyłać tego towardu do serwisu. Uwagi serwisu powinny zawierać powód odrzucenia towaru.
stanUS - Long - 0 lub 1 - "uwagi serwisu" - oznacza, że serwis zapisał uwagi w polu "uwagi serwisu"
uwagi_serwisu - uwagi serwisu - zwykle przy odrzuceniach, ale nie tylko. Również serwis może zapisać uwagę, zaznaczając stanUS i oczekiwać na zmianę zgłoszenia.
data_rozpatrzenia - data rozpatrzenia lub odrzucenia rewersu zgłoszeniowego.

pola ze związanego rewersu wypełniane przy operacjach serwisowych. Te pola są wypełniane po przyjęciu do serwisu w trakcie operacji serwisowych. Są one zapisywane w rewersie, ale dla wygody widoczne są także dla zgłoszenia.

nr_rew - Long - numer rewersu - czyli numer przyjęcia towaru
stanZ - Long - 0 lub 1 - do Zwrotu - oznacza, że towar jest gotowy do zwrotu (naprawiony lub przydzielony)
stanO - Long - oznacza, że towar został wydany klientowi - wysłany paczką dla odbioru typu „F”. Klient może oczekiwać na towar.

7.7 Metoda CancelPack – oznacz paczkę zgłoszeń jako nieaktualną

```
CancelPack(ByRef PozycjeZgloszen As PozycjeZgloszen, config As String)
```

Działanie: Anuluje nierozpatrzone rekordy zgłoszeniowe, wczytane wcześniej metodą FindPack. Zgłoszenia zostaną oznaczone jako rozpatrzone negatywnie z odpowiednią adnotacją.
Parametry:
pozycje - parametr wejściowo - wyjściowy - kolekcja obiektów PozycjaZgloszenia
config - stały ciąg konfiguracji, opis na początku rozdziału

7.8 Metoda FindPack – znajdź paczki zgłoszeń

```
FindPack(fp As FindPackParams, config As String) As PozycjeZgloszen
      Klasa FindPackParams:
            kodklienta As String
            nipklienta As String
            nrrewzgl As String
            nrzgloszenia As String
            nrrma As String
            nrrew As String
            stanR As String
            stanC As String
            datawyslaniaod As String
            datawyslaniado As String
            fakturazakupu As String
            umowa As String
            nrlistuzgloszenia As String
```

Działanie: Wyszukuje zapisane zgłoszenia (paczki). Funkcja powinna być wykorzystywana do sprawdzania przez klienta stanu wybranej paczki - czy nadano numer RMA, czy dotarła do serwisu, czy jest do zwrotu, czy została wydana (te stany dotyczą oczywiście poszczególnych składowych paczki, czyli rewersów zgłoszeniowych).
Wynik:
kolekcja obiektów klasy PozycjaZgloszenia
Parametry:
kodklienta - kod klienta - wymagane
nrrewzgl, nrzgloszenia, nrrma, nrrew - pola typu String oznaczające numer pojedynczego rewersu zgłoszeniowego, numer zgłoszenia (całej paczki), numer RMA (dot. całej paczki lub części paczki), numer rewersu.
stanR - stan rozpatrzenia - "" - dowolny, "1" - tylko rozpatrzone, "0" - tylko nierozpatrzone
Muszą być podane takie filtry, które powodują ograniczoną listę wyników.
stanC - oczekujący - "" - wszystkie, "1" - rozpatrzone i oczekujące na przyjęcie
datawyslaniaod, datawyslaniado - filtracja wg dat zapisu zgłoszenia
fakturazakupu - faktura zakupu towaru przez klienta
umowa - numer umowy
config - stały ciąg konfiguracji, opis na początku rozdziału

7.9 Metoda FindProducts – przeszukaj kartotekę towarów:

```
FindProducts(kod As String, opcje As Integer, config As String) As Produkty
```

Działanie: Wyszukuje towary o podanym kodzie lub fragmencie kodu. Funkcja będzie potrzebna do wyboru towaru, gdy klient nie poda numeru seryjnego. Podanie fragmentu kodu jest konieczne do przedstawienia listy - w przeciwnym razie lista byłaby zbyt długa. Najlepszą propozycją jest umożliwienie wyboru towaru przez podanie początkowych znaków.
Wynik:
kolekcja obiektów klasy Produkt
Parametry:
kod - kod towaru lub fragment kodu
opcje - opcje wyszukiwania. Możliwe są następujące opcje:

- 0 - wyszukiwanie całego kodu,
- 1 - wyszukiwanie początku kodu,
- 2 - wyszukiwania w dowolnym miejscu kodu,
- 3 - wyszukiwanie wg dowolnego fragmentu nazwy,
- 4 - wyszukiwanie wg kategorii,
- 5 - wyszukiwanie wg początku kategorii
- 6 - wyszukiwanie wg PN.

