BSX PRINTER

PODRĘCZNIK PROGRAMISTY

Autor: **BinSoft**

29 grudnia 2023

SPIS TREŚCI

WSTĘP

Aplikacja bsxPrinter to serwer wydruków (na klasycznych drukarkach, jak również drukarkach etykiet i drukarkach fiskalnych), skanowania (wspiera protokół TWAIN), pośrednik (brama) dla drukarek fiskalnych, usługi KSeF (Krajowy System e-Faktur), pośrednik urządzeń USB, serwer obsługujący niektóre terminale płatnicze, jak również serwer pozwalający na dowolne rozszerzanie swoich funkcjonalności poprzez tzw. wtyczki. Posiada również funkcję emulatora drukarek fiskalnych, dzięki czemu ułatwia przygotowywanie wdrożeń z tymi urządzeniami.

Aplikacja jest stale rozwijana, dzięki czemu w kolejnych jej wersjach mogą pojawiać się różne dodatkowe funkcjonalności.

W chwili obecnej program obsługuje następujące protokoły komunikacyjne dla wydruków fiskalnych*:

- Posnet,
- Posnet Online,
- Thermal,
- Novitus,
- Novitus XML,
- Innova,
- FAREX,
- ELZAB;

Należy wspomnieć, że powyżej jest mowa o **protokołach**, a nie konkretnych modelach drukarek fiskalnych. Oznacza to, że z aplikacją może współpracować większość dostępnych na rynku urządzeń fiskalnych, gdyż większość z nich obsługuje któryś z

podanych wyżej protokołów. Należy jednak zwrócić uwagę, że czasami wymaga to wprowadzenia zmian w konfiguracji danej drukarki. W tym celu należy zasięgnąć informacji w podręczniku użytkownika danej drukarki lub u jej producenta.
Nie wszystkie urządzenia fiskalne oferują każdą z funkcji przewidzianych przez dany protokół. Zatem nie wszystkie modele drukarek będą pozwalały na drukowanie np. faktur, czy specjalnych raportów. Z tego też względu należy pamiętać, że aplikacja bsxPrinter oferuje tylko te funkcje, które dana drukarka udostępnia, oraz które zostały dla niej zaimplementowane. Oznacza to, że nie na każdej drukarce będzie można wykonać dowolną czynność z nią związaną. Użytkownik musi mieć tego świadomość wybierając oprogramowanie bsxPrinter.
Firma BinSoft gwarantuje, że na każdej ze wspieranych modeli drukarek fiskalnych będzie można wystawić paragon z podstawowymi produktami, z formą płatności „Gotówka", będzie można wydrukować raport dobowy oraz raport miesięczny.
Nie gwarantuje natomiast dostępności innych funkcji jak np. nadawanie rabatów/ upustów, obsługa opakowań, dodatkowe pola na dokumentach, dodatkowe raporty, możliwość programowania drukarki itp. Nasza aplikacja jest jednak stale rozwijana i kolejne funkcje będą się pojawiać w stale ukazujących się aktualizacjach.

Oprócz urządzeń fiskalnych obsługiwane są protokoły terminali płatniczych. W chwili obecnej są to protokoły:

- eService

bsxPrinter obsługuje również wydruki na klasycznych drukarkach. Umożliwia na nich wydruk dowolnych dokumentów PDF oraz HTML. Dzięki temu można drukować dowolne dokumenty np. faktury na papierze A4, etykiety, koperty, listy przewozowe. Obsługa

wydruków odbywa się poprzez wywołania systemowe, dlatego aby możliwe było poprawne korzystanie z danej drukarki, musi być ona poprawnie zainstalowana w systemie.

)* Dostępność określonych protokół może być ograniczona w dystrybucjach macOS i Linux.

BSXPRINTER DLA PROGRAMISTÓW

Program bsxPrinter posiada podstawowy interfejs pozwalający na drukowanie dokumentów fiskalnych wprost z jego okna. Tryb ten przeznaczony jest dla klientów indywidualnych kupujących program bsxPrinter w wersji podstawowej, bezpośrednio od producenta lub partnerów. Aplikacja posiada jednak szereg mechanizmów, które pozwalają innym programom na łączenie się do bsxPrinter i za jego pomocą drukowanie paragonów, faktur, czy też innych wydruków. Oznacza to, że zewnętrzni programiści mogą wyposażyć swoje aplikacje w odpowiednie mechanizmy współpracy z bsxPrinter i w ten sposób wzbogacić swoje aplikacje o obsługę urządzeń fiskalnych, automatyczne i zdalne drukowanie etykiet, kopert, listów przewozowych, czy też zdalne skanowanie dokumentów.

Od wersji 20.1.1 bsxPrinter oferuje nową funkcjonalność w postaci emulatora drukarek fiskalnych. W chwili obecnej obsługiwany jest tylko protokół Posnet i w pewnym ograniczonym zakresie. Będzie on jednak nadal rozwijany oraz będzie dodawana obsługa kolejnych protokołów.

DLACZEGO WARTO UŻYĆ BSXPRINTER W SWOIM PROJEKCIE?

Potrzebujesz wydruków fiskalnych (paragonów, faktur na drukarkach fiskalnych)?

Istnieje wiele producentów drukarek fiskalnych, m.in. Posnet, Novitus, Elzab, Farex itd. Każdy z nich udostępnia wiele modeli swoich urządzeń fiskalnych. Różne z tych modeli obsługują różne protokoły komunikacyjne. W ramach konkretnego modelu drukarki mogą występować dodatkowe różnice w ramach określonego protokołu. Zaimplementowanie obsługi drukarki fiskalnej w swojej aplikacji wiąże się zatem z koniecznością wykonania szeregu czynności:

- zapoznaniem się z dokumentacją techniczną poszczególnych protokołów komunikacyjnych;
- implementacją obsługi tych protokołów w swoich aplikacjach;
- implementacją obsługi komunikacji za pomocą portu szeregowego RS232;
- implementacja obsługi różnego rodzaju przejściówek USB<->RS232;
- implementacja obsługi Bluetooth,
- implementacja obsługi komunikacji TCP/IP,
- zdobyciem wielu różnych modeli drukarek fiskalnych celem przetestowania i dostosowania swojej aplikacji do każdej z nich;
- bieżące wsparcie dla danych drukarek oraz dodawanie obsługi do kolejnych modeli i funkcjonalności, które mogą się pojawić w przyszłości; w ostatnim czasie zmiany jakie nastąpiły tego typu to np. wprowadzenie obowiązkowego wydruku numerów NIP nabywcy, czy też obsługę e-Dokumentów;

Realizacja tych czynności wiąże się z dodatkowymi kosztami dla danej firmy. Wymaga też dodatkowych umiejętności programistycznych, posiadania dodatkowych bibliotek itp.

Oprócz powyższego, implementując obsługę drukarek w swoich aplikacjach deweloper musi uwzględnić sytuację, jak obsłużyć drukarki z poziomu wielu stanowisk - jeśli jego aplikacja pozwala na pracę sieciową. W swoim produkcie musi zatem zaimplementować odpowiednie mechanizmy kolejkowania.

Jeśli programista próbuje stworzyć swoją aplikację na urządzenia mobilne lub pod systemy macOS, czy Linux - pojawiają się kolejne bariery do pokonania. W jaki sposób z poziomu aplikacji mobilnej (np. tabletu) komunikować się poprzez RS232, skoro urządzenie może nie oferować w ogóle takiego interfejsu?

Wreszcie, jeśli developer tworzy aplikację webową - także nie ma możliwości zaoferowania swoim klientom obsługi urządzeń fiskalnych. Bowiem z poziomu

przeglądarki internetowej nie ma możliwości nawiązania połączenia poprzez port szeregowy (bez użycia dodatkowo instalowanych rozszerzeń w przeglądarce).
Wszystkie te problemy znikają - kiedy wykorzystamy bsxPrinter.

Potrzebujesz wydruków klasycznych dokumentów?

Drukowanie klasycznych dokumentów na drukarkach A4 - nie jest już tak problematyczne jak drukowanie dokumentów fiskalnych. Ale nadal mogą pojawić się trudności - jak zaimplementować takie funkcje z poziomu aplikacji mobilnej czy webowej? W jaki sposób zautomatyzować drukowanie np. etykiety do listu przewozowego z poziomu strony WWW - by odbywało się jednym kliknięciem, na konkretnej drukarce. Jak z poziomu strony WWW zdefiniować odpowiednio format papieru by móc wydrukować np. etykietę czy kopertę?

Posiadasz drukarkę fiskalną podłączoną kablem USB a chciałbyś się z nią połączyć z wielu stanowisk poprzez sieć LAN?

bsxPrinter może pełnić rolę „pośrednika" (bramki). Możesz w niej skonfigurować drukarkę fiskalną poprzez USB czy port szeregowy oraz uruchomić w nim tzw. serwer TCP/IP. Następnie w aplikacjach, w których chcesz dodać obsługę takiej drukarki (np. programy sprzedażowe, księgowe itp.) wskazujesz, że posiadasz drukarkę podłączoną w sieci lokalnej i podajesz dane dostępowe do bsxPrinter. Kiedy taka aplikacja spróbuje wysyłać komendy do bsxPrinter ten „rozpozna", że są to komendy danego protokołu (Posnet, Thermal, Novitus XML) i automatycznie przekieruje je bezpośrednio do drukarki.

Chcesz się w prosty sposób zintegrować z usługą KSeF (Krajowy System e-Faktur)?

bsxPrinter obsługuje system KSeF. Dzięki temu w prosty sposób możesz wysyłać faktury do KSeF, jak również odbierać faktury wystawione przez innych kontrahentów Tobie.

Twoja wiedza ogranicza się jedynie do obsługi samego formatu XML KSeF. Nie musisz wdrażać się w zawiłości API KSeF, obsługę kluczy kryptograficznych RSA itp.

NA CO POZWALA BSXPRINTER?

Aplikacja bsxPrinter pozwala na nawiązanie z nią połączenia w jeden następujących sposobów:

- poprzez komunikację TCP/IP,
- poprzez komunikację WebSocket (WS + WSS),
- poprzez lokalny serwer HTTP/HTTPS,
- poprzez współdzielony folder lokalny,
- poprzez folder na serwerze FTP/SFTP,
- poprzez odpytywanie określonego adresu URL i pliki XML,
- poprzez bramę BinSoft Gate;

W dowolnej aplikacji desktopowej czy mobilnej - programista może w prosty sposób nawiązywać połączenia TCP/IP, użyć WebSocket lub wywoływać określony adres URL by łączyć się z serwerem HTTP. Nie ma znaczenia z jakiego języka programowania korzysta, czy też pod jaki system operacyjny tworzy swoją aplikację. Może to być Windows, macOS, Linux, iOS, Android... Ważne jest jedynie, aby na jednym stanowisku klienta był uruchomiony komputer z systemem Windows lub Mac, a na nim bsxPrinter. Aplikacje z dowolnych urządzeń w sieci mogą się z nim połączyć i korzystać z jego usług.

Programista nie musi posiadać umiejętności programowania portu szeregowego, obsługi sterownika TWAIN itp. Nie musi borykać się z problemami emulacji portu w nowszych komputerach, gdzie porty emuluje się odpowiednimi przejściówkami USB. Nie musi znać protokołów komunikacyjnych poszczególnych drukarek, znać szczegóły

implementacyjne każdej z nich itp. Wystarczy, że pozna zestaw komend oferowanych przez bsxPrinter i w ten sposób może drukować dokumenty fiskalne na dowolnym, wspieranym urządzeniu, drukować dokumenty A4 „w tle", skanować dokumenty itp.
Jeśli z jakiegoś powodu programista nie może nawiązać połączenia TCP/IP lub korzystać z komunikacji HTTP z serwerem bsxPrinter, może użyć współdzielonego folderu i sterować procesem drukowania paragonów, czy innymi czynnościami za pomocą prostych plików tekstowych. Istnieje także możliwość umieszczania takich plików tekstowych na serwerze FTP lub SFTP. Wówczas bsxPrinter może połączyć się z tym serwerem i szukać na nim stosownych rozkazów do wykonania.
Wreszcie, jeśli programista tworzy aplikację webową, np. sklep internetowy, może stworzyć odpowiednią wtyczkę do bsxPrinter, by to bsxPrinter łączył się do serwera bazodanowego, sprawdzał czy oczekują w nim dokumenty do wydrukowania, i drukował te dokumenty automatycznie. bsxPrinter pozwala na nawiązywanie połączeń do serwerów bazodanowych: SQLite, MySQL, MariaDB, MS SQL, Firebird, PostgreSQL. Pozwala na analizę i przetwarzanie plików XML i wiele więcej.

Od wersji bsxPrinter 19.10.18 istnieje także możliwość skorzystania z bramki BinSoft Gate. W tym celu wystarczy wywoływać określone API dostarczone przez BinSoft i w ten sposób zlecać wydruk paragonów, sprawdzać ich status itp.

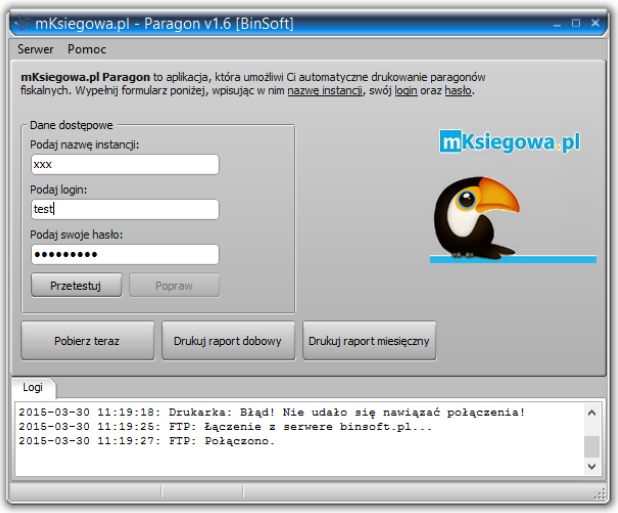
Od wersji 20.1.1 bsxPrinter wyposażony został w funkcję emulatora drukarki fiskalnej (w chwili obecnej protokół Posnet). Dzięki temu programista przygotowując swoją integracje, nie potrzebuje fizycznie urządzenia fiskalnego. W zupełności wystarczy mu sam bsxPrinter.

Od wersji 20.8.1 bsxPrinter pozwala na wydruki na klasycznych drukarkach oraz drukarkach etykiet. W prosty sposób można zatem zaimplementować te funkcjonalności u siebie.

Od wersji 23.7.31 dostępna jest obsługa KSeF (Krajowego Systemu e-Faktur). Dzięki temu możliwe jest proste integrowanie się z tą usługą Ministerstwa Finansów.

DOSTOSOWANIE APLIKACJI DO POTRZEB UŻYTKOWNIKA

Aplikacja bsxPrinter umożliwia dostosowanie jej według wytycznych programisty. Możliwe jest wyłączenie trybu podstawowego - tak by użytkownik widział jedynie panel logów, miał dostęp do ustawień programu i ewentualnie przygotowanych przez programistę wtyczek. Wreszcie, programista może ustalić informacje adresowe jakie mają się pojawić w oknie "O programie" oraz... ustalić dowolną nazwę dla aplikacji. W ten sposób deweloper może przygotować własną wersję bsxPrinter, opatrzoną własną nazwą i wyglądem i taki program oferować swoim klientom - jako np. opcja dodatkowo płatna lub wraz z paczką instalacyjną swojej aplikacji. Oto przykładowe wdrożenia:



LICENCJONOWANIE

Istnieją trzy modele współpracy licencyjnej dla aplikacji bsxPrinter.

Wariant 1

Użytkownik końcowy może kupić oprogramowanie bsxPrinter od producenta BinSoft lub jednego z Partnerów. Następnie może go odpowiednio skonfigurować lub doinstalować do niego wtyczkę dostarczoną od developera danego rozwiązania (Partnera) lub stworzyć własną wtyczkę dostosowującą program do jego aplikacji.

Wariant 2

Firma BinSoft może przygotować indywidualną wersję aplikacji bsxPrinter. Może ona mieć zupełnie zmieniony wygląd, nazwę, stosować z góry ustaloną metodę komunikacji itp. Deweloper otrzymuje dostęp do panelu B2B i tam generuje klucze dla swoich Klientów. Użytkownik końcowy będzie musiał zarejestrować swój program otrzymanym kluczem.

Po roku będzie musiał przedłużyć swoje klucze u dewelopera. Ceny za poszczególne klucze to standardowe ceny z aktualnego cennika pomniejszone o rabat dewelopera (20% na start). Fakturowanie za wygenerowane klucze odbywa się raz na koniec miesiąca rozliczeniowego.

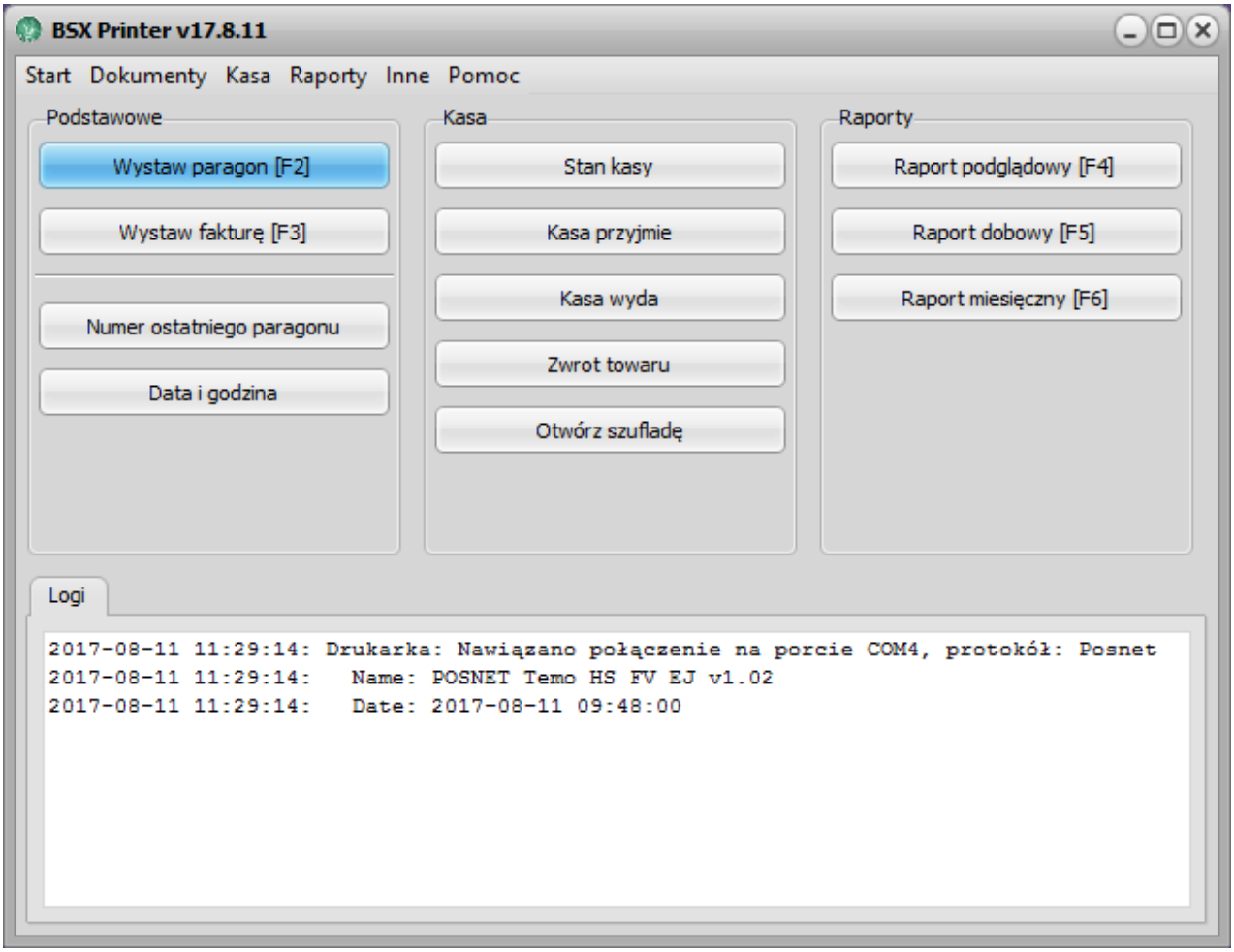
Wariant 3

Podobnie jak przy wariancie 2. firma BinSoft może przygotować indywidualną wersję aplikacji bsxPrinter. Jest ona jednak pozbawiona zabezpieczenia kluczami. Użytkownik końcowy po jej instalacji nie musi rejestrować programu, przedłużać kluczy itp. Nie ma zatem żadnego ograniczenia co do liczby stanowisk czy ważności licencji. Koszt takiej wersji jest ustalany indywidualnie.

