

DOKUMENTACE KNIHOVNY PORTÁL ZP DLL

**ŘEŠENÍ A SPRÁVA
INTERNETOVÉHO PORTÁLU**

ZDRAVOTNÍCH POJIŠŤOVEN

**listopad 2005 – leden 2023
Verze 2.14**

**Asseco Central Europe, a.s.
Copyright © 2005-2023 Asseco Central Europe, a.s.**

Autor: Pavel Koukal

Obsah

1.	<i>Popis knihovny.....</i>	<i>4</i>
2.	<i>Licence.....</i>	<i>4</i>
3.	<i>Instalace knihovny.....</i>	<i>5</i>
3.1	Pro využití COM třídy PZPDLL.....	5
3.2	Pro naimportování funkcí z DLL.....	5
3.3	Závislosti.....	5
3.3.1	Podepisovací komponenta Signer.....	5
3.3.2	ZIP komprimační knihovna.....	5
3.3.3	OpenSSL knihovny.....	5
4.	<i>Vlastnosti a metody třídy PZPDLL.....</i>	<i>5</i>
4.1	function Init().....	6
4.2	function Run().....	6
4.3	function SMSAuth().....	7
4.4	function SMSRun().....	7
4.5	function RunNonVis().....	8
4.6	function SMSAuthNonVis().....	9
4.7	function SMSRunNonVis().....	9
4.8	function SchrOper().....	9
5.	<i>Vyexportované funkce z DLL knihovny.....</i>	<i>10</i>
5.1	function DIIPZP_Run().....	10
5.2	function DIIPZP_SMSAuth().....	10
5.3	function DIIPZP_SMSRun().....	10
5.4	function DIIPZP_RunNonVis().....	10
5.5	function DIIPZP_SMSAuthNonVis().....	11
5.6	function DIIPZP_SMSRunNonVis().....	11
5.7	function DIIPZP_SchrOper().....	11
6.	<i>Informace o SW, který knihovnu využívá.....</i>	<i>11</i>
7.	<i>Průběh komunikace.....</i>	<i>11</i>
7.1	Sestavení a podepsání XML výzvy.....	12
7.2	Odeslání XML výzvy na klientskou bránu ZP.....	13
7.3	Přijetí XML odpovědi a její zpracování.....	13
7.3.1	Komunikace skončila chybou.....	13
7.3.2	Komunikace skončila v pořádku, zpracování na serveru vrátilo chybu.....	13
7.3.3	Komunikace i zpracování na serveru skončilo v pořádku.....	14
7.3.4	Ověření podpisu XML odpovědi.....	14
8.	<i>Použití knihovny ve vývojovém prostředí.....</i>	<i>16</i>

8.1	Návod pro importování COM komponenty do Delphi.....	16
8.2	Návod pro importování COM komponenty do Visual Studia.....	16
9.	<i>Příklady použití třídy PZPDLL.....</i>	<i>17</i>
9.1	COM verze v Borland Delphi 7.....	17
9.2	DLL verze v Borland Delphi 7.....	18
9.3	COM verze ve Visual Studiu a C# Windows Forms Application projektu.....	19
10.	<i>Chybové stavy.....</i>	<i>21</i>
11.	<i>Debug výstup knihovny.....</i>	<i>21</i>

1. Popis knihovny

Knihovna PortalZP (název souboru PortalZP.dll) je určená pro implementaci komunikace softwaru třetích stran s portálem zdravotních pojišťoven (<https://www.portalzp.cz>), a to pro klienty tohoto portálu formou komunikační brány. Vlastní komunikace s portálem je popsána v Technickém řešení č. 6, které je k dispozici na https://www.portalzp.cz/distribuce.ext/PZP-kom_brana_klient/

Knihovna obsahuje jednu COM třídu PZPDLL, která je určená pro komunikaci s klientskými komunikačními bránami portálu ZP a dále obsahuje vyexportované funkce, které interně vytvářejí instanci třídy PZPDLL a volají její metody. Je tedy možné využít tyto vyexportované DLL funkce bez využití a registrace COM serveru.

Knihovnu lze využít pro odeslání souborů:

- vyúčtování plateb (VYU)
- hromadného odeslání zaměstnavatele (HOZ)
- přehledu plateb pojistného zaměstnavatele (PPPZ)
- ověření pojištěnce (VERPOJ)
- ověření zdravotnického zařízení (VERZZ)
- odeslání registračních lístků (ORL)
- přesměrování (PRESMER)
- dotaz na schránky klienta (SCHRANKA)
- dotaz exekutora na právnickou osobu (EXE_PO)
- dotaz exekutora na fyzickou osobu (EXE_FO)
- prvotní import N Příloh 2 (P2_IMPORT)
- zavedení nového certifikátu (NOVY_CERT)
- formuláře schránky (SCHRANKA_PODANI)
- žádost Celní správy o bezdlužnost (BZD)
- hlášení Policie o dopravních nehodách (HDN)

Tyto soubory či dotazy lze odeslat jak do pěti pojišťoven na Portálu ZP (ČPZP, OZP, RBP, ZPŠ a VoZP ČR), tak do společné zóny všech pojišťoven (PZP) nebo do pilotního prostředí projektu Portál ZP a jeho společné zóny (PILOT, PILOT-PZP).