7.10 Opis klasy Produkt (składnik kolekcji Produkty, wykorzystywanej przez metodę FindProducts).

kod - kod towaru
kod2 - alternatywny kod towaru
nazwa - nazwa towaru
pn - PN towaru
kategoria - kategoria towaru

7.11 Metoda FindClients – przeszukaj kartotekę klientów:

```
FindClients(kod As String, opcje As Integer, config As String) As Klienci
```

Działanie: Wyszukuje klientów o podanym kodzie lub fragmencie kodu.
Wynik:
kolekcja obiektów klasy Klient
Parametry:
kod - kod klienta lub fragment kodu
opcje - opcje wyszukiwania. Możliwe są następujące opcje:

0 - wyszukiwanie całego kodu,
1 - wyszukiwanie początku kodu,
2 - wyszukiwania w dowolnym miejscu kodu,
3 - wyszukiwanie wg dowolnego fragmentu nazwy,
4 - wyszukiwanie wg NIPu,
5 - wyszukiwanie wg adresu e-mail
6 - wyszukiwanie wg www_login
7 - wyszukiwanie całego kodu2,
8 - wyszukiwanie początku kodu2,
9 - wyszukiwania w dowolnym miejscu kodu2,

config - stały ciąg konfiguracji, opis na początku rozdziału

7.12 Opis klasy Klient (składnik kolekcji Klienci używanej przez FindClients, SaveClients)

kod - kod towaru
kod2 - alternatywny kod klienta
nazwa - nazwa klienta
nip - NIP klienta
email - e-mail klienta
kategoria - kategoria
adres1 - adres
adres2 - doprecyzowanie adresu
tel - telefon
fax - fax
kontakt - osoba kontaktowa
opis - opis
login - login (używany w przypadku zapamiętywania loginów i haseł www w Synapsie)
haslo - haslo (używane w przypadku zapamiętywania loginów i haseł www w Synapsie)

7.13 Metoda SaveClients – zapisz dane klienta:

```
SaveClients(ByRef Klienci As Klienci, config As String)
```

Działanie: zapisuje dane o klientach wczytanych wcześniej metodą FindClient. Metoda służy m. in. do zmiany pola *email*, które jest wykorzystywane przez moduł automatycznych wysyłek potwierdzeń o zgłoszeniach, naprawach i przyjęciach warunkowych.
Za pomocą metody SaveClients możliwe jest też dodanie nowego klienta do kartoteki, jeśli ustawiono pole „nowy” na wartość 1.

Parametry:
Klienci - parametr wejściowo - wyjściowy - kolekcja obiektów Klienci
config - stały ciąg konfiguracji, opis na początku rozdziału

7.14 Metoda FindStorages – znajdź magazyny

```
FindStorages(typ As Integer, ByVal config As String) As Magazyny
```

Działanie: Znajduje magazyny (działy serwisowe) wg podanych kryteriów.
Wynik:
kolekcja obiektów klasy Magazyn
Parametry:
typ - filtracja wg typu magazynu
1 – wyszukanie wszystkich przyjęć
2 – wyszukanie magazynów przyjęć do filii
3 – wyszukanie wszystkich magazynów Synapsy
config - stały ciąg konfiguracji, opis na początku rozdziału

7.15 Opis klasy Magazyn (składnik kolekcji Magazyny używanej przez metodę FindStorages)

kod - kod towaru
kod2 - alternatywny kod klienta
nazwa - nazwa klienta
nip - NIP klienta
email - e-mail klienta
kategoria - kategoria
adres1 - adres
adres2 - doprecyzowanie adresu
tel - telefon
fax - fax
kontakt - osoba kontaktowa
opis - opis
login - login (używany w przypadku zapamiętywania loginów i haseł www w Synapsie)
haslo - haslo (używane w przypadku zapamiętywania loginów i haseł www w Synapsie)