Aby uzyskać więcej szczegółów na temat licencjonowania, należy skontaktować się z producentem BinSoft pod adresem e-mail: kontakt@binsoft.pl i ustalić szczegóły współpracy.

INTERFEJS PROGRAMU

Po uruchomieniu aplikacji bsxPrinter pojawi się okno podstawowe podobne do tego poniżej:



Okno to składa się z następujących elementów:

- *Pasek menu* - zawierający dostęp do wszystkich funkcji oferowanych przez program;
- *Przyciski* - zapewniające dostęp do najczęściej wykorzystywanych funkcji;
- *Okno logów* - prezentujące logi (dziennik zdarzeń) związane z działaniem aplikacji.

Developer ma możliwość modyfikacji tego okna, np. ukrywając wszystkie dostępne tu przyciski. Jeśli zatem przygotowuje własną aplikację, która będzie się komunikowała

z bsxPrinter poprzez jeden z dostępnych modeli, oraz planuje dołączyć bsxPrinter do swojej wersji instalacyjnej, okno bsxPrinter może maksymalnie uprościć. Obsługa procedury drukowania będzie się bowiem odbywała za pośrednictwem aplikacji programisty.

KONFIGURACJA PROGRAMU

Pierwszą czynnością jaką należy wykonać po zainstalowaniu i uruchomieniu aplikacji bsxPrinter jest jej konfiguracja. W tym celu należy z menu **Start** wybrać **Ustawienia**. Okno konfiguracji składa się z czterech zakładek: **Monitorowanie**, **Integracje,** **Urządzenia** oraz **Zaawansowane**.

ZAKŁADKA MONITOROWANIE

Zakładka **Monitorowanie** zawiera podstawowe ustawienia programu. Ich znaczenie jest następujące:

- *Uruchamiaj program wraz ze startem systemu* - zaznaczenie tej opcji spowoduje, że aplikacja bsxPrinter będzie automatycznie uruchamiana w momencie startu systemu Windows;
- *Minimalizuj przy zamykaniu* - zaznaczenie tej opcji spowoduje, że klikając ikonkę X w celu zamknięcia bsxPrinter sprawimy, że zostanie on jedynie ukryty w zasobniku systemowym nadal będąc w pełni funkcjonalnym;
- *Serwer TCP/IP | Telnet | WebSocket* - zaznaczenie tej opcji spowoduje uruchomienie serwera TCP/IP. Dzięki temu inne aplikacje będą mogły połączyć się z bsxPrinter poprzez owy protokół i drukować w ten sposób dokumenty fiskalne. Z obsługą serwera TCP/IP wiążą się dwie opcje: *Port* - określa port, na którym pracuje program oraz *Hasło* - określa hasło bezpieczeństwa.
  Po określeniu hasła bezpieczeństwa tylko aplikacje znające to hasło będą mogły podłączyć się do serwera.
- *Serwer HTTP / HTTPS* - zaznaczenie tej opcji spowoduje uruchomienie serwera HTTP. Dzięki temu będzie możliwe komunikowanie się z bsxPrinter poprze odpowiednie wywołania URL. Podobnie jak w przypadku serwera TCP/IP - należy określić port na jakim serwer będzie uruchomiony oraz hasło komunikacji.

- *Folder lokalny* - zaznaczenie opcji powoduje, że bsxPrinter obserwuje zawartość folderu zdefiniowanego poniżej tej opcji, w poszukiwaniu zadań do wykonania. Poszukiwane są pliki z rozszerzeniem *.in* lub *.xml* zawierające kod do wykonania.
- *Folder FTP/SFTP* - zaznaczenie tej opcji oraz podanie poniżej danych dostępowych spowoduje, że bsxPrinter połączy się z wybranym serwerem FTP lub SFTP i będzie na nim poszukiwał plików *.in*.
- *Odpytywanie HTTP/HTTPS* - zaznaczenie tej opcji spowoduje, że bsxPrinter będzie automatycznie wywoływał podane adresy URL w celu wymiany informacji w formacie XML. Można w ten sposób podać trzy różne adresy URL.
- *BinSoft Gate* - zaznaczenie tej opcji spowoduje wygenerowanie unikalnego klucza. Następnie zlecanie paragonów może się odbywać poprzez API dostarczone przez BinSoft. W API tym wystarczy posługiwać się tym kluczem przy zlecaniu paragonów czy też uzyskiwaniu informacji o statusie danego paragonu.
- *Klient WebSocket* - zaznaczenie tej opcji spowoduje, że bsxPrinter będzie próbował połączyć się z wybranym serwerem WebSocket.

UWAGA! *Jeśli serwer TCP/IP i/lub HTTP w aplikacji bsxPrinter jest uruchomiony, a mimo to zewnętrzne aplikacje nie chcą się z nim połączyć, należy sprawdzić czy port na którym pracuje jest odblokowany na firewall-u. Informacje o tym jak tego dokonać należy szukać w systemie pomocy Windows lub u administratora systemu.*

ZAKŁADKA URZĄDZENIA

Na zakładce **Urządzenia** dostępne są cztery sekcje **Drukarki fiskalne**, **Terminale płatnicze**, **Drukarki klasyczne** i **Skanery**.

W sekcji **Drukarki fiskalne** możemy dodawać dowolnie wiele drukarek fiskalnych podłączonych do danego komputera lub dostępnych poprzez sieć LAN. Aby dodać drukarkę należy kliknąć przycisk **Dodaj**. W oknie, które się wówczas pojawi należy podać *Protokół*, port *COM* lub adres *IP* i *Port* dla komunikacji w sieci LAN. Dostępne tam są również opcje dodatkowe:

- *Kodowanie znaków* - należy ustawić kodowanie, jakie jest skonfigurowane w drukarce fiskalnej;
- *Drukuj NIP nabywcy* - należy zaznaczyć, jeśli chcemy drukować NIP nabywcy na paragonie;
- *Drukuj NIP nabywcy jako dodatkowa linia* - jeśli korzystamy z drukarki fiskalnej, która nie oferuje funkcji drukowania NIP-u nabywcy, możemy zaznaczyć dodatkowo tę opcję. Wówczas NIP nabywcy będzie drukowany na drukarce w postaci dodatkowe linii informacyjnej;

W sekcji **Terminale płatnicze** możemy skonfigurować terminale płatnicze, jakie mamy dostępne. W celu skonfigurowania terminala płatniczego należy kliknąć przycisk **Dodaj** i w formularzu, który się pojawi podać adres *IP* tego terminala, *port* na którym nasłuchuje oraz określić jego *Protokół*. Dostępne są tam również opcje dodatkowe:

- *Drukuj potwierdzenie na drukarce fiskalnej* - jeśli posiadamy terminal bez funkcji wydruków, wówczas zaznaczenie tej opcji spowoduje, że bsxPrinter będzie drukował potwierdzenia transakcji na podłączonej drukarce fiskalnej;

W sekcji **Drukarki klasyczne** dodajemy standardowe drukarki, w tym drukarki etykiet.

W sekcji **Skanery** możemy dodać skanery, które posiadamy i z których chcemy korzystać.

ZAKŁADKA KSEF

W zakładce **KSeF** możemy konfigurować integrację z usługą KSeF (Krajowy System e-Faktur). Poszczególne opcje mają następujące znaczenie:

- *Środowisko* - możliwość wyboru środowiska, w którym pracujemy. Jeśli chcemy przetestować integracje z KSeF możemy wybrać w tym miejscu Ś*rodowisko testowe*.
- *Włącz obsługę Krajowego Systemu e-Faktur (KSeF)* - włączenie integracji;
- *NIP Firmy* - NIP firmy, w kontekście której działa integracja;
- *Token* - token autoryzacyjny do usługi KSeF;
- *Import faktur* - w tej sekcji możemy określać, czy system ma również automatycznie pobierać wystawione nam faktury, a jeśli tak - od której daty.

ZAKŁADKA ZAAWANSOWANE

W zakładce **Zaawansowane** dostępne są następujące opcje:

- *Automatyczne rozłączanie z drukarką fiskalną* - domyślnie bsxPrinter nawiązuje połączenie z drukarką fiskalną w momencie uruchomienia programu i trzyma to podłączenie tak długo, aż program zostanie wyłączony; W tym momencie dostęp do drukarki zablokowany jest dla innych aplikacji. Zaznaczenie tej opcji spowoduje, że bsxPrinter będzie łączył się z drukarką dopiero w momencie, kiedy będzie próba jakiegoś wydruku, a po tym wydruku będzie się z nią automatycznie rozłączał;
- *Automatyczne podtrzymywanie połączenia z drukarką fiskalną* - w przypadku stałego połączenia z drukarką fiskalną, niektóre modele urządzeń będą same rozłączały połączenie kiedy przez pewien czas nie dostaną żadnej komendy. Zaznaczenie tej opcji spowoduje, że bsxPrinter będzie co pewien czas komunikował się z drukarką w celu podtrzymania połączenia;

- *Nie próbuj ponownego wydruku w przypadku błędu* - podczas komunikacji z bsxPrinter i zlecania mu wydruków mogą występować różnego typu błędy. Niektóre błędy - np. brak papieru, odłączona drukarka fiskalna, problem z siecią - powodują, że bsxPrinter wstrzymuje swoją pracę, ale po pewnym czasie próbuje wznowić taki wydruk - do skutku. Zaznaczenie tej opcji spowoduje, że wystąpienie opisanych wyżej błędów nie będzie powodowało ponownych prób wydruku;
- *Emulator drukarki fiskalnej* - włączenie tej opcji i określenie *Protokołu* i *Portu* spowoduje uruchomienie wbudowanego w bsxPrinter emulatora. W oknie głównym pojawi się dodatkowa zakładka *Emulator*, gdzie będą prezentowane wydruki dokumentów. Funkcja emulatora dostępna jest tylko dla zarejestrowanych wersji bsxPrinter.

E-PARAGON

Począwszy od wersji 22.8.10 (BETA) bsxPrinter obsługuje tzw. e-paragony. W tym miejscu zostały przedstawione podstawowe informacji na temat zasady działania tych dokumentów oraz sposobu ich obsługi w aplikacji.

DRUKARKI FISKALNE ONLINE

Już od kilku lat producenci drukarek fiskalnych zaczęli sprzedać tzw. *drukarki fiskalne online* - dostosowując się w ten sposób do nowych przepisów. Drukarki te charakteryzowały się tym, że oprócz klasycznego wydruku paragonu wysyłały go automatycznie do Ministerstwa Finansów. Dzięki temu MF miało stały podgląd na wystawiane dokumenty, miało ułatwioną również możliwość przeprowadzania kontroli itp. Drukarka jednak nadal drukowała paragon fiskalny i trzeba go było fizycznie przekazać klientowi.

Od 2022 roku nastąpiła rewolucja i wprowadzono tzw. e-paragon. Możliwe się zatem stało wystawianie dokumentów fiskalnych i nie drukowanie ich fizycznie na drukarce lecz dostarczenie ich do klienta w innej formie, np. poprzez e-mail czy SMS. Aby skorzystać z tej możliwości nadal trzeba jednak posiadać drukarkę fiskalną online, uzyskać zgodę klienta na otrzymanie dokumentu fiskalnego tą drogą oraz posiadanie stosownego oprogramowania, które pozwoli na wykonanie tych czynności.

Nie wszystkie dotychczasowe drukarki online obsługują e-paragony. W chwili obecnej takie funkcje posiadają drukarki: **Posnet Thermal XL2 S1**, **Posnet Thermal HS**, **Novitus HD II Online**.

DYSTRYBUTOR E-DOKUMENTÓW

Obsługa e-Dokumentów wiąże się także z wprowadzeniem dodatkowego podmiotu zwanego **Dystrybutorem e-Dokumentów**. To jego rolą jest dostarczanie dokumentów

fiskalnych do klientów. Przykładem takiego dystrybutora, z którym współpracuje oprogramowanie bsxPrinter, jest aleParagon.pl, ale też BinSoft Gate, abcFaktury i inne**.**

PRZEPŁYW INFORMACJI

Każdy e-Paragon powinien mieć nadany specjalny **identyfikator**. Kiedy zlecamy drukarce fiskalnej przetworzenie e-Dokumentu wysyłamy również owy identyfikator. Drukarka wysyła wówczas dokument zarówno do Ministerstwa Finansów jak i do „zaprogramowanego" dystrybutora.

Warto tutaj zauważyć, że ani Ministerstwo Finansów, ani Dystrybutor - nie mają informacji o „kliencie". Nie mają jego adresu e-mail, numeru telefonu itp. Z tego względu oprogramowanie, z którego korzystamy (np. bsxPrinter) powinien dodatkowo skomunikować się z dystrybutorem i przekazać mu informacje, że gdy otrzyma dokument z drukarki fiskalnej o zadanym **identyfikatorze**, to powinien go wysłać na zadany adres e-mail.

SCHEMAT DZIAŁANIA NA PRZYKŁADZIE ALEPARAGON.PL

Aby skorzystać z funkcjonalności e-Dokumentów na przykładzie dystrybutora **aleParagon.pl**, należy wykonać następujące czynności:

1. Należy posiadać drukarkę fiskalną on-line - która obsługuje wspomnianą funkcję.
2. Należy posiadać wybranego *Dystrybutora e-paragonów*. bsxPrinter obsługuje kilku dystrybutorów, w tym m.in.: **aleParagon.pl**. Należy zatem założyć u tego dystrybutora konto i wprowadzić je w ustawieniach drukarki fiskalnej w bsxPrinter. Następnie bsxPrinter „zaprogramuje" drukarkę tak, by przesyłała ona dokumenty do tego dystrybutora.
3. Należy uzyskać zgodę klienta aby ten otrzymał dokument fiskalny w formie elektronicznej i uzyskać od niego adres e-mail.
4. W momencie zlecania wydruku paragonu/faktury należy podać adres e-mail klienta. Przy protokole XML jest to atrybut `email` w znaczniku `<receipt>`, a przy komendach `#RECEIPT` i `#INVICICE` dodatkowe pole.
5. To wszystko!

bsxPrinter kiedy otrzyma adres e-mail klienta i rozpozna, że ma skonfigurowaną drukarkę fiskalną z włączoną opcją e-dokumentów zajmie się wszystkimi pozostałymi czynności. Proces ten wygląda następująco:

1. bsxPrinter zleca wydruk dokumentu drukarce fiskalnej informując ją o tym, że będzie to e-dokument oraz przekazując nadany przez siebie **unikalny identyfikator**.
2. Drukarka fiskalna przetwarza dokument i wysyła go do Ministerstwa Finansów.
3. Drukarka fiskalna zamiast wydrukować paragon przesyła go do zaprogramowanego *Dystrybutora e-Dokumentów*.
4. bsxPrinter komunikuje się z *Dystrybutorem e-Dokumentów* i zleca mu wysyłkę dokumentu o wygenerowanym **identyfikatorze** pod wskazany adres e-mail.

E-PARAGONY PRZY OBSŁUDZE BINSOFT GATE

BinSoft Gate to jedna z metod komunikacji z programem bsxPrinter. Polega ona na wywoływaniu określonych metod API dostępnych pod adresem api.bsxprinter.pl i w ten sposób przekazywanie do bsxPrinter dokumentów do wydruku. Więcej na ten temat można odnaleźć w dalszej części podręcznika, gdzie te wywołania są wyjaśnione dokładniej. W tym miejscu zostaną przedstawione dodatkowe informacje omawiające użycie e-Paragonów w kontekście BinSoft Gate.

Mając wybranego dystrybutora e-paragonów np. aleParagon.pl - zlecanie e-Paragonów poprzez BinSoft Gate wygląda „standardowo". Wystarczy tylko w znaczniku `<receipt>` podać dodatkowo adres e-mail, np.:

```
https://api.bsxprinter.pl/insert?
key=KLUCZ_BINSOFT_GATE&data=<root><receipts><receipt
email="adres@email.pl"><item name="Product" price="1" /></receipt></
receipts></root>
```

Można jednak zaprogramować w drukarce, by jako dystrybutora używała ona również bramki **BinSoft Gate**. Wystarczy wejść w ustawienia drukarki fiskalnej i wskazać takiego właśnie dystrybutora, następnie kliknąć przycisk *„Zaprogramuj"*. Po wybraniu dystrybutora BinSoft Gate, drukarka fiskalna będzie wysyłała e-Paragony oprócz do Ministerstwa Finansów, właśnie do wspomnianego dystrybutora.

Bramka (dystrybutor) BinSoft Gate nie wysyła jednak paragonów do klientów, lecz je przechwytuje z drukarki fiskalnej. Zlecając zatem wydruk jak w poprzednim przykładzie,

jeśli otrzymalibyśmy `id` zleconego wydruku np. `828211`, moglibyśmy sprawdzić informacje o tym paragonie w ten sposób:

```
https://api.bsxprinter.pl/edocument?key=KLUCZ_BINSOFT_GATE&id=828211
```

W ten sposób otrzymalibyśmy informacje o statusie danego dokumentu, a w dodatkowym polu `ndocraw` - informacje o paragonie, jakie wysłała drukarka fiskalna. Są to dane „surowe" RAW - dokładnie tak, jak przesyła je drukarka fiskalna, dlatego należy je przetworzyć samodzielnie.

Do wywołania można przekazać dodatkowy atrybut `format=html`, a wówczas otrzymamy wizualizację wydruku w postaci dokumentu HTML. Dostępny jest jeszcze format: `json` (zdekodowany plik, ale nadal w formacie danej drukarki fiskalnej).

Przy tym wariancie osoba integrująca się z bsxPrinter powinna samodzielnie pobrać dokument z BinSoft Gate i przekazać go klientowi.

Komunikacja bez adresu e-mail

Aby bsxPrinter rozpoznał, że mamy do czynienia z e-Dokumentem w znaczniku `<receipt>` umieszczaliśmy adres e-mail. Istnieje jednak metoda obsługi e-Paragonów bez podawania adresu e-mail przy dystrybutorze BinSoft Gate. Zlecając paragon możemy dodać w znaczniku `<receipt>` atrybut `idz`. Atrybut ten będzie jednocześnie **identyfikatorem dokumentu** (tym, który w poprzednich przykładach był generowany automatycznie).

Uwaga!

Korzystanie z bramki BinSoft Gate jako dystrybutora e-Paragonów „koszt" pojedynczego paragonu wynosi 10 tokenów (w chwili pisania tego podręcznika)

KOMUNIKACJA Z BSXPRINTER

Dostępnych jest kilka trybów współpracy zewnętrznej aplikacji z bsxPrinter.
Nawiązanie połączenia poprzez TCP/IP (Telnet, WebSocket), odpytywanie serwera HTTP, skorzystanie z współdzielonego folderu (lokalnego lub folderu na serwerze FTP/SFTP), łączenie się z wybranym serwerem WebSocket lub wymiana informacji poprzez odpytywanie określonego adresu URL.

KOMUNIKACJA POPRZEZ TCP/IP

Aplikacja bsxPrinter uruchamia automatycznie **serwer TCP/IP**, do którego można nawiązać połączenie z dowolnej innej aplikacji. Domyślnie serwer nasłuchuje na porcie 7361, aczkolwiek port może zostać zmieniony w ustawieniach programu.

Zaimplementowany w bsxPrinter serwer TCP/IP obsługuje kilka pod-protokołów. Są to:

- protokół **bsxPrinter** - opisany poniżej
- protokół **Telnet** (dzięki temu można połączyć się z bsxPrinter z dowolnego klienta Telnet, np. Putty)
- protokół **WebSocket** (ws + wss)
- protokół **Thermal** / **Posnet** / **Novitus XML** - do bezpośredniej komunikacji z drukarką fiskalną

PROTOKÓŁ BSXPRINTER

Serwer obsługuje komendy różnego typu. Znacznik typu to jednobajtowy kod wysyłany zawsze na początku komendy. W chwili obecnej programista może skorzystać tylko z typu podstawowego - tekstowego.

- **Typ tekstowy:** `[01]Ciąg znaków[13][10]`

W typie podstawowym, programista wysyła bajt o wartości ASCII `#01`, a następnie komendę w postaci ciągu znaków ASCII, zakodowanych w UTF8 i zakończonym bajtami `#13`, `#10`.
Odpowiedź serwera także może być różnego typu. Typ określony jest pierwszym bajtem odsyłanym przez serwer. Jeśli zatem bajt odpowiedzi będzie miał wartość `#01`, należy oczekiwać ciągu znaków, zakodowanego w UTF8, aż do napotkania bajtów `#13`, `#10`.

W dalszej części podręcznika będą omówione wszystkie komendy obsługiwane przez bsxPrinter. Są one bowiem wspólne w tej metodzie wymiany informacji i plikach .in, które bsxPrinter może przetwarzać automatycznie.

Przykład. Chcąc uzyskać listę dostępnych portów szeregowych, należy wydać komendę `COMLIST`. Pełny ciąg znaków wysyłanych do serwera powinien mieć zatem postać:

```
[01]COMLIST[13][10]
```

Jeśli w ustawieniach programu użytkownik ustawi hasło bezpieczeństwa, wówczas pierwszą czynnością po nawiązaniu połączenia z serwerem powinno być "zalogowanie się" za pomocą komendy `PASSWORD`. Bez procesu uwierzytelniania większość komend nie będzie dostępna.

UWAGA! *Każda komenda wysyłana do bsxPrinter i zwracana w odpowiedzi z bsxPrinter zakończona jest dwuznakiem o kodach ASCII 13, 10. Oznacza to, że wewnątrz komendy nie można używać tych znaków, gdyż może to zaburzyć komunikację. Dlatego jeśli do jakiejś komendy chcemy przekazać jako parametr te symbole, należy*

w ich miejsce użyć ciągu znaków: ^13^ (dla oznaczenia kodu ASCII 13) i ^10^ (dla oznaczenia kodu ASCII 10).

TELNET

Począwszy od wersji 19.1.1 bsxPrinter obsługuje w ograniczonym zakresie komunikację wg. protokołu *Telnet*. Oznacza to, że można się z nim połączyć dowolną aplikacją obsługującą ten protokół i komunikować się prostymi komendami. Przy tym protokole przed komendami nie ma potrzeby wysyłania bajtu `#01`, lecz od razu wysyłamy ciąg znaków zakończony `#13#10`. bsxPrinter również nie odpowiada bajtem początkowym `#01`, lecz ciągiem znaków.

Dzięki powyższej funkcji w prosty sposób można przetestować działanie aplikacji bsxPrinter jak również z użyciem typowych narzędzi (np. Putty) się z nim połączyć.

WEBSOCKET

Począwszy od wersji 22.7.25 bsxPrinter obsługuje także protokół WebSocket. Po nawiązaniu połączenia na wybranym porcie TCP/IP bsxPrinter rozpozna, że nadawca komunikuje się według tego protokołu i odpowiednio przełączy swoją obsługę. Obsługiwany jest również WebSocket poprzez SSL (wss). Wówczas należy łączyć się na port TCP/IP + 1. Domyślnie bsxPrinter używa certyfikatu samo-podpisanego, dlatego by takie połączenie zostało nawiązane może być konieczne dodanie tego certyfikatu do listy zaufanych.

Oto przykład prostej strony demonstrującej komunikację poprzez WebSocket.

```
<!DOCTYPE HTML>
<html>
   <head>
      <script type = "text/javascript">
      var ws = false;
      function show(s) {
        txtArea = document.getElementById("mm");
        txtArea.value += s + '\r\n';
      }
```

```
function send(cmd) {
  if (!ws) { show('Brak połączenia!'); return; }
  show('<'+cmd);
  ws.binaryType = "blob";
  ws.send(cmd);
}

function sendClick() {
  cmd = document.getElementById('cmd').value;
  send(cmd);
  document.getElementById('cmd').value = '';
}

function sendFile() {
  if (!ws) { show('Brak połączenia!'); return; }
  var file = document.getElementById('filename').files[0];
  var reader = new FileReader();
  var rawData = new ArrayBuffer();

  reader.loadend = function() {
  }

  reader.onload = function(e) {
     rawData = e.target.result;
     ws.binaryType = "arraybuffer";
     ws.send(rawData);
  }
  reader.readAsArrayBuffer(file);
}

function WebSocketStart() {
```

```
        if ("WebSocket" in window) {
            show("Łączenie...");
            ws = new WebSocket("ws://localhost:7363/");
            ws.onopen = function() {
              show("Połączono!");
              send('HELLO')
            }
            ws.onmessage = function (evt) {
               var received_msg = evt.data;
               show('>' + received_msg);
            }
            ws.onclose = function() {
               show("Połączenie zamknięte!");
            }
            ws.onerror = function(err) {
               show("Błąd!");
            }
         } else {
           show("Twoja przeglądarka nie obsługuje WebSocket!");
         }
      }
      </script>
   </head>
   <body>
      <a href = "javascript:WebSocketStart()">Połącz z serwerem</a><hr />
   <input type="text" name="cmd" id="cmd" /> <a href =
"javascript:sendClick()">Wyślij</a><hr />
       <h2>Upload plików</h2>
       <input type="file" id="filename" />
       <a href="javascript:sendFile()">Wyślij plik</a>
       <hr />
       <textarea name="mm" id="mm" cols="100" rows="25"></textarea>
   </body>
```

```
</html>
```

THERMAL / POSNET / NOVITUS XML

Jeśli bsxPrinter rozpozna, że nadchodzące dane są w formatach protokołów Thermal, Posnet lub Novitus XML - będzie automatycznie przekierowywał ten ruch bezpośrednio do drukarki fiskalnej. Dzięki temu możliwa jest samodzielna implementacja tych protokołów w swoich rozwiązaniach, a skorzystanie z bsxPrinter jedynie jako pośrednika zapewniającego dostęp do stacjonarnej drukarki z sieci.

SQL

bsxPrinter prowadzi wewnętrzną bazę danych, w której przechowuje informacje o wydrukowanych paragonach, transakcjach kartami płatniczymi itp. Począwszy od wersji 19.10.16 komunikacja TCP pozwala na wydawanie dowolnych komend SQL i w ten sposób dowolne pobieranie i operowanie na tych danych. Wystarczy aby komenda wysłana do bsxPrinter rozpoczynała się od `SELECT`, `UPDATE`, `DELETE`, `INSERT`, `ALTER` lub `TABLES` - a będzie ona kierowana do silnika SQL.

KOMUNIKACJA POPRZEZ SERWER HTTP/HTTPS

Aplikacja bsxPrinter może uruchomić własny serwer HTTP na wybranym porcie (domyślnie 8001). Aby sprawdzić, czy serwer działa poprawnie można spróbować wejść z przeglądarki użytkownika pod adres http://localhost:8001. Powinna pojawić się strona informacyjna, że serwer działa poprawnie.

Chcąc korzystać z tej komunikacji, wystarczy wywoływać podany adres URL przekazując metodą GET lub POST dodatkowe informacje. Niezbędne parametry to:

- `cmd=KOMENDA`
- `password=HASŁO`

W zależności od komendy, mogą być wymagane dodatkowe parametry wywołania. Serwer odpowiada danymi zakodowanymi w formacie JSON.

Jeśli wystąpi błąd, serwer odpowie `{"error":"Komunikat"}`

Serwer HTTP programu bsxPrinter przesyła nagłówek `Access-Control-Allow-Origin:*` dzięki czemu wywołanie tego adresu powinno być możliwe z innych adresów URL - w tym również z sieci. Możliwy jest zatem scenariusz, że użytkownik odwiedza stronę partnera (np. sklepu internetowego), klika w tym sklepie przycisk *„Drukuj”* i drukuje natychmiast paragon. W sklepie tym Partner może bowiem z poziomu języka JavaScript i np. technologii AJAX wywołać adres URL http://localhost:8001 i zlecić paragon do wydrukowania. Przykład:

```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <script type="application/javascript"
              src="http://code.jquery.com/jquery-3.2.1.min.js"></script>
  </head>
 <body>
 <button id="drukuj">Drukuj</button>
 <script type="application/javascript">
    $('#drukuj').click(function(){
        xmldata='<root><receipts><receipt id="m4"><item name="Abc" price="5" /
></receipt></receipts></root>';
     $.ajax({
         url: "http://localhost:8001/",
         data: {
             cmd: 'parsexml',
             password: 'BinSoftBSX',
             data: xmldata,
         },
         type: "POST",
```

```
          // dataType : "json",
     })
         .done(function( json ) {
             alert(json);
         })
         .fail(function( xhr, status, errorThrown ) {
             alert( "Sorry, there was a problem!" );
             console.log( "Error: " + errorThrown );
             console.log( "Status: " + status );
             console.dir( xhr );
         });
    });
 </script>
 </body>
</html>
```

Powyższy przykład spowoduje natychmiastowe wydrukowanie paragonu.

Należy zwrócić uwagę, że jeśli strona WWW, z poziomu której będziemy próbować połączyć się do serwera bsxPrinter, wykorzystuje połączenie szyfrowane SSL (protokół HTTPS) - wywołanie adresu http://localhost będzie zapewne zablokowane przez przeglądarkę. Wynika to z mechanizmów bezpieczeństwa, które blokują odwołania do elementów w Sieci poprzez połączenia „niebezpieczne" (niezaszyfrowane) ze strony wykorzystującej „bezpieczne" (szyfrowane) połączenie. Aby powyższe ominąć, należy również uruchomić szyfrowane połączenia w bsxPrinter. Jest to możliwe, potrzebujemy jednak do tego celu tzw. certyfikat SSL.

Jeśli posiadamy odpowiedni certyfikat, zapisujemy go w pliku o nazwie `cert.pem` i umieszczamy w folderze, razem z plikiem wykonywalnym bsxPrinter. Następnie

ponownie uruchamiamy bsxPrinter. Kiedy ten wykryje powyższy plik w swoim katalogu, uruchomi nasłuchiwanie również na porcie 443, gdzie wykorzystywać będzie połączenia szyfrowane. Obsługiwał będzie także automatyczne rozpoznawanie połączeń, więc jeśli spróbujemy wejść w przeglądarce na adres https://localhost - bsxPrinter powinien się automatycznie zgłosić.

Jeśli nie posiadamy odpowiedniego certyfikatu, możemy go sobie wygenerować samodzielnie. Będzie nam w tym celu potrzebne **OpenSSL**. Po jego pobraniu i zainstalowaniu wydajemy komendę w konsoli:

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout
cert.pem -out cert.pem
```

i postępuje zgodnie z instrukcjami, które się ukażą. Powstanie plik `cert.pem`, który umieszczamy w folderze bsxPrinter jak wspomniano wcześniej.

UWAGA! *Domyślnie bsxPrinter posiada wgrany samo-podpisany certyfikat SSL właśnie dla localhost. W folderze głównym programu znajdziemy także plik RootCA.crt z certyfikatem „wydawcy" certyfikatu cert.pem. Możemy dodać ten certyfikat w swojej przeglądarce jako zaufany, wówczas wszystkie połączenia z bsxPrinter będą nawiązywane poprawnie.*

Należy pamiętać, że powyżej wygenerowany certyfikat wystarczy do obsługi połączeń szyfrowanych SSL. Nie jest on jednak certyfikatem zaufanym, dlatego przeglądarki będą ostrzegały o takich połączeniach. Dodając jednak wyjątek dla `localhost` (czy 127.0.0.1) lub dodając certyfikat wydawcy jako zaufany, połączenia będą działać i

będzie można z nich korzystać również z poziomu stron WWW wykorzystujących połączenia SSL zaufane.

UWAGA! *Istnieje możliwość zakupienia certyfikatu zaufanego dla danej domeny np. mojlocalsklep.pl. Następnie w pliku* ***\|system32\|drivers\|etc\|hosts*** *w systemie Windows określenie by domena ta, mojlocalsklep.pl wskazywała właśnie na localhost. Dzięki takiemu postępowaniu certyfikat będzie widoczny jako zaufany.*

Dostępne komendy przy połączeniach HTTP:

parsexml - Przetworzenie pliku XML

```
cmd=parsexml
password=hasło
data=XMLDATA
```

Powoduje przetworzenie przesłanego pliku XML. Plik powinien mieć format opisany w dalszej części dokumentacji (opisano przy okazji metody monitorowania plików).
Plik XML zostaje zapamiętany w wewnętrznej bazie danych bsxPrinter i wywołanie to zwróci numer ID nadany temu wpisowi. Dzieje się tak dlatego, gdyż bsxPrinter przetwarzania pliki asynchronicznie, z opóźnieniem.

parsein - Przetworzenie pliku IN

```
cmd=parsein
password=hasło
data=INDATA
```

Powoduje przetworzenie pliku IN. Analogicznie jak powyżej, plik powinien mieć format opisany w dalszej części dokumentacji. Wywołanie zwróci ID nadane temu dokumentowi w wewnętrznej bazie danych.

printpdf - Drukowanie pliku PDF

```
cmd=printpdf
password=hasło
data=INDATA
```

Powoduje wydrukowanie pliku PDF na klasycznej drukarce. W polu `data` należy przekazać zawartość binarną pliku PDF zakodowaną przez Base64. Funkcja zwraca numer ID nadany plikowi z dokumentem.

printhtml - Drukowanie dokumentu HTML

```
cmd=printpdf
password=hasło
data=INDATA
width=szerokość
height=wysokość
rotate=czy obrócić wydruk
```

Powoduje wydrukowanie dokumentu HTML na klasycznej drukarce. W polu `data` należy przekazać treść dokumentu HTML. Parametrami `width` i `height` definiujemy szerokość i wysokość papieru (w mm). Pole `rotate` przyjmuje wartości `true` lub `false` i określa, czy papier ma być obrócony czy nie. Funkcja zwraca numer ID nadany plikowi z dokumentem.

gethistory - Pobranie historii

```
cmd=gethistory
password=hasło
type=files|documents|ksef
[limit=po ile rekordów]
[start=od którego rekordu]
[id=ID]
[iddoc=ID]
```

Wywołanie powoduje zwrócenie listy rekordów lub pojedynczego rekordu.

Gdy `type=files`, zwrócona zostanie informacja o przesłanych plikach (XML/IN), gdy `type=documents` - informacja o przesłanych paragonach/fakturach, natomiast dla `type=ksef` - informacja o przetworzonych e-Fakturach KSeF itp. Jeśli podamy parametr `id=XX` - wówczas zwrócony zostanie jeden rekord z historii plików lub dokumentów. Jeśli podamy parametr `iddoc=ID` (dla `type=documents`) - zwrócona zostanie lista dokumentów powiązanych z danym ID pliku. Poprzez parametry `start` i `limit` można listować rekordy. Wywołanie zwraca dane w kolejności od najnowszego.

#KOMENDA - Dowolna komenda

Umieszczając w zmiennej `cmd` instrukcję zaczynającą się od `#` wywołamy dowolną komendę opisaną w dalszej części podręcznika. Może to być np. wywołanie okna formularza płatności itp. Możliwe jest również umieszczenie wielu komend umieszczając je w kolejnych liniach.

Przykładowy scenariusz:

1. Przekazujemy metodą `cmd=parsexml` dane o paragonach do wydrukowania w formacie XML;
2. bsxPrinter zwróci ID nadane temu plikowi;
3. Wywołujemy `cmd=gethistory`, `type=documents`, `iddoc=ID` uzyskane uprzednio, by poznać status tych paragonów. Czy zostały wydrukowane lub wystąpiły błędy.

Pola opisujące pliki:

id - ID wpisu w bazie danych

status - status przetworzenia pliku

date - data wprowadzenia pliku

errorcode - kod błędu

errorstr - opis błędu

Pola opisujące dokumenty:

id - ID wpisu w bazie danych

status - status wydruku: 0=W kolejce, 1=W trakcie wydruku, 2=Wydrukowano, 5=Błąd (będzie ponowienie), 6=Błąd (Nie będzie ponowienia)

data - data dodania wpisu

source - źródło wpisu: 0=TCP/IP, 1=XML, 2=Ręcznie, 3=XML (Request), 4=Plik (Monitor), 5=Plik (FTP), 6=HTTP (Request);

type - typ dokumentu: 1=Paragon, 2=Faktura

send - czy przesłano odpowiedź do serwera

errorcode - kod błędu

errorstr - opis błędu

nodoc - nadany numer dokumentowi (format RID/DATA/RIDG)

rid - nadany przez drukarkę numer dokumentowi (wzgl. raportu dobowego)

ridg - nadany przez drukarkę numer dokumentowi (numer globalny)

remoteid - numer ID przekazany w pliku XML

printdate - data wydruku

Jeśli nie wystąpiły błędy - pola `errorcode` i `errorstr` będą puste.

Drukarki fiskalne nadają numery paragonom/fakturom. Nadawane są dwa numery. Jeden, tzw. dobowy (licznik jest zerowany po wydrukowaniu raportu dobowego) i drugi globalny. Pierwszy przedstawiany jest w bsxPrinter jako `RID`, drugi to `RIDG`. W bsxPrinter wprowadzono jeszcze jeden numer, ogólny wyświetlany dla użytkownika, jest nim `nodoc`.

UWAGA! *Niektóre drukarki nie zwracają informacji o obu numerach. Stąd może się okazać, że na niektórych z nich RID=RIDG.*

KOMUNIKACJA POPRZEZ WSPÓŁDZIELONY FOLDER

Kolejna z metod współpracy zewnętrznych programów z bsxPrinter - to współdzielony folder. W ustawieniach programu bsxPrinter można wskazać folder, który będzie "nasłuchiwany". bsxPrinter będzie szukał w tym folderze plików z rozszerzeniem: `.in` lub `.xml`. Dotyczy to również serwera FTP, jeśli ustawione jest na nim nasłuchiwanie.

Po odnalezieniu takiego pliku, automatycznie zmieniane jest mu rozszerzenie na `.pr` (co oznacza, że plik został przetworzony) - a następnie wykonywane są komendy umieszczone w tym pliku. Na koniec tworzony jest plik z rozszerzeniem .out zawierający wynik działania przekazanych komend.

Programista może zatem utworzyć we współdzielonym folderze plik tekstowy z rozszerzeniem `.in`, w którym zapisze instrukcje jakie ma wykonać bsxPrinter. Te instrukcje to może być ciąg komend drukujących paragon. Następnie aplikacja programisty może oczekiwać na plik o tej samej nazwie, lecz rozszerzeniem `.out`. Kiedy się pojawi, może odczytać jego zawartość i na tej podstawie uzyskać informacje o wyniku wykonania przekazanych komend.

Wszystkie komendy w pliku tekstowym powinny być zapisywane w oddzielnych liniach. Plik powinien być zakodowany w ANSI (Windows 1250) lub UTF-8.

Ostatnia linia pliku (`.in`) powinna zawierać komendę `EXECUTE`. Jeśli w pliku nie będzie tej komendy, bsxPrinter w ogóle nie będzie przetwarzał takiego pliku. (Jest to dodatkowe zabezpieczenie przed próbą przetworzenia pliku, zanim nie zostanie on do końca zapisany).

Uwaga! Od wersji 19.11.26 bsxPrinter oprócz folderu wskazanego w ustawieniach, przeszukuje również podfoldery z tego folderu, nazywające się tak, jak skonfigurowane drukarki fiskalne. Jeśli wykryje w nich pliki, które obsługuje, przetworzy je i wydrukuje na danej drukarce.

Na przykład, jeśli mamy dwie drukarki skonfigurowane pod nazwami *Drukarka1* i *Drukarka2* oraz wskazany w ustawieniach folder: *Z:\Paragony* - wówczas bsxPrinter będzie przeszukiwał foldery:

- `Z:\Paragony` - i paragony z tego folderu będzie drukował na drukarce domyślnej;
- `Z:\Paragony\Drukarka1` - i te paragony będzie zawsze drukował na *Drukarce 1*;
- `Z:\Paragony\Drukarka2` - i te paragony będzie zawsze drukował na *Drukarce 2*;

KOMUNIKACJA POPRZEZ XML

Ta forma współpracy polega na tym, że to aplikacja bsxPrinter wywołuje określony adres URL i przekazuje do niego parametry metodą POST. W odpowiedzi na takie żądanie, serwer powinien odpowiedzieć odpowiednio wygenerowanym plikiem XML.

Procedura wygląda następująco:

1. Użytkownik ustala i podaje w konfiguracji programu adres URL, jaki ma być wywoływany co określony czas (np. co 30 sekund - użytkownik może ten czas modyfikować w Ustawieniach).
2. bsxPrinter wywołuje wspomniany adres URL i przekazuje metodą POST parametr `cmd=getreceipts`.
3. bsxPrinter buforuje ower paragony, a następnie je drukuje. Po wydrukowaniu ponownie wywołuje wspomniany adres URL tym razem przekazując w parametrze

`cmd=results` oraz w parametrze `results` ciąg XML zawierający opis wydrukowanych paragonów (status wydruku).

Skrypt partnera zwraca wówczas wartość OK lub 1 lub `<root>ok</root>` informując w ten sposób, że poprawnie zanotował wynik wydruku paragonów.

Na żądanie `cmd=getreceipts` skrypt powinien zwrócić plik XML według formatu (fragment):