7.16 Metoda FindReverses – znajdź rewersy serwisowe

```
FindReverses(fp As FindReversesParams, config As String) As Rewersy
      Klasa FindReversesParams:
            kodklienta As String
            nipklienta As String
            nrrewersu As String
            nrrewzgl As String
            rewzewn As String
            oddane As String
            dowymianywewn As String
            dataprzyjeciaod As String
            dataprzyjeciado As String
            datawydaniaod As String
            datawydaniado As String
            nrlistuprzyjecia As String
            nrlistuwydania As String
            fakturasprzedazy As String
            umowa As String
            poziomszczeg As String
```

Działanie: znajduje rewersy, gromadzące informacje o operacjach serwisowych od momentu przyjęcia towaru do serwisu do wydania towaru lub innego rodzaju rozliczenia z klientem.
Wynik:

kolekcja obiektów klasy Rewers
Parametry:
kodklienta - filtracja wg kodu klienta
nipklienta - filtracja wg nipu klienta
nrrewersu - filtracja wg nr rewersu
nrrewzgl - filtracja wg nr oznaczającego pozycję zgłoszenia - należącego do zgłoszenia lub do rozpatrzenia rma, które mogą zawierać 1 lub więcej pozycji zgłoszeniowych. Rewers jest powiązany z pozycją zgłoszenia, jeśli było wykonywane zgłoszenie przed przyjęciem.
rewzewn - filtracja wg własnego dowolnego oznaczenia pozycji (klient może oznaczyć pozycję zgłoszenia swoim oznaczeniem)
oddane - ciąg pusty oznacza brak filtracji, 0 oznacza rewersy nierozliczone (nieoddane), 1 oznacza rewersy rozliczone z klientem (ustawiony stan 'O - oddane')
dowymianywewn - wyszukanie rewersów przeznaczonych do wymiany wewnętrznej, czyli wydania nowego towaru z magazynu głównego lub wydania faktury korygującej (wyszukuje rekorddy ze statusem 'WW – wysyłka wewnętrzna' i niewypełnionym numerem dokumentu przydzielenia, uwzględnia również podrewersy)
dataprzyjeciaod - data przyjęcia od (data przyjęcia towaru do serwisu - czyli data otwarcia zlecenia serwisowego)
dataprzyjeciado - data przyjęcia do
datawydaniaod - data wydania od (w przypadku rewersów oddanych czasem wykorzystuje się filtrację wg daty wydania zamiast przyjęcia)
datawydaniado - data wydania do
nrlistuprzyjecia - numer listu przewozowego z operacji przyjęcia towaru
nrlistuwydania - numer listu przewozowego z operacji wydania towaru
fakturasprzedazy - faktura sprzedaży towaru klientowi
umowa - numer umowy
poziomszczeg - poziom szczegółowości w zwracanych rewersach. Możliwe są następujące wartości: 0 – poziom domyślny. 3 – pobiera listę szczegółów naprawy w polu „lista_szczegolow_naprawy”. Pozostałe wartości nie powinny być używane.

config - stały ciąg konfiguracji, opis na początku rozdziału

Uwagi:
Metoda służy do pokazania rewersów, czyli rekordów zawierających wszystkie dane o przebiegu procedury serwisowej od momentu przyjęcia towaru do serwisu. Rewers tworzony jest na podstawie zgłoszenia i jest z nim powiązane (praktycznie relacją 1 do 1).
Użytkownik serwisu internetowego (klient) powinien mieć możliwość łatwego przeglądania rewersów z podanego przedziału dat i wg stanu oddany/nieoddany. Powinna być też możliwość, aby z przeglądania listy zgłoszeń dało się łatwo przejść do przeglądania rewersów związanych z wybranym zgłoszeniem.
Klient powinien oglądać listę uproszczoną, a jeśli wejdzie do pełnej edycji rewersu mogą się w pojawić wszystkie powyższe dane dla wybranego rekordu.

7.17 Opis klasy Rewers (składnik kolekcji używanej przez SaveReverses, FindReverses)

kod_klienta - kod klienta
nip_klienta - NIP klienta
nr_rew - nr rewersu
nr_rew_zewn - oznaczenie nadane przez klienta, np obcy numer rewersu
do_zwrotu - czy już jest towar do zwrotu (choć zwracana może być też np faktura korygująca)
typ_zobowiazania - 'G (gwarancyjne), P (płatne), R (ryczałt), W (warunkowe)
oddane - 0 - nieoddane, 1 - oddane (rozliczone z klientem)
status_naprawy - stan naprawy - podpowiada klientowi, jak przebiega (lub przebiegała) naprawa
możliwe wartości podano niżej. W zależności od wartości klient powinien otrzymać czytelniejszy opis podany obok (przy pełnej edycji rewersu):
UDOSTAWCY - *Wysłany do serwisu zewnętrznego*
OCZEKUJĄCY - *Oczekuje towar lub faktura korygująca*
WYMIENIANY - *Oczekuje na wymianę lub fakturę korygującą*
ROZPATRZONY - *Przygotowany do wymiany*