```
<?xml version="1.0" encoding="utf-8"?>

<root>
 <receipts>
   <receipt id="IDENTYFIKATOR_PARAGONU(np. numer zamówienia)"
            step="PRÓBA=0,1..."
            barcode="numer do wydrukowania jako kod kreskowy"
            total="WARTOŚĆ_PARAGONU"
            cash="ILE_WPŁACONO_GOTÓWKĄ"
            card="ILE_WPŁĄCONO_KARTĄ"
            other="ILE_WPŁACONO_INNĄ_METODĄ"
            othername="NAZWA_INNEJ_METODY_PŁATNOŚCI"
            rest="RESZTA">


     <item name="NAZWA_PRODUKTU"
           price="CENA"
           quantity="SZTUK"
           vatrate="STAWKA_VAT"
           total="WARTOŚĆ" />
      …

   <receipt>
   …
 </receipts>

<root>
```

Jeśli nie ma paragonów do wydrukowania, to w znaczniku `<receipts>..</receipts>` nie przekazujemy żadnych znaczników `<receipt>` - natomiast dane nadal mają wspomniany format. W jednym wywołaniu może znajdować się dowolnie wiele paragonów - należy umieścić wówczas odpowiednie sekcje `<receipt>…</receipt>` dla każdego z paragonów.

UWAGA! *Poniżej opisano szczegóły poszczególnych znaczników. Format XML opisany w tym miejscu dotyczy również plików XML analizowanych w wariancie „monitorowania folderu" jak również danych przekazywanych przy metodzie „serwera HTTP", czy też metodzie BinSoft Gate.*

Przekazując pojedynczy paragon (znacznik `<receipt>`) określamy dla niego parametry:

- **[printer]** - nazwa drukarki, na której ma być wykonany wydruk;
- [**id]** - jest to identyfikator wydruku, po którym go rozpoznajemy w naszym systemie; może to być np. numer zamówienia; Program weryfikuje to pole i blokuje dodanie kolejnego wydruku z tym samym numerem id.
- [**step]** - liczba całkowita określająca „próbę" wydruku; Powinna być to wartość 0. Kiedy ponownie przekażemy dokument o tym samym id - bsxPrinter go drugi raz nie wydrukuje (nawet, jeśli za pierwszym razem wydruk się nie powiódł). Aby zatem ponowić wydruk należy kolejny raz przekazać dany paragon zmieniając ten parametr na kolejną liczbę całkowitą;
- [**total]** - wartość całkowita paragonu; (bez rabatów)
- [**cash]** - kwota wpłaty przez klienta w gotówce;
- [**card]** - kwota wpłaty przez klienta kartą płatniczą;
- [**other]** - kwota wpłaty przez klienta inną metodą płatności;
- [**othername]** - nazwa innej metody płatności;
- [**rest]** - kwota reszty do wydania klientowi;
- **[terminal]** - nadanie wartości „true" i przekazanie kwoty w polu **card** spowoduje włączenie obsługi terminala płatniczego;
- **[printconfirm]** - nadanie wartości „true" powoduje automatyczne drukowanie kopii potwierdzenia transakcji; Gdy pole przyjmie wartość „false" - kopia potwierdzenie

nie będzie drukowana; Wartość domyślna „auto” powoduje wyświetlenie pytania czy wydrukować kopię potwierdzenia.

- **[discounttype]** - rodzaj nadania rabatu: 0 - procentowo, 1 - kwotowo;
- **[discountname]** - nazwa rabatu;
- **[discountvalue]** - wartość rabatu;
- **[type]** - rodzaj wydruku: receipt (paragon), invoice (faktura), document (dokument PDF lub HTML);
- **[nodoc]** - numer faktury (dotyczy wydruku faktur);
- **[nip]** - NIP nabywcy (dotyczy wydruku faktur/paragonów);
- **[buyer]** - dane nabywcy (dotyczy wydruku faktur);
- **[paymentdate]** - termin płatności (dotyczy wydruku faktur);
- **[paymentform]** - forma płatności (dotyczy wydruku faktur);
- **[person1]** - imię i nazwisko sprzedawcy (dotyczy wydruku faktur);
- **[person2]** - imię i nazwisko odbiorcy (dotyczy wydruku faktur);
- **[email]** - adres e-mail odbiorcy (w przypadku e-Dokument jest to adres, na który będzie wysłany dokument);

Parametry podane w nawiasach kwadratowych nie są obowiązkowe.

Wewnątrz znacznika `<receipt>` może znajdować się wiele znaczników `<item>` - opisujących pozycje dla paragonów lub faktur fiskalnych, `<line>` - dodatkowe linie na paragonie, `<command>` - dodatkowe komendy do wykonania lub `<file>` - dokumenty do wydrukowania na klasycznej drukarce.

Jak podano wyżej, w przypadku paragonów i faktur fiskalnych, kolejne pozycje definiuje się znacznikiem `<item>`. Znaczenie poszczególnych pól jest następujące:

- **name** - nazwa produktu; z reguły maksymalnie 40 znaków, aczkolwiek niektóre drukarki mogą mieć inne ograniczenia;
- **price** - cena produktu;
- [**quantity]** - ilość;
- [**vatrate]** - stawka VAT (np. 23, A, PTUA, zw);
- [**total]** - wartość produktu;
- [**discount]** - czy nadać rabat/narzut na produktu; 0 - brak, 1 - rabat, 2 - narzut;
- [**discountname]** - nazwa rabatu;
- [**discountvalue]** - wartość rabatu kwotowo;
- [**discountvalueproc]** - wartość rabatu w procentach;

Chcąc nadać rabat/narzut na pozycję należy przekazać parametr `discount=1` (dla rabatu) lub `discount=2` (dla narzutu) oraz zdefiniować JEDNO z pól: `discountvalue` - by określić kwotę rabatu/narzutu lub `discountvalueprc` - by określić procentowo wartość rabatu/narzutu. Pole `discountname` nie jest obowiązkowe. Należy pamiętać, aby nadając rabat na produkcie odpowiednio uwzględnić go w wartości produktu jeśli je podajemy (pole `total`).

UWAGA! *Wszystkie kwoty na paragonie są kwotami BRUTTO. Jako separator dziesiętny używa się symbolu kropki „.”. Wszystkie kwoty podajemy z maksymalną dokładnością do 2 miejsc po przecinku.*

Po wydrukowaniu dokumentów bsxPrinter wywoła ten sam adres URL przekazując w parametrze `cmd=results`, a w parametrze `results=DATAXML` dane w XML opisujące rezultat wydruku dokumentów. Format tego pliku jest następujący:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<root>

    <receipt status="STATUS"
        id="IDENTYFIKATOR_PARAGONU(np. numer zamówienia)"
        nodoc="NUMER NADANY PRZEZ DRUKARKĘ"
        date="DATA_I_GODZINA_WYDRUKU YYYY-MM-DD HH:II:SS"
        errorcode="KOD BŁĘDU JEŚLI WYSTĄPIŁ"
        errorstr="KOMUNIKAT_O_BŁĘDZIE_JEŚLI_WYSTĄPIŁ"
    />

…

</root>
```

Dla zachowania kompatybilności ze starszą wersją bsxPrinter - znaczniki odpowiedzi mogą posiadać dodatkowe atrybuty zgodne ze starszą specyfikacją.

Na powyższe żądanie skrypt powinien zwrócić wartość OK lub 1 lub `<root>OK</root>`. W przeciwnym wypadku bsxPrinter będzie za każdym raz wysyłał status tych samych paragonów.

Poszczególne paragony opisane są znacznikiem `<receipt>`. W atrybucie **id** przekazywany jest id paragonu (przekazany wcześniej przez Partnera również w parametrze **id**), w **nodoc** znajduje się identyfikator nadany przez drukarkę. Atrybut **date** definiuje datę i godzinę wydruku, zaś **status** - status wydruku. W przypadku błędów pojawiają się dodatkowe atrybuty **errorstr** z treścią komunikatu i **errorcode** z numerem błędu.

Możliwe statusy:

- **0** - oczekuje na przetworzenie
- **1** - w trakcie przetwarzania
- **2** - przetworzony
- **-5** - wystąpił błąd - będzie ponowiona próba wydruku
- **-6** - wystąpił błąd - nie będzie ponowionej próby wydruku

Parametr **errorcode** może przyjmować różne wartości - w zależności od drukarki (kody błędów zwracane przez drukarki). Czasami przyjmuje jednak stałe wartości, tj.:

- **-400** - nie znaleziono drukarki o podanej nazwie
- **-401** - nie udało się połączyć z drukarką
- **-402** - nie można wydrukować paragonu (przekroczono limit)
- **-403** - nie udało się uaktualnić bazy liczników paragonów
- -**404** - nierozpoznany typ dokumentu
- **-450** - brak zainicjowanego terminala płatniczego;
- **-451** - brak połączenia z terminalem płatniczym;
- **-461...-469** - różne rodzaje odmowy transakcji;

Oto dwa proste przykłady:

```
<?xml version="1.0" encoding="utf-8"?>

<root>

 <receipts>

   <receipt id="1">

     <item name="Mleko" price="2.5" quantity="2"
           vatrate="23" total="5" />

   <receipt>

 </receipts>

</root>


<?xml version="1.0" encoding="utf-8"?>

<root>

 <receipts>

   <receipt id="1" step="2" nip="6572911384" >

       <item name="Mleko" price="2.5" quantity="2"
           vatrate="23" total="5" />
```

```
      <item name="Usługa" price="100" quantity="1"
          vatrate="zw" total="100" />

   <receipt>

 </receipts>

</root>
```

Dodatkowe komendy przy protokole XML

W znaczniku `<receipt>` istnieje możliwość użycia oprócz znaczników `<item>` również znacznik `<command>`, `<line>`, oraz `<file>`.

Pierwszy ze znaczników pozwala wykonać dowolną komendę na drukarce fiskalnej. Znaczników `<command>` może być wiele. Wykonywane one będą sekwencyjnie, jedna pod drugą. Komendy podajemy w atrybucie `name` i są one takie same, jak komendy opisane w podręczniku przy protokole TCP/IP czy plików `.IN`. Chcąc wykonać tylko same komendy, można stworzyć „pusty" paragon, tj. nie podawać żadnych jego pozycji (`<items>`).

Przykład:

```
<?xml version="1.0" encoding="utf-8"?>

<root>

 <receipts>

   <receipt id="2">

     <command name="CASHIN#5.00" />

   </receipt>

 </receipts>

</root>
```

Powyższy kod spowoduje wydrukowanie „kasa przyjmie" na kwotę 5 zł.

Inny przykład:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <receipts>
   <receipt>
     <command name="FormStart"/>
     <command name="FormLine#Dowolna linia"/>
     <command name="FormEnd"/>
   </receipt>
 </receipts>
</root>
```

Powyższy przykład wydrukuje „Pokwitowanie” z jedną linią „Dowolna linia”.

Znacznikiem `<line>` można zlecić wydruk dodatkowych linii na paragonie. Przekazujemy je w atrybucie `caption`, na przykład:

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <receipts>
    <receipt id="258" step="0" barcode="250" total="18.40"
     discounttype="0" discountname="Stały klient"
     discountvalue="50" cash="9.20">
      <item name="Produkt 1" price="10.45" quantity="1"
            vatrate="23" total="10.45" />
      <item name="Produkt 2" price="7.95" quantity="1"
            vatrate="23" total="7.95" />
      <line caption="Dziękujemy za zakupy" />
    </receipt>
  </receipts>
</root>
```

Drukując dokumenty klasyczne (na zwykłych drukarkach lub drukarkach etykiet) - w znaczniku `<receipt>` umieszczamy znacznik `<file>` określający plik, który ma być wydrukowany. Oto jego możliwe wartości:

- **type** - typ pliku; możliwe wartości: pdf lub html;
- **[width]** - szerokość papieru (dla typu html);
- **[height]** - wysokość papieru (dla typu html);
- **[rotate]** - czy obrócić kartkę (dla typu html, wartości: true lub false);

Treść dokumentu do wydrukowania podajemy wewnątrz tego znacznika. Dla typu HTML jest to wprost kod HTML, dla PDF - wartość binarna pliku zakodowana w Base64. Oczywiście całość powinno umieścić się w `<![CDATA[ … ]]>`. Oto przykład:

```
<root>
 <receipts>
   <receipt>
     <file type="html" width="62" height="62"><![CDATA[
       <html>
         <head>
         </head>
         <body>
           Przykładowy <b>plik</b> HTML
         </body>
       </html>
     ]]></file>
   </receipt>
 </receipts>
</root>
```

BINSOFT GATE

BinSoft Gate to nowa metoda integracji z programem bsxPrinter. Dostępna jest ona od wersji 19.10.18 i wykorzystuje pośrednictwo firmy BinSoft (lub Partnera) w komunikacji z programami bsxPrinter.

W programie bsxPrinter wybierając z menu **Start** -> **Ustawienia**, odnajdziemy w zakładce **Monitorowanie** sekcję **BinSoft Gate**. Aby włączyć tę metodę komunikacji wystarczy zaznaczyć *Korzystaj z bramy BinSoft Gate*. W polu *Klucz* wygenerowany zostanie unikalny klucz.

Następnie aby zlecić wydrukowanie paragonu na takiej drukarce wystarczy skorzystać z dostarczonego przez BinSoft bardzo prostego API.

Dane do API przekazywane są metodą GET lub POST zależnie od funkcji. Zwracane dane są zawsze zakodowane w JSON.

W odpowiedzi powinniśmy otrzymać pole `success` o wartości `true` - w przypadku powodzenia oraz dodatkowe pola zależnie od wybranej funkcji. Gdy wystąpi błąd, zamiast pola `success` otrzymamy pole `error` z opisem błędu i `errorCode` z kodem błędu.

ZLECANIE WYDRUKU PARAGONU

Aby zlecić wydrukowanie paragonu należy wywołać adres:

https://api.bsxprinter.pl/insert?key=KLUCZ

i metodą GET lub POST przekazać jeden parametr **data**. Umieszczamy w nim plik XML z paragonem (fakturą, komendami, e-Fakturą KSeF itp.), który chcemy wydrukować (wykonać lub przetworzyć).
W odpowiedzi na to wywołanie otrzymamy w polu `id` nadany danemu dokumentowi numer ID.

To wszystko! bsxPrinter jak będzie uruchomiony, w ciągu paru sekund powinien wydrukować zlecony dokument lub go przetworzyć w inny sposób, w zależności od sytuacji.

UWAGA! *Plik XML podajemy w formacie jak opisano w poprzednim podrozdziale, przy opisie metody Komunikacja poprzez XML. Ważne jest jednak, aby w takim pliku umieszczać tylko jeden dokument (jedną sekcję <receipt>…</receipt>! Dotyczy to*

oczywiście sytuacji gdy zlecamy wydruk paragonu. W przypadku e-Faktur przekazujemy po prostu XML e-Faktury.

SPRAWDZENIE STATUSU WYDRUKOWANIA PARAGONU

Chcąc sprawdzić status wydrukowanego dokumentu należy wywołać adres:

https://api.bsxprinter.pl/check?key=KLUCZ&id=ID

W parametrach podajemy klucz danej instancji bsxPrinter oraz ID dokumentu, jaki otrzymaliśmy przy jego zlecaniu. W odpowiedzi otrzymamy pola: `id` - id danego wpisu, `ptype` - typ wpisu oraz zestaw dodatkowych pól, zależnych od tego z jakim dokumentem mamy do czynienia.

Dla `ptype` = `RECEIPT` (paragon, faktura itp.) otrzymamy pola:

- `status` - status dokumentu
- `nodoc` - numer dokumentu
- `nida` - numer dobowy
- `nidb` - numer ogólny
- `ndate` - data wydruku
- `nerrorcode` - kod błędu
- `nerrorstr` - opis błędu
- `nres` - dodatkowe odpowiedzi
- `d_termstatus` - status transakcji na terminalu
- `d_tdate` - data wykonania transakcji na terminalu
- `d_trnumber` - nazwa karty
- `d_maskednbr` - zamaskowany numer karty

Analizując otrzymane informacje może zatem się dowiedzieć kiedy nastąpił wydruk, jaki został nadany mu numer, ewentualnie czy wystąpiły jakieś błędy.

Oto możliwe statusy:

- **0** - oczekuje na przetworzenie
- **1** - w trakcie przetwarzania
- **2** - przetworzony
- **-5** - wystąpił błąd - będzie ponowiona próba wydruku
- **-6** - wystąpił błąd - nie będzie ponowionej próby wydruku

Kody błędów są identyczne jak przy komunikacji XML.

Dla `ptype` = `FA` (e-Faktura) otrzymamy pola:

- `status` - status dokumentu
- `nodoc` - numer dokumentu
- `nida` - numer KSeF
- `nidb` - numer referencyjny e-Faktury
- `upo` - UPO
- `ndate` - data przyjęcia do KSeF
- `nerrorcode` - kod błędu
- `nerrorstr` - opis błędu

Statusy mają następujące znaczenie:

- **0** - oczekuje na wysłanie do bsxPrinter
- **1** - odebrany przez bsxPrinter
- **2** - wysłano do KSeF
- **-3** - Odrzucony przez KSeF
- **4** - Trwa weryfikacja po stronie KSeF
- **5** - Potwierdzony przez KSeF
- **6** - Dostępne jest UPO

Przykład:

https://api.bsxprinter.pl/insert?
key=b83b3de7b4705790bd4d42e7e7d2701d4dc03032&data=<root><receipts><rece
ipt><item name="Produkt" price="1.00" /></receipt></receipts></root>

W odpowiedzi otrzymamy liczbę np.