PRZYDZIELONY - *Wymieniono lub wystawiono fakturę korygującą*
WYMIENIONY - *Wymieniono*
NAPRAWIONY - *Naprawiono*
SPRAWNY - *Towar był sprawny*
ODMOWA - *Odmowa naprawy*
Uwagi:
Stan UDOSTAWCY oznacza stan W (wysłany do dostawcy) programu Synapsa
Stan WYMIENIANY oznacza stan WW (wysyłka wewnętrzna) programu Synapsa
Stan OCZEKUJĄCY oznacza, że klient może odebrać towar lub fakturę korygującą na miejscu, po przekroczonym czasie krytycznym naprawy.
Stan ROZPATRZONY oznacza, że wpisano dane o wydaniu, choć nie ustawiono jeszcze docelowego sposobu rozliczenia
Stan PRZYDZIELONY oznacza ustawienie sposobu rozliczenia na I, N, MM, RR.
Stan NAPRAWIONY oznacza ustawienie sposobu rozliczenia Z (zwykła naprawa lub naprawa przez wysyłkę i odbiór od dostawcy)
Stan WYMIENIONY oznacza to samo co stan PRZYDZIELONY ale po wydaniu towaru klientowi
Stan SPRAWNY oznacza, że testy wykazały brak usterki pomimo zgłoszenia jej przez klienta
Stan ODMOWA oznacza odmowę naprawy

rewers_zbiorczy - numer rewersu nadrzędnego. Podczas wyszukiwania rewersów na potrzeby RMA domyślnie nie są pobierane rewersy podrzędne. Jedynie przy filtracji na potrzeby wymian wewnętrznych rewersy nadrzędne są uwzględniane.
data_przyjecia - data przyjęcia towaru
nrrewzgl - numer pozycji zgłoszenia, jeśli przed przyjęciem było wykonywane zgłoszenie - np telefoniczne lub internetowe
sn - numer seryjny
kod_towaru - kod towaru przyniesionego do serwisu
nazwa_towaru - nazwa towaru
pn_towaru - product number (kod kreskowy EAN)
data_sprzedazy - data sprzedaży
faktura_sprzedazy - faktura sprzedaży
umowa - numer umowy
wyposazenie - wyposażenie
opis_uszkodzenia - opis uszkodzenia (o ograniczonej długości, w zgłoszeniu długość opisu usterki może być dowolna)
transport - typ dostarczenia, W- własny, P - poczta, F- firma przewozowa, I-inny
odbior - deklarowany typ odbioru, W- własny, P - poczta, F- firma przewozowa, I-inny
nr_listu_przyjecia nr listu przewozowego przyjęcia
uwagi_dla_klienta - uwagi serwisowe dla klienta, wpisywane zwykle w momencie przyjęcia towaru i drukowane na rewersie
data_naprawy - data naprawy lub data wymiany (nie musi być identyczna jak data wydania)
opis_naprawy - opis naprawy
nr_listu_wydania - nr listu wydania
data_wydania - data wydania
data_wys_wewn - data wysyłki wewnętrznej (data zlecenia do wystawienia dokumentu FK lub MM)
nazwa_towaru_wyd - nazwa towary wydawanego (po zamianie) lub pusta, jeśli nie było zamiany
kod_towaru_wyd - kod towaru wydawanego (po zamianie) lub pusty, jeśli nie było zamiany
pn_towaru_wyd - PN towaru wydawanego (po zamianie) lub pusty, jeśli nie było zamiany
sn_towaru_wyd - SN towaru wydawanego (po zamianie) lub pusty, jeśli nie było zamiany
dok_przydz - numer dokumentu przydzielenia towaru lub wydania (zwykle numer dokumentu MM lub faktury korygującej)
data_dok_przydz - data dokumentu przydzielenia/wydania
gwarancja - gwarancja w miesiącach
typ_gwarancji - typ gwarancji (G gwarancja, R rękojmia, F-Fabryczna, W -wieczysta, B-brak, "" niezdefiniowana)
data_gwarancji - data końcowa gwarancji (dla typów G lub R)
tp - ilość dni od przyjęcia do wydania (lub do chwili obecnej)
tn - ilość dni do naprawy (lub do chwili obecnej)

lista_szczegolow_naprawy – kolekcja typu SzczegolyNaprawy. Lista szczegółów naprawy zawierająca rekordy klasy SzczegolNaprawy, zawierającej pola: id, opis, koszt, czas pracy. Kolekcja jest edytowalna. Za jej pomocą można dodawać i edytować listę szczegółów naprawy.

7.18 Opis klasy SzczegolNaprawy (składnik kolekcji SzczegolyNaprawy, używanej w klasie Rewers, w metodach FindReverses, SaveReverses)

Id – identyfikator rekordu
opis – opis wykonanej czynności
koszt – koszt czynności
czas_pracy – czas wykonania czynności

7.19 Metoda SaveReverses – zapisuje do rewersu dane i wywołuje określoną operację serwisową

```
SaveReverses(ByRef Rewersy As Rewersy, config As String)
```

Działanie: Zapisuje do rewersu (wczytanego metodą FindReverses lub utworzonego ręcznie) dane o wydaniu lub innej operacji i wykonuje tę operację w imieniu określonego użytkownika.

Kod użytkownika i hasło podawane są w parametrze config, np
config = ”uzytkownik=admin; haslo=123”

Operacja będzie wykonana, jeżeli podano ją w parametrze status_naprawy. Tekst pola musi być sformatowany w następujący sposób:
„EXEC:Operacja1[_Operacja2][...][_OperacjaN]”
Jednocześnie może być wykonanych kilka operacji, np. przyjęcie, naprawa, wydanie, np:
„EXEC:PRZYDZIEL_WYDAJ”.

Lista możliwych operacji jest następująca:

- PRZYJMIJ – przyjęcie nowego rewersu
- WYSLIJWEW – wysyłka wewnętrzna (ustawia stan WW)
- PRZYDZIEL – odbiór z wysyłki wewnętrznej z przydzieleniem nowego towaru lub faktury korygującej, z oznaczeniem rewersu jako gotowy do wydania (zdejmuje WW, oznacza sposób rozliczenia na MM, ustawia status gotowości)
- PRZYPISZ – przypisanie nowego towaru lub faktury korygującej, bez odbioru z wysyłki wewnętrznej i bez oznaczenia rewersu jako gotowy do wydania
- NAPRAW – oznaczenie towaru klienta jako naprawiony (stan Z)
- WYDAJ – operacja wydania towaru

W zależności od operacji uwzględniane są różne pola rewersu. Przykładowo, dla operacji przydzielenia znaczące pola to:

- kod_klienta
- kod_towaru_wyd
- sn_towaru_wyd
- dok_przydz
- data_dok_przydz

7.20 Metoda SaveReversesAssign – zapisuje do rewersu dane o wydaniu (metoda dla kompatybilności wstecz, zastąpiona przez SaveReverses)

```
SaveReversesAssign(ByRef Rewersy As Rewersy, config As String)
```

Działanie: Zapisuje do rewersu wczytanego metodą FindReverses dane o wydaniu przypisane w module magazynowym.

Zapisywane są następujące dane rewersu:

- kod_klienta
- kod_towaru_wyd
- sn_towaru_wyd
- dok_przydz
- data_dok_przydz

Jeżeli w polu status_naprawy zostanie ustawiona wartość „EXEC:PRZYDZIEL_WYDAJ”, zostanie wykonana procedura zapisu razem z operacją wydania (ustawienie stanu „O - oddany” w rewersie), z ewentualnym usunięciem statusu wysyłki wewnętrznej. Sposób rozliczenia zostanie ustalony na „N” (jeśli nie było wysyłki wewnętrznej) lub „MM” (jeśli była wysyłka wewnętrzna).

7.21 Metoda SetProxy – podmienia główny obiekt dla wywołań metod

```
SetProxy(proxy as Variant) As Long
```

Działanie: Podmienia główny obiekt dla wywołań metod. Ma zastosowanie do niejawnych wywołań SOAP, jeśli obiektem proxy jest zainicjalizowany SoapClient.
Metoda dostępna z poziomu bezpośredniego wywołania klasy SynapsaCtrlX.SynapsaRMA na kliencie.
Wykorzystywane np. w środowisku Java z wykorzystaniem MS Soap 3.0, w którym metody webService wywoływane są pośrednio przez lokalny obiekt COM z ustawioną referencją do zainicjalizowanego obiektu klasy mssoap30.SoapClient30.