{"success":true,"id":"663037"}

https://apibsxprinter.pl/check?
key=b83b3de7b4705790bd4d42e7e7d2701d4dc03032&id=663037

w odpowiedzi otrzymamy:

{"success":true,"data":
{"id":"663038","type":"RECEIPT","status":"1","nodoc":"33\/2023-08-08\/
23","nida":"33","nidb":"23","ndate":"2023-08-08
10:29:38","nerrorcode":"0","nerrorstr":null,"nres":null,"d_termstatus":
"0","d_tdate":null,"d_trnumber":null,"d_maskednbr":null}}

Paragon został zatem wydrukowany 8 sierpnia 2023 r. o 10:29 i otrzymał numer 33/2023-08-08/23.

WŁASNE SERWERY DLA BINSOFT GATE

Domyślnie usługa BinSoft Gate korzysta z serwerów firmy BinSoft. Jednakże istnieje możliwość współpracy z partnerami, by ich indywidualne wersje programu bsxPrinter korzystały z ich serwerów na potrzeby tej usługi. Dzięki temu wszystkie informacje przekazane do bsxPrinter nie będą trafiały na serwery BinSoft lecz na serwery Partnerów.

Aby skorzystać z powyższej możliwości należy skontaktować się z nami w celu ustalenia szczegółów.

BinSoft udostępni kod źródłowy w języku PHP skryptu obsługującego BinSoft Gate i pomoże w przygotowaniu takiego rozwiązania.

KLIENT WEBSOCKET

Od wersji 22.8.10 bsxPrinter umożliwia nawiązanie połączenia z dowolnym serwerem WebSocket. Wystarczy w jego konfiguracji wprowadzić odpowiedni adres URL. W adresie tym można zdefiniować port, na który ma być nawiązane połączenie po znaku dwukropka. Adres można poprzedzić również identyfikatorem protokołu `ws://` lub `wss://` (dla połączeń szyfrowanych), np.: `wss://www.adres-serwera-wss.pl/api:7654`

Jeśli w ustawieniach określimy nazwę użytkownika i/lub hasło, wówczas bsxPrinter po nawiązaniu połączenia z serwerem automatycznie wyśle do niego linię:

```
#LOGIN#użytkownik#hasło
```

Przykładowy scenariusz:

Partner uruchamia własny serwer WebSocket i informuje swoich użytkowników, by w programie bsxPrinter wprowadzili stosowny adres i podali swoje nazwy użytkowników oraz hasła. Następnie po stronie serwera implementuje obsługę komendy `#LOGIN` by weryfikować poprawność tych danych.

Z poziomu serwera można następnie wysyłać do klienta dowolne komendy opisane w tym podręczniku - jak dla plików `.in` czy przy obsłudze protokołu TCP/IP.

KOMENDY BSXPRINTER

System bsxPrinter zawiera szereg komend, które możne wykonać - bądź to za pośrednictwem połączenia TCP/IP lub poprzez plik tekstowy umieszczony we współdzielonym katalogu (lub na serwerze FTP). Wszystkie komendy mogą być poprzedzone znakiem #. Nie jest to konieczne i występuje tylko dla zachowania kompatybilności ze starszymi wersjami programu. Jeśli zawierają jakieś dodatkowe parametry, te powinny być przekazywane po znaku #. Oto kilka przykładów:

- `#KOMENDA` - bez parametru
- `#KOMENDA#Parametr1` - jeden parametr
- `#KOMENDA#Parametr1#Parametr2` - dwa parametry

W odpowiedzi na każdą komendę bsxPrinter odpowie ciągiem znaków. Będzie on rozpoczynał się od:

- `+OK[#...]` - jeśli komenda została wykonana poprawnie; po znaku `#` mogą być dodatkowe informacje związane z wykonaną komendą;
- `+ERR[#KOD BŁĘDU]#Opis błędu` - jeśli wystąpi błąd;

UWAGA! *Umieszone informacje w nawiasach kwadratowych oznaczają, że dane informacje (np. parametry) występują opcjonalnie.*

UWAGA! *Kolejne parametry oddzielone są znakiem #. Oznacza to, że chcąc podać kolejny parametr (np. 3), należy podać wszystkie poprzednie. Nie możemy również użyć symbolu # jako wartość jakiegoś parametru. Nie możemy więc w nazwie produktu wyświetlić tego symbolu.*

DOSTĘPNE KOMENDY PODSTAWOWE

- `#HELLO` - bsxPrinter przedstawi się swoją nazwą oraz numerem wersji;
- `#DATE` - zwraca datę na serwerze bsxPrinter;
- `#TIME` - zwraca czas na serwerze bsxPrinter;
- `#DATETIME` - zwraca datę i czas na serwerze bsxPrinter;
- `#PROTOCOLS` - zwraca listę obsługiwanych protokołów;
- `#PASSWORD#hasło` - autoryzacja użytkownika;
- `#CLOSE` - wyłączenie programu bsxPrinter;
- `#CONNECTIONS` - zwraca listę wszystkich podłączonych do serwera połączeń;
- `#SERVERS` - zwraca status wszystkich serwerów i monitorów;
- `#DATABASE[#Value]` - zwraca informacje o aktualnej bazie danych, z której korzysta aplikacja. Jeśli przekazano wartość `Value` - ustawia nową bazę danych; Jako `Value` można wpisać: `default` - by użyta była domyślna baza danych lub ciąg reprezentujący dostęp do bazy danych np.: `MySQL#localhost#3306#root#bsx#bsxprinter` lub `MSSQL#192.168.188.25\SQLEXPRESS#1433#sa#sa#bsxprinter`
- `#LICENSE[#Name#E-mail#Phone#Key]` - zwraca informacje o aktualnej licencji lub pozwala na zarejestrowanie programu;
- `#LOGMONITOR#Value[#Name]` - ustawienie monitora logów; `Value` przyjmuje wartość 1 / 0 (włączony/wyłączony). Parametr `Name` gdy przyjmie wartość `low` oznacza logi niskopoziomowe, w przeciwnym wypadku logi standardowe;
- `#GETACTDOCUMENTS` - licznik aktualnie wydrukowanych dokumentów;
- `#GETLIMITDOCUMENTS` - pobranie limitu dokumentów;
- `#GETAVAILABLEDOCUMENTS` - pobranie liczby możliwych do wydrukowania jeszcze dokumentów;

OBSŁUGA KONFIGURACJI

- `#CONFIGFILE` - zwraca ścieżkę do pliku konfiguracyjnego;
- `#RELOADCONFIG` - przeładowuje plik konfiguracyjny;
- `#CONFIG` - zwraca zawartość pliku konfiguracyjnego;
- `#GET#Section#Name` - zwraca wartość z pliku konfiguracyjnego;
- `#SET#Section#Name#Value` - modyfikuje wpis w pliku konfiguracyjnym;

OBSŁUGA PORTU SZEREGOWEGO I KONFIGURACJI PROGRAMU

- `#COMLIST` - lista dostępnych portów szeregowych;
- `#PRINTERS` - lista dostępnych drukarek fiskalnych;
- `#PRINTER` - aktywna drukarka fiskalna;
- `#GPRINTERS` - lista dostępnych drukarek klasycznych;
- `#GPRINTER` - aktywna drukarka klasyczna;
- `#CONNECT` - połączenie z drukarką fiskalną;
- `#CONNECTED` - sprawdza czy bsxPrinter jest połączony z drukarką fiskalną;
- `#DISCONNECT` - rozłączenie z drukarką fiskalną;
- `#SETPRINTER#name|port#protocol` - zainicjowanie drukarki o danej nazwie (gdy podano jej nazwę w parametrze name) lub domyślnej drukarki na zadanym porcie komunikacyjnym i protokole;
- `#RECONNECTPRINTER` - rozłączenie i ponowne połączenie z drukarką;
- `#PRINTPDF#data` - wydrukowanie pliku PDF dla drukarce klasycznej;
  W polu data podajemy binarną postać pliku zakodowaną w Base64;
- `#PRINTHTML#width#height#rotate#data` - wydrukowanie dokumentu HTML na drukarce klasycznej; W polu data podajemy wprost kod HTML; `Width` i `Height`

określają szerokość i wysokość papieru (w mm); Parametr `rotate` (1/0) określa czy strona ma być obrócona;

OBSŁUGA DRUKARKI FISKALNEJ

- `#GETVERSION` - pobranie nazwy i wersji drukarki;
- `#MAXLENPRODUCTNAME[#length]` - pobranie informacji o maksymalnej długości nazwy produktu lub ustawienie tego parametru;
- `#CHARSET[#NAME]` - pobranie informacji o kodowaniu znaku lub ustawienie kodowania; Możliwe wartości: UTF8, MAZOVIA;
- `#GETVERSION` - pobranie informacji o nazwie i wersji drukarki;
- `#GETPROTOCOL` - pobranie informacji o aktywnym protokole;
- `#GETCOM` - pobranie informacji o aktywnym porcie COM;
- `#GETDATETIME` - pobranie daty i czasu z drukarki fiskalnej;
- `#GETLASTDOCID` - pobranie numer ostatnio wydrukowanego paragonu;
- `#GETLASTDOC` - pobranie identyfikator ostatnio wydrukowanego paragonu;
- `#GETVATRATES` - pobranie informacji o zaprogramowanych stawkach VAT;
- `#SUPPORT#funkcja` - sprawdzenie czy dana drukarka obsługuje określoną funkcjonalność;
- `#DRAWER` - otwarcie szuflady;
- `#BEEP` - wygenerowanie sygnału dźwiękowego;
- `#PAPERFEED` - wysunięcie papieru;
- `#CASHIN#price` - kasa przyjmie;
- `#CASHOUT#price` - kasa wyda;
- `#CASHBACK#price` - zwrot towaru;
- `#CASH` - stan kasy;

- `#TESTPRINTER` - sprawdzenie, czy można wydrukować dokument tj. czy nie przekroczono limitu paragonów lub czy drukarka nie jest w jakiś sposób zablokowana;
- `#TRANSACTIONLOCK` - zablokowanie możliwości rozpoczęcia transakcji;
- `#TRANSACTIONUNLOCK` - odblokowanie możliwości rozpoczynania transakcji;
- `#RECEIPT[#barcode][#nip][#email]` - rozpoczęcie paragonu; można również podać kod kreskowy jaki ma być wydrukowany jeśli drukarka obsługuje tę funkcję oraz numer NIP jeśli ma być wydrukowany przy paragonie; Podając parametr `email` - w przypadku włączonej obsługi e-Dokumentów sprawimy, że zamiast wydruku dokument zostanie wysłany na dany adres;
- `#INVOICE#nodoc#nip#buyerName#paymentTerm#paymentForm[#buyer#seller][#email]` - rozpoczęcie faktury; podajemy numer faktury, dane kupującego, termin płatności, formę płatności, imię i nazwisko kupującego, i sprzedającego; Podając opcjonalny parametr `email` i mając włączoną obsługę e-Dokumentów sprawimy, że zamiast wydruku dokument zostanie wysłany na dany adres;
- `#INTRANSACTION` - sprawdzenie, czy drukarka jest w trakcie transakcji (w trakcie wydruku paragonu lub faktury);
- `#ROLLBACK` - anulowanie paragonu/faktury;
- `#ITEM#name#price#quantity#vat#total[#unit#pkwiu#discountType#discountName#discountProc#discountValue]` - linia paragonu/faktury; Określamy nazwę produktu (`name`), jego cenę (`price`), ilość (`quantity`), stawkę VAT (`vat`), wartość (`total`), jednostkę (`unit`), kod PKWiU (`pkwiu`); Możemy także na pozycję nadać rabat (`discountType=1`) lub narzut (`discountType=2`). W parametrze `discountProc` podajemy wartość rabatu/narzutu w procentach, a w `discountValue` - jako wartość bezwzględną. Należy podać tylko jedną z wartości `discountValue` albo `discountProc`. Jeśli nie chcemy określać rabatów nie ma potrzeby przekazywania w ogóle tych parametrów. Nadając

rabaty i podając kwotę `total` - należy pamiętać że powinna ona owe rabaty uwzględniać. Stawkę VAT podajemy jako wartość, np. 23, 7 lub jako literę: A, B, C itp.

- `#COMMIT[#total#cash#visa#end#printerCode#other#othername#discountName#discountType#discountValue]` - zatwierdzenie paragonu/faktury; Parametry są opcjonalne i określają odpowiednio: `total` - wartość dokumentu, `cash` - kwotę wpłaty gotówką, `visa` - kwotę wpłaty kartą płatniczą, `other` - kwotę wpłaty inną metodą płatności, `othername` - nazwę innej formy płatności; `discountName` - opis rabatu; `discountType` - czy nadano rabat procentowo (=0) czy kwotowo (=1), `discountValue` - wartość rabatu; `end` - czy to koniec transakcji (1/0); Jeśli chcemy przekazać dodatkowe linie do wydruku nie należy kończyć transakcji, lecz podać dodatkowe linie komendą `LINE` i transakcje zakończyć komendą `ENDLINES`; Jeśli nie planujemy przekazywać dodatkowych linii - zakańczamy transakcję komendą `COMMIT` z parametrem `end=1`;
- `#LINE#nr#text` - dodatkowa linia na dokumencie;
- `#ENDLINES` - koniec dodatkowych linii na dokumencie;
- `#REPORTCASH` - raport stanu kasy w drukarce;
- `#REPORTMONTHLY` - raport miesięczny;
- `#REPORTDAY` - raport dobowy;
- `#FORMSTART[#FORMATKA]` - dowolny wydruk niefiskalny; jako parametr podajemy numer formatki obsługiwany przez daną drukarkę fiskalną; brak numeru spowoduje wybranie formatki uniwersalnej;
- `#FORMLINE#tekst[#FORMATKA]` - linia na formatce;
- `#FORMEND[#FORMATKA]` - zakończenie formatki;

- `#XMLDATA#<xml>` zlecenie przetworzenia danych zapisanych w XML; w ten sposób można przekazać do bsxPrinter cały paragon opisany w XML (jak opisano w innej części podręcznika). Należy pamiętać by w kodzie XML nie używać znaku przejścia do nowej linii (#13, #10), gdyż te znaki kończą komendę. Jeśli bardzo nam na nich zależy, należy je zakodować jako ciągi: ^13^ i ^10^. Przekazując dane w XML bsxPrinter zapisuje je do wewnętrznej bazy danych tzw. *bazy plików*, a następnie przetwarza ten wpis w bazie (*plik*). W efekcie przetworzenia w drugiej bazie (bazie *dokumentów*) tworzy zlecenie wydruku odpowiednich paragonów/faktur itp. Dzieje się tak z tego względu, że w jednym pliku XML może znajdować się wiele paragonów i te przetwarzane są niezależnie i asynchronicznie.

  W odpowiedzi na poprawne wykonanie tej komendy zwrócone zostaną dane w postaci: `+OK#ID_PLIKU#ID_DOKUMENTU1;ID_DOKUMENTU2….`
  Zatem zwrócony zostanie ID pliku, który utworzył się w bazie *plików* i numery ID paragonów utworzonych w bazie *dokumentów*, oddzielone średnikami.
- `#FILES` - zwraca listę plików z wewnętrznej bazy plików; Komenda zwraca maks. 100 ostatnich wpisów. Struktura zwracanych danych jest następująca: `+OK#ID_PLIKU|STATUS|KOD_BŁĘDU;.…`
- `#FILE#id` - zwraca opis pliku w bazie *plików*, o podanym numerze ID. Struktura zwracanych wartości: `+OK#ID_PLIKU|STATUS|KOD_BŁĘDU|OPIS_BŁĘDU`.
- `#DOCUMENTS` - zwraca listę dokumentów w wewnętrznej bazie dokumentów; Komenda zwraca maks. 100 ostatnich dokumentów. Struktura zwracanych danych jest następująca: `+OK#ID_DOKUMENTU|STATUS|KOD_BŁĘDU|DATA|TYP|ID_TRANSAKCJI_PŁATNICZEJ;…`

- `#DOCUMENT#id` - zwraca opis dokumentu w bazie *dokumentów*, o podanym numerze ID. Struktura zwracanych wartości: +`OK#ID_DOKUMENTU|STATUS|KOD_BŁĘDU|OPIS_BŁĘDU|DATA|TYP`.
-

TERMINALE PŁATNICZE

- `#TERMINAL_FORM[#PRICE#AUTOCLOSE#PRINTCONFIRM#TRANSACTIONTYPE]`

  - wyświetlenie formularza płatności; Można w wywołaniu przekazać kwotę wpłaty (`#PRIC`E), wówczas od razu rozpocznie się procedura płatności; Jeśli nie podamy kwoty, wyświetli się jedynie okno, a to użytkownik powinien wprowadzić kwotę i rozpocząć procedurę; Jeśli przekażemy parametr `AUTOCLOSE=1` - wówczas po zakończeniu transakcji formularz obsługi płatności zostanie automatycznie zamknięty; Przekazanie parametru `PRINTCONFIRM=1` spowoduje, że automatycznie zostanie wydrukowane potwierdzenie transakcji dla klienta; Ostatni z parametrów `TRANSACTIONTYPE` określa rodzaj transakcji.

  Dostępne wartości:

  0 - transakcja standardowa,

  1 - anulowanie ostatniej transakcji,

  2 - zwrot środków;

- `TERMINAL_FORM_STATUS[#WAIT]` - zwraca status realizacji transakcji z formularza płatności; Jeśli przekazano parametr `#WAIT=1` - wówczas funkcja zwróci wartość dopiero jak okno z terminala zostanie zamknięte; Jeśli nie przekażemy tego pola - funkcja zwróci status „aktualny” (na daną chwilę);

  Możliwe wartości:

  -5 - okno terminala zostało zamknięte ręcznie;

  -6 - okno nie było jeszcze wyświetlone;

-1 - okno terminala jest otwarte;

-2 - trwa przetwarzanie;

>=0 - transakcja zakończona - status zwrócony przez terminal;

Wartości zwracane przez terminal:

=0 - Transakcja zaakceptowana,

=1 - Transakcja odrzucona,

=2 - Brak połączenia,

=7 - Przerwano przez użytkownika;

- `TERMINAL_STATUS` - zwraca „aktualny” status komunikacji z terminalem oraz informacje o ostatnim błędzie; Format informacji zwracanych: `STATUS_CODE#STATUS_DESC#STATUS_EXT#LOG_CODE#LOG_STR`.

  Możliwe wartości dla `STATUS_CODE`:

  0 - Gotowy do transakcji,

  1 - Oczekiwanie na kartę,

  2 - Oczekiwanie na PIN,

  3 - Oczekiwanie na wybór EMV,

  4 - Komunikacja z serwerem,

  5 - Oczekiwanie na podpis,

  6 - Oczekiwanie na zakończenie,

  7 - Oczekiwanie na wyjęcie karty,

  8 - Terminal zajęty,

  9 - W trakcie transakcji,

  10 - Oczekiwanie na kopię,

  11 - Oczekiwanie na wprowadzenie kodu,

  12 - Oczekiwanie na użytkownika,

  13 - Oczekiwanie na pobranie danych,

14 - Oczekiwanie na wybór waluty; LOG_STATUS i LOG_STR - przekazują informacje z formularza transakcji;

Gdy LOG_STATUS=0 - operacja poprawna, gdy LOG_STATUS=1 - wystąpił błąd;

- TERMINAL_GETCOM - zwraca listę portów COM;
- CARDTRANSACTIONS[#ID] - zwraca listę (rekord o zadanym ID) transakcji płatniczych z wewnętrznej bazy danych;
- LASTCARDTRANSACTION - zwraca opis ostatniej transakcji płatniczej;

Ostatnie dwie komendy (CARDTRANSACTIONS[#ID] i LASTCARDTRANSACTION) zwracają linie opisującą wybraną transakcję. Poszczególne pola oddzielone są znakiem pionowej linii. Kolejne pola mają znaczenie:

```
ID|STATUS|DATE|BRAND_CARD|TRANSACTION_DOC_NUMBER|TRANSACTION_NUMBER|
TRANSACTION_CODE|TRANSACTION_AID|TRANSACTION_MID|TRANSACTION_TC|
TRANSACTION_SALE|MASKED_CARD_NUMBER|TRANSACTION_TYPE
```

Opis pól:

- TRANSACTION_TYPE:

  =1 - transakcja standardowa;

  =6 - zwrot środków;

  =10 - Anulowanie ostatniej transakcji;

- STATUS:

  =0 - Transakcja zaakceptowana,

  =1 - Transakcja odrzucona,

  =2 - Brak połączenia,

  =7 - Przerwano przez użytkownika;

SQL

- `SELECT, INSERT, UPDATE, DELETE, ALTER` - komendy SQL odnoszące się do wewnętrznej bazy danych; Pozwalają na wydobywanie dowolnych informacji z tej bazy, jak również dokonywanie zmian w bazie;
- `SHOW TABLES` - wyświetla listę wbudowanych tabel;

INNE KOMENDY

- `#RUNLOOP` - uruchamia przetwarzanie kolejek dla wszystkich integracji (XML, FTP, lokalny folder itp.);
- `#RUNPOOL` - uruchamia przetworzenie wewnętrznej kolejki dokumentów do przetworzenia;
- `#RUNFISCALPOOL` - uruchomienie przetworzenia wewnętrznej kolejki plików do przetworzenia;
- `#RUNONTIMER` - uruchomienie we wszystkich rozszerzeniach i wtyczkach funkcji `onTimer();`

SCENARIUSZE UŻYCIA

Użycie pliku XML:

1) Generujemy dokument XML z paragonem do wydrukowania, np.: `<root><receipts><receipt id="1"><item name="produkt" price="1.00" vat="23" quantity="1"></item></receipt></receipts></root>` i wysyłamy komendą: `#XMLDATA#<root><receipts><receipt id="1"><item name="produkt"`

```
price="1.00" vat="23" quantity="1"></item></receipt></receipts></
root>[13][10]
```

2) W odpowiedzi uzyskujemy: `+OK#10#22` - oznacza to, że bsxPrinter utworzył *plik* o ID=10 i stworzył *dokument* o ID=22.
3) Wywołujemy komendę: `#DOCUMENT#22` - aby uzyskać status dokumentu.
   W odpowiedzi otrzymujemy: `+OK#22#2##2018-09-27#1`

Użycie komend / ręczne tworzenie paragonu:

1) Wysyłamy komendę `#RECEIPT` - w celu rozpoczęcia paragonu;
2) Wysyłamy komendy `#ITEM` - z kolejnymi pozycjami na paragonie;
3) Wysyłamy komendę `#COMMIT` - aby zakończyć paragon;
4) Wysyłamy komendę `#GETLASTDOCID` - aby uzyskać numer wydrukowanego paragonu;

Przykłady:

Paragon, z wydrukowanym kodem kreskowym: 123,

produkt w cenie 10 zł, sztuk 2, z rabatem 2 zł

```
#ROLLBACK
#RECEIPT#123
#ITEM#Produkt#10#2#23#20.00#m2##1#Rabacik#0#2
#COMMIT#18
#CLEARRESULT
#GETLASTDOCID
#EXECUTE
```

Paragon z jedną pozycją, wydrukowanym numerem transakcji 123,

dodatkowym komentarzem „*Komentarz*”. Zapłacono 6 zł w gotówce i 10 zł kartą płatniczą.

```
#ROLLBACK
#RECEIPT
#ITEM#Produkt#2#4#23#8.00
#COMMIT#8#6#10#0
#LINE#0#123
#LINE#25#Komentarz
#ENDLINES
#CLEARRESULT
#GETLASTDOCID
#EXECUTE
```

Ręczne wydrukowanie dokumentu na klasycznej drukarce:

```
#PRINTHTML#62#62#0#<center>Dokument <b>HTML</b></center>
```

WARUNKOWANIE PROTOKOŁÓW

Od wersji 20.1.10 bsxPrinter obsługuje mechanizm „warunkujący” wykonanie danej komendy, w zależności od używanego protokołu. Aby skorzystać z tego mechanizmu wystarczy poprzedzić nazwę komendy - nazwą protokołu umieszczoną w nawiasach klamrowych.

Oto przykład:

```
#FORMSTART
#{Posnet}FORMLINE#Linia dla Posnet
#{Thermal}FORMLINE#Linia dla Thermal
#FORMEND
```

Powyższy kod spowoduje wydrukowanie „pokwitowania”. Jeśli będzie zlecony na drukarce z aktywnym protokołem Posnet - wydrukowana będzie linia „*Linia dla Posnet*”. W przypadku protokołu Thermal będzie drukowana linia „*Linia dla Thermal*”.

BEZPOŚREDNIE WYSYŁANIE KOMEND DO DRUKARKI FISKALNEJ

Od wersji 20.1.10 bsxPrinter pozwala na wysyłanie komend niskopoziomowych, bezpośrednio do drukarki fiskalnej. Umożliwia to uzyskanie wszystkich funkcjonalności danej drukarki, również tych nieobsługiwanych wprost poprzez odpowiednie komendy bsxPrinter. W celu wysłania danej komendy niskopoziomowej bezpośrednio do drukarki, należy użyć symbolu „<" lub „<<" (o różnicach powiemy dalej). Podczas tworzenia takich komend możemy używać specjalnych znaczników dla wstawienia pewnych symboli opisanych w dokumentacjach danych protokołów.
Dostępne są symbole:

- [TAB] - znak tabulacji ($09);
- [STX] - kod $02;
- [ETX] - kod $03;
- [LF] - kod $0A;
- [CR] - kod $0D;
- [ENQ] - kod $05;
- [BEL] - kod $07;
- [CAN] - kod $18;
- [DLE] - kod $10;
- [ESC] - kod $1B;

Spójrzmy na przykład:

```
#<formstart[TAB]fn200[TAB]fh11
#<formline[TAB]s1Komunikat[TAB]fn200[TAB]fl206
#<formend[TAB]fn200
```

Powyższy przykład spowoduje wydrukowanie „pokwitowania" z jedną linijką „*Komunikat*". Zastosowano tu komendy niskopoziomowe wysyłane bezpośrednio do drukarki fiskalnej.

Należy zwrócić uwagę, że protokoły Posnet i Thermal przewidują wysyłanie przed każdą komendą, specjalnej sekwencji startowej, a na końcu komendy, sekwencji końcowej. Tym zajmuje się bsxPrinter i nie podajemy tych informacji w komendach.

Większość komend wysyłanych do drukarek fiskalnych wymaga również przesłania specjalnej sumy kontrolnej na ich końcu. Tym również zajmuje się bsxPrinter, tj. automatycznie wylicza sumy kontrolne dla podanych komend. Jeśli chcemy wysłać komendę bez dołączonej sumy kontrolnej, zamiast symbolu „<" powinniśmy wówczas użyć „<<".

Oczywiście opisane wyżej możliwości możemy także stosować do komend wysyłanych z poziomu XML. Spójrzmy na ciekawy przykład:

```
<root>
<receipts>
  <receipt>
   <command name="FormStart"/>
      <command name="{posnet}<formline[TAB]s1Posnet-
TEST[TAB]fn200[TAB]fl206"/>
   <command name="{thermal}<200;206$wThermal-TEST"/>
   <command name="FormEnd"/>
  </receipt>
</receipts>
</root>
```

Zastosowano tu znacznik `<command>` poprzez który przekazujemy do bsxPrinter komendy. Rozpoczęcie formatki i zakończenie wysyłamy „zwyczajnie" poprzez komendy `FormStart` i `FormEnd`. Natomiast linię formatki wysyłamy bezpośrednio poprzez wysłanie komendy niskopoziomej w danym protokole. Stworzyliśmy tutaj jednak dwie wersje - jedną dla protokołu Posnet i drugą dla protokołu Thermal.

BSXPRINTER JAKO USŁUGA WINDOWS

Istnieje możliwość zainstalowania aplikacji bsxPrinter jako usługi Windows**(*)**. Aby tego dokonać należy uruchomić wiersz poleceń Windows (koniecznie z uprawnieniami Administratora), a następnie przejść do folderu, gdzie został zainstalowany program (domyślnie `c:\Program Files (x86)\bsxPrinter`).

Należy wydać polecenie:

```
bsxServer.exe /install
```

Można też uruchomić instalację „po ciuchu" - wówczas:

```
bsxServer.exe /install /silent
```

Po zainstalowaniu usługi możemy ją uruchomić, wstrzymać, zatrzymać - korzystając z narzędzia „Usługi" w systemie Windows.

Chcąc odinstalować usługę wydajemy z wiersza poleceń komendę:

```
bsxServer.exe /uninstall
```

lub „po cichu"

```
bsxServer.exe /uninstall /silent
```

W Usługach Windows program bsxPrinter będzie widoczny jako „`bsxServer`".

***) Uwaga!** Aby korzystać z bsxPrinter w postaci usługi Windows może być konieczne posiadanie odpowiedniej licencji.

OBSŁUGA PROGRAMU

Usługi nie posiadają interfejsu, dlatego nie można ich w prosty sposób konfigurować i obsługiwać. bsxPrinter zawsze uruchamia serwer lokalny TCP/IP działający domyślnie na porcie `7361`, oraz ma skonfigurowane hasło domyślne `BinSoftBSX`. Możemy się zatem w nim połączyć poprzez terminal (np. z użyciem programu Putty) i w ten sposób nim sterować.

PLIKI KONFIGURACYJNE

bsxPrinter przechowuje swoje pliki konfiguracyjne w katalogu: `c:\Users\Nazwa_użytkownika\AppData\Roaming\bsxPrinter\data`. Jednak

uruchomiony jako usługa korzysta z folderu, w którym jest zainstalowany, tj. w folderze `c:\Program Files (x86)\bsxPrinter` stworzy folder `data` i tam zacznie tworzyć swoje pliki.

W katalogu tym odnajdziemy plik `config.ini`, w którym zapisana jest cała konfiguracja aplikacji. Możemy w ten sposób ustawiać konfigurację do aplikacji.

Warto tutaj podpowiedzieć, że jeśli uruchomimy bsxPrinter z prawami Administratora, ona również uruchomi się wskazując konfigurację, jak przy usłudze. Zatem istnieje możliwość, abyśmy uruchomili bsxPrinter z prawami administratora, a następnie ją skonfigurowali według uznania i wyłączyli. Wówczas zapisze się konfiguracja i już aplikacji w wersji usługowej będzie z niej korzystała.

Uwaga! Uruchamiając bsxPrinter może ona ustawić opcję automatycznego startu przy uruchamianiu Windows. Warto tę opcję wyłączyć, jeśli planujemy korzystać z wersji usługowej, tak by w tym samym czasie nie działały zarówno bsxServer jak i bsxPrinter.

TWORZENIE WTYCZEK DO BSXPRINTER

bsxPrinter pozwala na tworzenie tzw. wtyczek (ang. *plugins*). Wtyczki mogą rozszerzać funkcjonalność aplikacji, dodając do niej np. obsługę sklepów on-line, integrując się z systemami CRM, CMS itp. Tworzenie wtyczek polega na przygotowaniu odpowiednich plików XML oraz PAS.

BUDOWA WTYCZKI

Wtyczki do programu bsxPrinter to odpowiednio przygotowane pliki z rozszerzeniem `.xml` oraz `.pas`. Każda wtyczka to folder, w którym powinny znajdować się owe pliki. Folder taki powinien być umieszczony w katalogu `APP\plugins` - gdzie APP - to folder, gdzie zainstalowany jest program bsxPrinter. Na przykład:

- `APP\plugins\Wtyczka\plugin.xml`
- `APP\plugins\Wtyczka\plugin.pas`

bsxPrinter w momencie uruchamiania przeszukuje wszystkie podfoldery folderu plugins i analizuje wszystkie pliki XML, które tam odnajdzie.

Każdy plik XML powinien mieć następujący format:

```
<plugin name="NazwaFirmy.NazwaPluginu" caption="Tytuł pluginu">
    <!-- dodatkowe sekcje -->
</plugin>
```

Każdy plugin możne zawierać dowolnie wiele okien dialogowych (formularzy). Okna te definiuje się w znaczniku `<forms>`. Każde okno opisuje się znacznikiem `<form>`. Znacznik `<form>` posiada następujące atrybuty:

- `name` - nazwa danego okna,
- `priority` - priorytet okna,
- `inherited` - nazwa okna, po którym chcemy dziedziczyć,
- `width` - szerokość okna,
- `height` - wysokość okna,

- `caption` - tytuł okna.

W atrybucie name podajemy nazwę tworzonego formularza. bsxPrinter uzupełni tę nazwę o nazwę samego pluginu i rozdzieli symbolem kropki. Jeśli zatem plugin ma `name="NazwaFirmy.NazwaPluginu"`, a formularz ma `name="Okno"`, to pełna nazwa tego okna (którą posługujemy się w innych miejscach programu) to `NazwaFirmy.NazwaPluginu.Okno`. Jeśli nie chcemy, aby bsxPrinter uzupełniał nazwę okna nazwą pluginu, powinniśmy ją poprzedzić symbolem `@`, np. `@Okno`.

Wewnątrz znacznika `<form>` umieszcza się znaczniki tworzące różne elementy interfejsu. Dostępne znaczniki to:

- `<field>` - pole ukryte;
- `<edit>` - pole tekstowe,
- `<editbtn>` - pole tekstowe z widocznym przyciskiem,
- `<memo>` - wielowierszowe pole tekstowe,
- `<combobox>` - pole z listą wyboru,
- `<listbox>` - lista wyboru,
- `<trackbar>` - pasek przesuwania,
- `<radio>` - pole typu *radio*,
- `<checkbox>` - pole typu *checkbox*,
- `<progressbar>` - pasek postępu,
- `<button>` - przycisk,
- `<label>` - etykieta,
- `<html>` - etykieta obsługująca podstawowe znaczniki HTML,
- `<groupbox>` - zgrupowanie obiektów,
- `<panel>` - panel z obiektami,
- `<tabs>` - panel z zakładkami,
- `<tab>` - pojedyncza zakładka,

- i wiele innych.

Znaczniki `<groupbox>` i `<panel>` mogą w swoim wnętrzu zawierać kolejne elementy, tworząc w ten sposób hierarchę drzewa. Wewnątrz znacznika `<tabs>` (zakładki) może znaleźć się tylko znacznik `<tab>` (zakładka), a w nim dowolne inne obiekty.

Umieszczanie kolejnych znaczników opisujących interfejs powoduje, że elementy te umieszczane są jeden pod drugim. Korzystając z atrybutów: `width` i `height` - można zmieniać wymiary tych obiektów; Natomiast atrybutami `top` i `left` - można zmieniać ich domyślne położenie.

Jeśli jako wartość parametru `top` wpiszemy `-1` - wówczas obiekt znajdzie się na wysokości obiektu poprzedniego, a jego `left` ustawi się automatycznie tak by nowy element był „obok" poprzedniego. Dzięki temu możemy w prosty sposób umieszczać kilka elementów obok siebie.

Każdy z obiektów posiada dodatkowo atrybuty:

- `name` - określający nazwę danego obiektu,
- `caption` - tytuł obiektu,
- `default` - domyślna zawartość,
- `visible` - widoczność obiektu,
- `enabled` - status dostępności obiektu,
- `anchors` - uchwyty obiektu,
- `align` - sposób wyrównania obiektu;

Właściwości `anchors` (uchwyty) nadajemy wartość tekstową, która może zawierać wyrazy: `left`, `right`, `top`, `bottom`. Każda z nazw odpowiada stworzeniu uchwytu odpowiednio lewego, prawego, górnego i dolnego. Utworzenie odpowiedniego uchwytu powoduje, że dana kontrolka "utrzymuje" stale odległość do danej krawędzi obiektu, w którym się znajduje, podczas zmiany wymiarów tego obiektu.

Właściwość `align` wpływa na sposób wyrównania obiektu. Możliwe wartości to:

- `top` - wyrównanie do góry,
- `left` - wyrównanie do lewej,

- `right` - wyrównanie do prawej,
- `bottom` - wyrównanie do dołu,
- `client` - wypełnienie;

bsxPrinter uruchamiając się automatycznie ładuje formularz o nazwie `@Main`. Domyślnie wyświetli się formularz dostarczony wraz z programem. Jeśli jednak zrobimy samodzielnie formularz o tej nazwie i wyższym priorytecie, wówczas to on zostanie załadowany.

Oto przykład:

```
<plugin name="Firma.Plugin">
 <forms>
     <form name="@Main" caption="DrukSoft for v1.0" width="600" height="380"
priority="1">
      <button caption="Kliknij mnie" onclick="clickMe" />
      <script src="skrypt.pas" />
     </form>
  </forms>
</plugin>
```

W znaczniku `<script>`, w atrybucie `src` podajemy nazwę pliku z kodem w Pascalu, który ma być powiązany z danym formularzem. W naszym przykładzie jest to plik `skrypt.pas`. Może on mieć postać jak poniżej:

```
procedure clickMe;
begin
  ShowMessage('Witaj Świecie!');
end;
```

BUDOWA INTERFEJSU W JĘZYKU HTML

Budując formularz można użyć kontrolki `<html>`. Domyślnie wypełnia ona cały dostępny obszar, gdzie się znajduje (atrybut `align=client`). Wewnątrz tej kontrolki można używać znaczników języka HTML. Przykład:

```
<plugin name="Firma.Plugin">
 <forms>
     <form name="@Main" caption="DrukSoft for v1.0" width="600" height="380"
priority="1">
      <html>
       <style> b { color: red; } </style>
       <p>Akapit <b>tekstu</b></p>
       <button onclick="pascal:clickMe">Kliknij mnie</button>
      </html>
      <script src="skrypt.pas" />
     </form>
  </forms>
```

Powyższy przykład wyświetli jeden akapit tekstu (z czerwonym słowem tekstu) i przyciskiem. Kliknięcie w przycisk wywoła funkcje `clickMe` z Pascala. W przykładzie widzimy zatem, że by wywołać dowolną funkcję z podpiętego Pascala należy jej nazwę poprzedzić słowem: pascal:.

Znacznik `<html>` posiada atrybut `style`, któremu jako wartość możemy podać nazwę pliku ze stylami CSS. Posiada również wszystkie inne standardowe atrybuty, np.: `visible`, `align`, `width` itp.

ELEMENTY MENU

Z poziomu wtyczki można także dodawać nowe pozycje do menu głównego aplikacji. W tym celu należy umieścić wewnątrz formularza `@Main` znacznik `<menu>`.
Kolejne elementy (pozycje) menu opisuje się znacznikiem `<item>`. Znaczniki `<item>` można w sobie zagnieżdżać tworząc w ten sposób dowolnie zagłębione menu.
Oto przykład:

```
<menu>
  <item name="mnPlugins" caption="Wtyczki">
    <item caption="Konfiguracja" onclick="oknoKonfiguracjiClick" />
  </item>
</menu>
```

Ten kod XML spowoduje, że pokaże się nowa pozycja w menu programu zatytułowana *Wtyczki*. Wewnątrz niej będzie jedna pozycja o etykiecie *Konfiguracja*. Wybranie tej pozycji spowoduje uruchomienie procedury o nazwie `oknoKonfiguracjiClick` opisanej w powiązanym pliku .pas.

Jak zatem widać w przykładzie, wewnątrz `<item>` można używać atrybutów:

- `name` - nazwa elementu menu;
- `caption` - tytuł elementu menu;
- `onlick` - nazwa procedury, która ma zostać uruchomiona po wybraniu danej pozycji;
- `visible` - widoczność elementu;

Wszystkie elementy menu powinny posiadać nazwy. Jeśli użyjemy nazwy już istniejącej, wówczas nie stworzymy nowej pozycji, lecz "dołączymy" się do tej istniejącej.

Po uruchomieniu aplikacji bsxPrinter w menu już znajdują się pewne elementy.
Ich nazwy są następujące:

- `mnStart` - *Start*
- `mnDocuments` - *Dokumenty*
- `mnCash` - *Kasa*
- `mnReports` - *Raporty*
- `mnOthers` - *Inne*
- `mnHelp` - *Pomoc*

Jeśli zatem użyjemy którejś z powyższych nazw w znaczniku `<item>`, możemy wpiąć swoje elementy menu w odpowiednią gałąź. Z tego mechanizmu można też skorzystać aby np. ukryć określone elementy menu. Przykład krótkiego pluginu:

```
<plugin name="Firma.Plugin">
 <forms>
     <form name="@Main" priority="1" inherited="@Main">
      <menu>
```

```
        <item name="mnDocuments" visible="false" />
        <item name="mnCash" visible="false" caption="Kaska" />
      </menu>
    </form>
  </forms>
</plugin>
```

Powyższy plugin tworzy własne okno główne - jednak dziedziczy w nim wszystko po oknie oryginalnym. W efekcie pokaże się użytkownikowi domyślne okno główne. Dodano w nim definicję menu, która odwołuje się do dwóch istniejących elementów. Nadaje im atrybut `visible="false"` co spowoduje ukrycie tych elementów.

JĘZYK SKRYPTOWY - PASCAL

Za pomocą plików XML możliwe jest budowanie formularzy, określanie elementów menu, wpływanie na wygląd aplikacji. Aby do niej dodać logikę, należy utworzyć odpowiednie pliki skryptowe. Są to pliki zapisane z rozszerzeniem .pas i powiązane z danym formularzem za pomocą znacznika `<script>`. W plikach tych stosuje się język programowania bazujący na Pascalu. Nie jest to dokładna implementacja oryginalnego Pascala, lecz pewna jego odmiana. W dalszej części tej instrukcji język ten jednak będziemy nazywać po prostu "Pascalem".

ELEMENTY SKŁADOWE PASCALA BSXPRINTER

W języku Pascal każda instrukcja powinna być zakończona średnikiem. Skrypt rozpoczyna się słowem kluczowym `begin` i kończy słowem `end`;. Kiedy bsxPrinter wczytuje dany skrypt automatycznie uruchamia kod umieszczony w tym miejscu.
Oto przykład:

plugin.pas

```
begin
  ShowMessage('Witaj');
end;
```

Jeśli plik ten powiążemy z formularzem poprzez znacznik `<script>`, wówczas po wyświetleniu się tego formularza pokaże się komunikat *Witaj*.

W języku Pascal wielkość liter nie ma znaczenia. W przykładzie wywołano instrukcję `ShowMessage()`. Można ją było wywołać również poprzez nazwy `showmessage()`, czy `SHOWMESSAGE()`.

W Pascalu można tworzyć procedury oraz funkcje. Procedury i funkcje to podprogramy wykonujące określone czynności. Mogą one przyjmować argumenty, wówczas przekazujemy je w nawiasach okrągłych oddzielając od siebie przecinkami.
Funkcje - w przeciwieństwie do procedur - mogą też zwracać wartości.
Zwrócenie wartości polega na przypisaniu jej do wewnętrznej zmiennej `Result`. Przykład:

```
procedure nazwaProcedury(Argument1,Argument2);
begin
  //kod procedury
end;
function nazwaFunkcji(Argument1):ZwracanyTyp;
begin
  //kod funkcji
  Result:='Wynik';
end;
```

Jeśli funkcja/procedura nie oczekuje żadnych argumentów wówczas nie umieszczamy części w nawiasach okrągłych. Na przykład:

```
procedure pobierzDane;
begin
  //kod procedury
end;
```

Oczywiście w obrębie procedury/funkcji można deklarować zmienne lokalne. Deklaruje się je po słowie kluczowym `var`, przed słowem `begin`.

```
procedure pobierzDane;
var x,y,z : String;
    a,b,c : Integer;
begin
  //kod procedury
end;
```

W języku Pascal do przypisywania wartości służy operator przypisania, tj. `:=`. Sam znak "=" jest operatorem porównywania. Inne operatory porównywania to: <, >, <=, >=, <>. Oprócz tego dostępne są operatory logiczne: `OR`, `AND`, `NOT`.

Instrukcja warunkowa w Pascalu ma postać:

```
if warunek then instrukcja;
```

Jeśli chcemy dodać klauzulę `else`, wówczas instrukcja ta ma postać:

```
if warunek then instrukcja1 else instrukcja2;
```

Należy zwrócić uwagę, że po `instrukcja1` nie ma średnika.

Pascal udostępnia tzw. instrukcję złożoną. Instrukcja złożona to dowolny blok instrukcji objęty słowami kluczowymi `begin` i `end`. Można z niej skorzystać wszędzie tam, gdzie składnia przewiduje użycie pojedynczej instrukcji. Zatem chcąc wykonać więcej linii kodu w instrukcji warunkowej może ona wyglądać tak:

```
if warunek then
begin
  instrukcja1;
  instrukcja2;
end else
begin
  instrukcja3;
  instrukcja4;
end;
```

W Pascalu można korzystać ze zmiennych. Aby móc użyć zmienną, należy ją najpierw zadeklarować. Używa się w tym celu słowa kluczowego `var`. Zmienne deklaruje się na samym początku pliku - wówczas są to zmienne globalne - lub też przed słowem `begin` procedur i funkcji - wówczas są to zmienne lokalne. Przykład:

```
var zmiennaGlobalna;

procedure test;
var zmienaLokalna1, zmiennaLokaln2;
```

```
begin
  //instrukcje
end;
```

W Pascalu można tworzyć pętle. Najbardziej popularna jest pętla `for`. Jej składnia jest następująca:

```
for licznik:=start to koniec do instrukcja;
```

Oczywiście w miejscu instrukcji może znaleźć się instrukcja złożona. Przykład:

```
for i:=1 to 10 do
begin
   ShowMessage(i);
end;
```

Aby zrobić pętlę odliczającą w dół należy użyć składni:

```
for licznik:=start downto koniec do instrukcja;
```

Istnieją jeszcze dwie inne formy pętli: `while` oraz `repeat`. Składnia instrukcji while jest następująca:

```
while warunek do instrukcja;
```

Pętla wykonywana jest tak długo, dopóki warunek jest spełniony.

Składnia instrukcji `repeat` jest następująca:

```
repeat
  instrukcja1;
  instrukcja2;
  itd.
until warunek;
```

Pętla jest wykonywana tak długo, aż warunek będzie spełniony.

W Pascalu występuje typowanie danych. Podczas deklaracji zmiennych można podać nazwę typu, jednak jest to z reguły robione tylko dla zachowania czytelności kodu. W praktyce nie ma potrzeby tego czynienia, gdyż typ definiuje się poprzez przypisanie do zmiennej wartości. Oznacza to, że jeśli do zmiennej przypiszemy wartość liczbową - będzie ona miała typ liczbowy. Jeśli przypisze się wartość tekstową - będzie miała typ tekstowy. (Wartości tekstowe umieszczamy zawsze w apostrofach!).
Nie można tworzyć wyrażeń składających się ze zmiennych różnych typów. Niedozwolony jestem np. taki zapis:

```
x:='Jan ma '+20+' lat';
```

W Pascalu można tworzyć tablicę. Robimy to w następujący sposób:

```
var T : Array[0..10] of String;
```

Można również:

```
var T : array;
```

Korzystając z symboli `[]` można tworzyć tablice dynamicznie, np.:

```
T:=['Jan','Piotr'];
```

Uwaga! W związku z tym, że symbole `[]` służą do tworzenia tablic, nie można ich używać do tworzenia zbiorów - jak ma to miejsce w klasycznym Pascalu. W tym celu należy używać funkcji `SetOf(array)`, który z tablicy tworzy zbiór.

PRZYDATNE FUNKCJE PODSTAWOWE

Pascal oferuje szereg funkcji podstawowych, z których z pewnością będziemy korzystać bardzo często. Są to:

- `ShowMessage(napis);` - wyświetlenie napisu w oknie dialogowym;
- `IntToStr(liczba);` - zamiana liczby na napis;
- `StrToInt(napis);` - zamiana napisu na liczbę;
- `StrToIntDef(napis,def);` - zamiana napisu na liczbę; jeśli się nie powiedzie zwróci wartość domyślną def;

- `Copy(napis,od,ilość);` - skopiowanie fragmentu ciągu;
- `Delete(napis,od,ilość);` - wycięcie z ciągu znaków określonego fragmentu;
- `Pos(ciąg,napis);` - szukanie ciągu wewnątrz napisu;
- `LowerCase(napis);` - zamiana liter na małe;
- `UpperCase(napis);` - zamiana liter na wielkie;
- `Length(napis);` - zwraca długość napisu;
- `Now;` - zwraca aktualną datę i godzinę (w postaci zmiennej typu `TDateTime`);
- `DateToStr(data)` - zamienia datę na napis;
- `DateTimeToStr(data)` - zamienia datę i godzinę na napis;
- `StrToDate(napis)` - zamienia napis na datę;
- `StrToDateTime(napis)` - zamienia napis na datę i godzinę;
- `SetOf(tablica)` - zamienia tablicę na zbiór;

Oprócz funkcji podstawowych bsxPrinter oferuje szereg własnych funkcji. Są to:

- `APP.ShowForm(sender,nazwa,default,modal);` - wyświetlenie formularza o nazwie `nazwa`; Jako `Sender` przekazujemy obiekt, z którego następuje wywołanie. Można w to miejsce podać wartość `Self`. `Default` to lista domyślnych wartości w postaci: `nazwa=wartość;nazwa=wartość` itd. Jako `modal` podajemy wartość: `_BOOL_TRUE` - jeśli okno ma być modalne lub `_BOOL_FALSE` - w przeciwnym wypadku.
- `Home.GetValue(nazwa,domyślnie);` - pobranie wartości z obiektu `nazwa`. Jeśli obiekt nie zostanie odnaleziony zwrócona zostanie wartość domyślna;
- `Home.SetValue(nazwa,wartość);` - nadanie wartości dla wybranego obiektu;
- `Home.FindFieldControl(nazwa);` - pobranie bezpośredniego uchwytu do obiektu;

bsxPrinter udostępnia także szereg klas oferujących różne rozszerzone funkcjonalności. Są to:

- `TRegistry` - obsługa rejestru Windows;

- `TIniFiles` - obsługa plików INI;
- `TStringList` - obsługa danych tekstowych;
- `TBinDatabase` - obsługa baz danych;
- `TBSXRest` - obsługa HTTP i REST;
- `TBSXUtils` - różne przydatne funkcje;
- `TXMLDocument` - obsługa XML;
- `TJSONPair`, `TJSONObject`, `TJSONArray`, `TJSONString`, `TJSONValue` - obsługa JSON;

KILKA PRZYKŁADÓW

Poniżej przedstawionych zostało kilka przykładowych fragmentów programu, które demonstrują sposób użycia niektórych funkcji.

PRZYKŁAD 1

```
procedure pobierzDane;
var L : TStringList;
begin
  L:=TStringList.Create;
  L.Text:=TBSXRest.SGET('http://127.0.0.1/
                         script.php','login=Janusz;pass=Hasło');
  L.SaveToFile('c:\1.htm');
  L.Free;
  ShowMessage('Przetworzono');
end;


begin
  Self.CreateTimer('Timer',60000,'pobierzDane');
end;
```

W przykładzie zastosowaną metodę statyczną klasy `TBSXRest` do pobierania plików ze zdalnych serwerów. Metoda przyjmuje dwa parametry: adres URL oraz listę parametrów przekazywanych metodą POST. Funkcja zwraca plik odebrany z serwera.

W przykładzie pokazano również obsługę timerów. Funkcja `CreateTimer` tworzy timery. Parametry przekazane tej funkcji to: nazwa timera, częstotliwość (w milisekundach) i nazwa funkcji/procedury, która ma być wywoływana cyklicznie z zadaną częstotliwością.

WSKAZÓWKI

Klasa `TStringList` pozwala na obsługę danych tekstowych. Najciekawsze jej metody i właściwości w kontekście obiektu `L : TStringList`:

- `L.Count` - zwraca liczbę linii tekstu, które przechowuje obiekt;
- `L.Add('test')` - dodaje kolejną linię tekstu;
- `L.Delete(numer)` - usuwa linię o podanym numerze;
- `L.Text` - daje dostęp do pełnej zawartości przechowywanego tekstu;
- `L.SaveToFile(nazwaPliku)` - zapisuje zapamiętany tekst do pliku;
- `L.LoadFromFile(nazwaPliku)` - wczytuje zawartość pliku;
- `L.Clear` - czyści przechowywany tekst;

PRZYKŁAD 2

```
procedure plikiINI;
var Ini : TIniFile;
begin
 Ini:=TIniFile.Create('c:\plik.ini','',false);
 Ini.WriteString('Sekcja','Nazwa','Wartosc');
 Ini.Free;
 ShowMessage('Zapisano');
end;
```

WSKAZÓWKI

Inne przydatne metody i właściwości w kontekście obiektu `Ini : TIniFile`:

- `Ini.WriteString('Sekcja','Nazwa','Wartość')` - zapamiętanie ciągu `wartosc` dla pola `nazwa` w sekcji `sekcja`;
- `Ini.WriteInteger('Sekcja','Nazwa',Wartość)` - metoda analogiczna do poprzedniej, jednak wartość jest typu Integer (liczba całkowita);
- `Ini.ReadString('Sekcja','Nazwa')` - pobiera wartość pola `nazwa` z sekcji `sekcja`;
- `Ini.ReadInteger('Sekcja','Nazwa')` - metoda analogiczna do poprzedniej, jednak zwraca wartość typu Integer (liczba całkowita);

PRZYKŁAD 3

```
procedure rejestrWindows;
var Reg : TRegistry;
begin
  Reg:=TRegistry.Create(KEY_ALL_ACCESS);
  Reg.RootKey:=HKEY_CURRENT_USER;
  if Reg.OpenKey('Software\BinSoft\BSX Printer', TRUE) then
  begin
    ShowMessage(Reg.ReadString('Folder'));
  end;
  Reg.Free;
end;
```

PRZYKŁAD 4

```
procedure bazaDanych;
var B : TBinDatabase;
    T : TBinTable;
    L : TStringList;
begin
  try

B:=TBinDatabase.CreateRemote('mysql','localhost',3306,'root','karol','binsoft',
'','',22,'','');
```

```
    L:=TStringList.Create;
    try
      B.Tables(L);
      ShowMessage('Tabel: '+IntToStr(L.Count));
      if L.Count>0 then
      begin
        T:=B.GetRow('SELECT count(*) FROM '+L[0]);
        if Assigned(T) then
        begin
          ShowMessage('W tabeli '+L[0]+' jest '+T.FieldValueS['count(*)']+'
rekordow.');
          T.Free;
        end;
      end;
    finally
      B.Free;
      L.Free;
    end;
  except
      ShowMessage('Nie dalo sie nawiazac polaczenia z baza danych. Sprawdz, czy
dane podano prawidlwo. '+_NL+'Komunikat: '+LastExceptionMessage);
  end;
end;
```

PRZYKŁAD 5

```
procedure plikiXML;
var XML,F,E;
begin
  XML := TXMLDocument.Create;
  try
    XML.LoadFromFile('c:\plugin.xml');
    XML.Active:=TRUE;
    if Assigned(XML) then
    begin
```

```
      F:=XML.DocumentElement.FindNodeS('forms');
      if Assigned(F) then
      begin
       E:=F.FindNodeS('form');
       while Assigned(E) do
       begin
         ShowMessage(E.getAttr('caption','Domyslne'));
         E:=E.Next;
       end;
      end;
    end;
  finally
    XML.Free;
  end;
end;
```

ZDARZENIA

Użytkownik ma możliwość tworzenia własnych procedur i funkcji wewnątrz plików skryptowych .pas. Tworząc te podprogramy, programista nadaje im własne nazwy. Jednakże jeśli nazwie on swoją procedurę w odpowiedni sposób, system bsxPrinter będzie ją wywoływał automatycznie, w określonych sytuacjach. Na przykład, jeśli w pliku znajdzie się procedura o nazwie `fOnFormCreate()` - wywołana zostanie zawsze automatycznie, w momencie tworzenia danego formularza. Funkcja o nazwie `fOnFormClose()` wywołana będzie w momencie zamykania okna. Poniżej znajduje się lista wszystkich podstawowych zdarzeń:

- `fOnFormCreate()` - wywoływana w momencie tworzenia okna;
- `fOnFormClose()` - wywoływana w momencie zamykania okna;
- `fOnTimer()` - funkcja wywoływana zgodnie z częstotliwością ustawioną w konfiguracji programu;

OBSŁUGA BAZ DANYCH

Z poziomu kodu Pascala mamy dostęp do klasy `TBinDatabase`. Klasa ta zapewnia obsługę wielu rodzajów baz danych, m.in.: SQLite, MySQL, MS SQL, Firebird, PostgreSQL.

Oto dostępne konstruktory:

- `CreateSQLite(nazwaPliku, Haslo)` - tworzy lub otwiera bazę danych SQLite zapisaną w pliku `nazwaPliku`. Jeśli baza jest zaszyfrowana (przy otwieraniu) lub chcemy by była zaszyfrowana (przy tworzeniu) należy podać hasło. W przeciwnym wypadku jako hasło wpisujemy ciąg pusty.
- `CreateFirebird(nazwaPliku)` - otwiera bazę danych Firebird z podanego pliku;
- `CreateRemote(protokół,host,port,login,pass,nazwaBazy,tunelHost,sshHost,sshPort,sshUser,sshPass)` - nawiązuje połączenie z dowolną bazą danych. Jako protokół podajemy nazwę protokołu bazy danych, tzn. ciągi: MySQL, PostgreSQL, MSSQL, Firebird, SQLite. `Host` - to adres serwera, `port` - numer portu na których serwer nasłuchuje, `login` - nazwa użytkownika, `pass` - hasło dostępu i `nazwaBazy` - nazwa bazy danych (musi istnieć). Jeśli chcemy użyć tunelowania HTTP możemy podać adres do pliku `tunnel.php`. Chcąc skorzystać z tunelowania poprzez SSH - należy podać kolejne parametry dające dostęp do powłoki SSH.

Oto kilka najciekawszych metod i właściwości jakie ta klasa oferuje:

- `ExecSQL(zapytanie)` - wykonanie zapytania typu INSERT, UPDATE, DELETE;
- `GetRow(zapytanie)` - wykonanie zapytania typu SELECT, zwracające jeden wiersz odpowiedzi; Jeśli zapytanie nie zwróci wyników metoda zwróci wartość `NIL`, w przeciwnym wypadku zwróci obiekt typu `TBinTable`;
- `GetRows(zapytanie)` - wykonanie zapytania typu SELECT zwracające dowolnie wiele wierszy; Metoda zawsze zwróci obiekt typu `TBinTable`;
- `Start` - rozpoczęcie transakcji;
- `Commit` - zatwierdzenie transakcji;
- `Rollback` - cofnięcie transakcji;
- `AddSlashes(tekst)` - zwraca tekst poprzedzony symbolem \ przed niebezpiecznymi znakami;

- `AddDate(data)` - dodanie daty (typu `TDateTime`) to zapytania w odpowiedni dla danego typu bazy danych sposób;
- `AddDateTime(data)` - dodanie daty i godziny (ze zmiennej typu `TDateTime`) do zapytania w odpowiedni dla danego typu sposób;

Dane zwracane z bazy danych są obiektami typu `TBinTable`. Oto najważniejsze metody i właściwości obiektów tego typu:

- `Eof` - zwraca true jak skończą się wyniki odpowiedzi;
- `Next` - przejście do następnego wiersza odpowiedzi;
- `FieldValueS['nazwa']` - zwraca zawartość pola nazwa jako ciąg znaków;
- `FieldValueL['nazwa']` - zwraca zawartość pola nazwa jako liczbę całkowitą;
- `FieldValueD['nazwa']` - zwraca zawartość pola nazwa jako liczbę zmiennoprzecinkową;
- `FieldValueB['nazwa']` - zwraca zawartość pola nazwa jako zmienną typu `Boolean`;
- `Columns[i]` - zwraca nazwę i-tej kolumny;
- `Types[i]` - zwraca typ i-tej kolumny;
- `ColCount` - zwraca liczbę kolumn odpowiedzi;
- `RowCount` - zwraca liczbę wierszy odpowiedzi;
- `FieldTypeByName['nazwa']` - zwraca typ kolumny o podanej nazwie;
- `FieldExists['nazwa']` - zwraca true jak istnieje kolumna o podanej nazwie;

Chcąc dodać lub modyfikować dane w bazie danych można użyć odpowiedniego zapytania i metody `ExecSQL` obiektu `TBinDatabase`, ale można również skorzystać z dodatkowej metody `InsertUpdateRow` tej klasy. Jako parametr tej metody podajemy nazwę tabeli, na której chcemy wykonywać zmiany. Metoda zwraca obiekt typu `TBinInsertUpdateRow`. Ta klasa z kolei oferuje metody:

- `AddS(nazwa,typ,wartość)` - ustalenie wartości dla pola `nazwa`, typu `typ`. `Wartość` jest zmienną typu String;

- `AddL(nazwa,typ,wartość)` - ustalenie wartości dla pola `nazwa`, typu `typ`. `Wartość` jest zmienną typu Integer;
- `AddD(nazwa,typ,wartość)` - ustalenie wartości dla pola `nazwa`, typu `typ`. `Wartość` jest zmienną typu Double;
- `AddB(nazwa,typ,wartość)` - ustalenie wartości dla pola `nazwa`, typu `typ`. `Wartość` jest zmienną typu Boolean;
- `Execute(id)` - wykonanie zapytania; Jeśli podano `id` (Integer) - nastąpi modyfikacja rekordu (`UPDATE`). Jeśli jako `id` wprowadzono wartość 0 - zostanie dodany nowy rekord.

Przykład 1:

```
var B, T;
begin
  B:=TBinDatabase.CreateSQLite('nazwa.db','');

  T:=B.GetRow('SELECT * FROM tabela WHERE id=1');
  if T<>nil then
  begin
    ShowMessage(T.FieldValueS['imie']);
    T.Free;
  end;
  B.Free;
end;
```

Przykład 2:

```
var B, T;
begin

B:=TBinDatabase.CreateRemote('MSSQL','localhost\SQLEXPRESS',1433,'sa','karol','
mpfirma','','',0,'','');

  T:=B.GetRows('SELECT nnodoc FROM bs_invoices LIMIT 5');
  while not T.Eof do
```

```
  begin
    ShowMessage(T.FieldValueS['nnodoc']);
    T.Next;
  end;
  T.Free;
  B.Free;
end;
```

Przykład 3:

```
var B, E;
begin
  B:=TBinDatabase.CreateSQLite('nazwa.db','');

  E:=B.InsertUpdateRow('imiona');
  E.AddS('imie','varchar','Karol');
  E.AddS('wiek','int','32');
  E.Execute(0);
  E.Free;

  B.Free;
end;
```

FILTRY

bsxPrinter posiada wbudowaną obsługę plików `.in` oraz `.xml` - zgodnie z opisem w tym podręczniku. Poprzez własną wtyczkę możemy tworzyć tzw. filtry i rozszerzać obsługę bsxPrinter i inne typy plików.

UWAGA! *Przypominamy, że kiedy bsxPrinter przetwarza plik (.xml lub .in) - przed tą operacją zmienia rozszerzenie tych plików na .pr - w celu zabezpieczenia przed powtórnym przetwarzaniem.*

Chcąc utworzyć filtr należy w pliku XML wtyczki umieścić sekcję `<filters>`, a w niej znaczniki `<filter>`. Znacznik ten może przyjąć atrybuty: `type` - rodzaj filtru oraz `src` - nazwa pliku ze skryptem, gdzie znajduje się procedura obsługująca dany filtr.

Oto dostępne typy filtrów:

- `xml` - przetwarzanie pliku XML;
- `file` - przetwarzanie innego typu pliku;
- `folder` - przetwarzanie folderu z plikami;

FILTR XML

Kiedy bsxPrinter przetwarza plik XML, przed rozpoczęciem tej procedury możne wykonać się zdefiniowany przez nas filtr. Spójrzmy na przykład:

```
<filters>
 <filter type="xml" src="plugin.pas" />
</filters>
```

Kiedy teraz będzie miał być przetwarzany plik XML, bsxPrinter załaduje plik `plugin.pas` i wywoła z niego funkcję `fOnConvert(lFileName : String; Code : String) : String`. Przekaże do tej funkcji nazwę przetwarzanego pliku (`lFileName`), a w parametrze `Code` - jego treść (kod XML). Funkcja ta może zwrócić „inną" treść pliku XML lub ciąg „`@false`" - jeśli nie chcemy by ten plik był przetwarzany. Spójrzmy na przykładową zawartość pliku plugin.pas.

```
function fOnConvert(lFileName : String; Code : String) : String;
var XML : TXMLDocument;
    E,E2 : IXMLNode;
    LP: Integer;
    R,lName,lPrice,lQuantity,lVAT,lNIP : String;
begin
  LP:=0;
  R:='<?xml version="1.0" encoding="utf-8"?>'+_NL+'<root>'+_NL+'
<receipts>'+_NL;
  XML:=TXMLDocument.Create;
  try
    XML.LoadFromCode(Code);
```

```
    try
      XML.Active:=TRUE;
    except
      Exit;
    end;
    if XML.DE=nil then Exit;


    if XML.DE.Name<>'fmdoc' then Exit;


    E:=XML.DE.FindNodeS('Documents');
    if E=nil then Exit;


    E:=E.FindNodeS('Document');
    while E<>nil do
    begin
      NIP:=E.GetElementValue('NIP','');


      E2:=E.FindNodeS('Pozycje');
      if E2=nil then Exit;
   R:=R+'  <receipt step="1" symbol="'+E.GetElementValue('Symbol','')+'"
vatid="'+lNIP+'">'+_NL;
      while E2<>nil do
      begin
          lName:=E2.GetElementValue('NM','');
          lPrice:=E2.GetElementValue('CN','');
          lQuantity:=E2.GetElementValue('QT','');
          lVAT:=E2.GetElementValue('VT','');


    R:=R+'   <item name="'+lName+'" price="'+CorrectS2S(lPrice)+'"
quantity="'+lQuantity+'" vatrate="'+lVAT+'" />'+_NL;


          E2:=E2.Next;
      end;
      R:=R+'  </receipt>'+_NL;
```

```
      E:=E.Next;
      Inc(LP);
    end;

  finally
    XML.Free;
  end;
  R:=R+' </receipts>'+_NL+'</root>';
  if LP>0 then Result:=R;
end;
```

Zadaniem tego filtra jest „konwersja" pliku XML do formatu obsługiwanego przez bsxPrinter.

FILTR FOLDER

Filtr typu `folder` pozwala na dodanie obsługi plików innego rodzaju. Standardowo kiedy bsxPrinter analizuje zawartość folderu, wyszukuje plików z rozszerzeniami `.in` i `.xml`. Po tej operacji uruchamia filtry typu `folder`. Mogą one „dodać" do listy plików do przetworzenia kolejne. Spójrzmy na przykład:

```
<filters>
 <filter type="folder" src="plugin.pas" />
</filters>
```

Teraz w momencie przetwarzania folderu bsxPrinter będzie ładował skrypt `plugin.pas` i wywoływał z niego funkcję: `fOnFolder(lFolder : String) : String`. Do tej funkcji w parametrze przekazuje nazwę analizowanego folderu. Funkcja ta może zwrócić ciąg tekstowy, będący listą dodatkowych plików do przetworzenia (oddzielonych znakiem nowej linii). Na przykład, gdybyśmy chcieli dodać obsługę plików `.csv` funkcja ta by wyglądała następująco:

```
function fOnFolder(lFolder : String) : String;
var L : TStringList;
begin
  L:=TBSXUtils.DirectoryList(lFolder,'*.csv',nil);
```

```
  try
    Result:=L.Text;
  finally
    L.Free;
  end;
end;
```

UWAGA! *Jeśli wskażemy dla bsxPrinter jaki inny plik ma być przetwarzany, jego rozszerzenie również będzie automatycznie zmieniane na .pr w momencie rozpoczęcia jego przetwarzania.*

FILTR FILE

Kiedy przetwarzany jest przez bsxPrinter plik inny niż `.xml` - moment ten możemy przechwycić właście filtrem typu `file`. W ten sposób możemy rozszerzyć obsługę bsxPrinter o obsługę niemal dowolnych plików. Spójrzmy na przykład XML:

```
<filters>
<filter type="file" src="plugin.pas" />
</filters>
```

Wskazaliśmy w nim, że skryptem powiązanym z filtrem file jest `plugin.pas`. Gdy taki plik będzie analizowany bsxPrinter automatycznie wywoła z niego funkcję: `fOnParseFile(lFileName,lOrgFileName,lPrinterName : String) : String`. Przekazane parametry to: nazwa przetwarzanego pliku (`lFileName` - jest to pliku z już zmienionym rozszerzeniem na `.pr`), oryginalna nazwa (`lOrgFileName`) oraz nazwa drukarki, do której kierowany jest wydruk (`lPrinterName`). Spójrzmy na przykład:

```
function fOnParseFile(lFileName,lOrgFileName,lPrinterName : String) : String;
var L,P : TStringList;
    E : TBinInsertUpdateRow;
    i : Integer;
    lNip : String;
begin
  if TBSXUtils.URLExtractFileExt(lOrgFileName)='.csv' then
  begin
     L:=TBSXUtils.LoadLinesFromFile(lFileName,nil);
```

```
    P:=TStringList.Create;
    try
      for i:=0 to L.Count-1 do
      begin
         if Trim(L[i])='' then Exit;
         TBSXUtils.ExplodeStrS(P,L[i],';',true);
         if P.Count>=5 then
         begin
           lNip:='';
           if P.Count>=6 then lNip:=P[5];
           E:=DB.InsertUpdateRow('bsx_receipts');
           try
             E.AddS('iddoc','int','0');
             E.AddS('psource','int','4');
             E.AddS('ptype','int','1');
             E.AddS('pstatus','int','0');
             E.AddS('perrorcode','int','0');
             E.AddS('perrorstr','varchar','');
             E.AddS('premoteid','varchar',Trim(P[0]));
             E.AddS('premotestep','varchar','1');
             E.AddS('pprinter','varchar',lPrinterName);
             E.AddS('pdata','text','<root><receipts><receipt id="'+Trim(P[0])
+'" step="1" symbol="'+Trim(P[0])+'" vatid="'+lNIP+'"
printer="'+lPrinterName+'"><item name="'+Trim(P[1])+'" price="'+Trim(P[4])+'"
quantity="'+Trim(P[3])+'" vatrate="'+Trim(P[2])+'" /></receipt></receipts></
root>');
             E.AddS('pdate','datetime',DateTimeToStr(Now));
             E.Execute(0);
           finally
             E.Free;
           end;
         end;
      end;
    finally
      P.Free;
```

```
    end;
  end;
end;
```

W ten sposób sprawiliśmy, że kiedy przetwarzany jest plik `.csv` - nasz filtr wczyta ten plik, a następnie przeanalizuje jego każdą linię. Na tej podstawie zrobi wpisy w wewnętrznej bazie danych (w tabeli `bsx_receipts`). To spowoduje ich automatyczne wydrukowanie.

WŁASNE KOMENDY

Łącząc się do bsxPrinter poprzez TCP/IP, WebSocket, serwer HTTP lub tworząc pliki `.in` - możemy przekazywać różne komendy, jakie mają być przez niego wykonane. W podręczniku tym komendy te zostały opisane w poprzednich rozdziałach. Programista ma jednak możliwość rozszerzenia funkcjonalności bsxPrinter w tym zakresie dodając obsługę własnych komend.

Aby tego dokonać tworzymy tzw. *moduły* do serwera. Robimy to poprzez stworzenie „wtyczki" składającej się z pliku XML oraz powiązanego z nim pliku PAS. W pliku XML tworzymy sekcję `<modservers>`, a w niej dowolnie wiele znaczników `<modserver/>` z jednym parametrem `src` wskazującym powiązany plik Pascal. W pliku tym powinna znaleźć się funkcja: `function onServerCommand(Sender, CMD, P, LINE, AIdent, AContext, AAuthorized) : String;`.

Mając tak utworzoną wtyczkę i zarejestrowany *moduł*, kiedy przyjdzie do bsxPrinter jakakolwiek komenda - np. poprzez telnet, WebSocket czy z pliku `.in` - bsxPrinter wywoła z każdego *modułu* funkcję `onServerCommand()` z dodatkowymi parametrami. Jeśli funkcja ta zwróci napis `„true"` - spowoduje przerwanie szukania obsługi danej komendy, gdyż znaczy to, że komenda została obsłużona w tym *module*. Parametry funkcji `onServerCommand` oznaczają: `Sender` - skąd przyszło wywołanie funkcji, `CMD` - komenda (pierwszy element do znaku #, zawsze zapisana wielkimi literami), `P` - parametry dodatkowe (po symbolu #), `LINE` - pełna linia tekstu komendy, `AIdent` - identyfikator „sesji", z której przyszła komenda, `AContext` - kontekst do sesji, z której przyszło polecenie, `AAuthorized` - wartość true/false informująca czy użytkownik wysyłający komendę jest uwierzytelniony.

Spójrzmy na przykład pliku XML:

```
<module name="BinSoft.Plugin">
<modservers>
 <modserver src="modServer.pas" />
</modservers>
</module>
```

Oraz plik `modServer.pas`:

```
function onServerCommand(Sender, CMD, P, LINE, AIdent, AContext, AAuthorized) :
String;
var k : Integer;
    lTo : String;
begin
  if CMD = '' then Exit;

  if CMD = 'TEST' then
  begin
    Main.SendText(AContext,'Działa!');
    Result := 'true';
  end;
end;
```

W pliku XML zdefiniowaliśmy jeden *moduł* z powiązanym plikiem `modServer.pas`, a w nim jest funkcja `onServerCommand()`. Kiedy do bsxPrinter przyjdzie jakakolwiek komenda, funkcja ta będzie wywoływana automatycznie. Sprawdzamy w niej, czy komenda to: `TEST`. Jeśli tak - to jest to komenda obsługiwana przez nasz *moduł*, dlatego zwrócimy wartość tekstową `„true”`. W pozostałych przypadkach nic nie zwracamy, więc bsxPrinter będzie dalej przetwarzał przychodzące komendy. W przypadku komendy `TEST` używamy funkcji `Main.SendText(kontekst, tekst)`. Pozwala ona na „wysłanie” odpowiedzi pod odpowiedni kontekst - w tym przypadku do osoby/miejsca, z którego przyszła komenda.

Uwaga! Kiedy ktoś łączy się do bsxPrinter poprzez TCP/IP lub WebSocket tworzony jest dla niego tzw. kontekst i przypisany identyfikator. Za jego pośrednictwiem serwer wie, do kogo ma wysyłać odpowiedzi itp.

PYTANIA I ODPOWIEDZI

Poniżej prezentujemy listę często zadawanych pyta, jakie otrzymujemy na w dziale pomocy technicznej. Zanim więc zadasz nam pytanie - zerknij na listę poniżej.

PROGRAM NIE CHCE NAWIĄZAĆ POŁĄCZENIA Z DRUKARKĄ FISKALNA. CO ROBIĆ?

Nawiązanie połączenia z drukarką fiskalną na porcie szeregowym, wiąże się z komunikacją poprzez ten port. Z tym związanych jest szereg różnych parametrów takich jak: prędkość transmisji, kontrola parzystości i przepływu itp. bsxPrinter stosuje „domyślną" konfigurację. Jeśli nie uda mu się poprawnie połączyć, zmienia niektóre parametry automatycznie. Wydłuża to całą procedurę połączenia. Czasami też może się i tak zakończyć nieudaną próbą połączenia. Może się też zdarzyć, że połączenie będzie nawiązane, a już przy kolejnym uruchomieniu nie. W takich sytuacjach można samodzielnie zdefiniować wszystkie parametry.

W celu ręcznej konfiguracji należy otworzyć plik konfiguracyjny: `config.ini`

Znajdują się w nim sekcje o nazwach `[FiscalPrinterXX]` gdzie `XX` to liczba całkowita. W każdej z takich sekcji opisana jest konfiguracja jednej drukarki. Możemy tam dopisać następujące parametry: `BaudRate` (prędkość transmisji, wymagane, np. 9600), `Parity` (bity parzystości: 0=Brak, 1=Odd, 2=Even, 3=Mark, 4=Space), `StopBits` (ilość bitów stopu: 1, 2, 3), `ByteSize` (wielkość bajtu), `RSMode` (tryb RS485: 1/0), `Stream` (protokół przepływu: 0=Brak, 1=XonXoff, 2=Hardware).

Przykład:

```
BaudRate=9600
Parity=0
StopBits=1
ByteSize=8
RSMode=1
Stream=2
```

CZY BSXPRINTER MOŻE PRZECHOWYWAĆ SWOJĄ BAZĘ W MYSQL?

Tak. Domyślnie program bsxPrinter tworzy lokalną bazę danych SQLite, w której przechowuje swoje informacje (listę przetworzonych plików, wydrukowanych dokumentów itp). Możemy jednak zmienić ten parametr i wykorzystać inne miejsce do tego celu. Wystarczy wyedytować plik `config.ini` i w sekcji `[Settings]` dopisać opcję `DatabaseURL` z parametrem opisującym bazę danych. Przykład:

```
DatabaseURL=MySQL#localhost#3306#root#bsx#bsxprinter
```

Powyższa linia spowoduje, że bsxPrinter podczas uruchamia połączy się z serwerem `MySQL` na komputerze pod adresem `localhost`, porcie `3306`. Użyje użytkownika `root`, hasła `bsx` i wybierze bazę danych o nazwie `bsxprinter`. Stworzy tam niezbędne dla siebie tabele i w nich będzie przechowywał swoje informacje.

CZY MOŻNA BEZPOŚREDNIO MANIPULOWAĆ WEWNĘTRZNĄ BAZĄ DANYCH BSXPRINTER?

Tak. Użytkownik może np. samodzielnie dodawać wpisy do wewnętrznej tabeli `bsx_receipts`, a bsxPrinter będzie te zmiany „widział" i będzie na nie reagował.

Możliwy scenariusz integracji z bsxPrinter jest zatem na przykład taki, że w jego pliku konfiguracyjnym podajemy dane dostępowe do swojego serwera MySQL, na którym działa nasza aplikacja (np. sklep on-line). Po uruchomieniu bsxPrinter połączy się do tej bazy i automatycznie stworzy w niej swoją strukturę tabel. Następnie programista z poziomu swojej aplikacji może dodawać wpisy do tabel bsxPrintera i w ten sposób zlecać wydruk paragonu, lub też odczytywać status wydruku itp.

POMOC TECHNICZNA

W przypadku wszelkich problemów w funkcjonowaniu aplikacji bsxPrinter lub w przypadku zaistnienia dodatkowych potrzeb jeśli chodzi o tą aplikację - prosimy pisać na adres: pomoc@bsxprinter.pl.