Tato knihovna je použitelná v jakémkoliv vývojovém nástroji, který podporuje technologii COM (Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual C#, Active Server Pages, Delphi atd.) nebo umožňuje volat exportované funkce z DLL knihovny.

2. Licence

Copyright (c) 2005-2023 Asseco Central Europe, a.s.

Copyright (c) 2021-2023 Svaz zdravotních pojišťoven ČR, z.s.

Copyright (c) 2005-2023 Česká průmyslová zdravotní pojišťovna

Copyright (c) 2005-2023 Oborová zdravotní pojišťovna

Copyright (c) 2005-2023 RBP, zdravotní pojišťovna

Copyright (c) 2007-2023 Vojenská zdravotní pojišťovna České republiky

Copyright (c) 2005-2023 Zaměstnanecká pojišťovna Škoda

Copyright (c) 2023 Zdravotní pojišťovna ministerstva vnitra České republiky

Všechna práva vyhrazena.

Šíření a využití je dovoleno za těchto podmínek:

1. Tato knihovna je dána k dispozici zdravotním pojišťovnám účastnícím se projektu Portál ZP, výrobcům SW vyvíjejícím software pro komunikaci s tímto Portálem a tím i klientům využívajícím software těchto výrobců SW.
2. Jakékoli zásahy do binární verze knihovny mimo Asseco Central Europe, a.s. jsou nepřipustné.
3. Závazným pro vlastní komunikaci je jen Technické řešení č. 6 (POPIS KOMUNIKACE S PORTÁLEM (KLIENT)), nikoli tato knihovna.

TENTO SOFTWARE JE POSKYTOVÁN SPOLEČNOSTÍ Asseco Central Europe, a.s. "TAK JAK JE" A VEŠKERÉ ZÁRUKY A PRÁVA NA POUŽITÍ, OBCHODOVATELNOST, ZPŮSOBILOST K URČITÉMU ÚČELU JSOU VYHRAZENY. SPOLEČNOST Asseco Central Europe, a.s. NENÍ

V ŽÁDNÉM PŘÍPADĚ ZODPOVĚDNÁ ZA JAKÉKOLIV PŘÍMÉ, NEPŘÍMÉ, NÁHODNÉ ČI NÁSLEDNÉ POŠKOZENÍ ZAPŘÍČINUJÍCÍ ZTRátu DAT, ZISKU NEBO PODOBNÉ ZPŮSOBENÉ JAKÝMKOLIV POUŽÍVÁNÍM SOFTWARE.

3. Instalace knihovny

3.1 Pro využití COM třídy PZPDLL

Pro využití COM třídy PZPDLL v PortalZP.dll je nutné knihovnu zaregistrovat do systému. Registrace se provede pomocí programu REGSVR32, který je součástí Windows.

Je jedno, kde bude PortalZP.dll umístěna, adresář nemusí být obsažen ve vyhledávací cestě nebo přímo v adresáři aplikace, která ji využívá. Doporučujeme ale nahrát buď přímo do systému (SYSTEM32) nebo do adresáře aplikace využívající tuto knihovnu.

Vlastní registrace se potom provede tímto příkazem pod admin právy:

```
regsvr32 PortalZP.dll
```

kde k souboru PortalZP.dll musí být uvedena plná cesta.

3.2 Pro naimportování funkcí z DLL

V tomto případě nemusí být PortalZP.dll v registrech zaregistrována, stačí, když je dostupná volané aplikaci, tedy ve stejném adresáři či adresáři definovaném v PATH proměnné prostředí.

3.3 Závislosti

Knihovna je závislá na těchto dalších knihovnách:

3.3.1 Podepisovací komponenta Signer

Knihovna PortalZP využívá podepisovací komponentu Signer pro podepisování a ověřování dat, stejně jako HTML stránky portálu ZP. Pro správný běh knihovny je zapotřebí verze Signera 4.2 nebo novější, neboť tato verze již obsahuje podporu pro generování podpisu pomocí hashovací funkce SHA-256. Tato funkce je ale závislá na operačním systému, a pokud ji nepodporuje, je vytvářen podpis pomocí SHA-1 funkce a tato informace je vrácena knihovnou po ukončení komunikace s PZP. Instalaci knihovny Signer naleznete také na stránkách Portálu ZP v sekci „Ke stažení“ -

<https://www.portalzp.cz/download.html#Signer>.

Signer je vyžadovaný i v případě, kdy se využívají pouze funkce s SMS autentizací a to z důvodu, že knihovna PortalZP pomocí Signer komponenty ověřuje podpis dat vrácených ze serveru.

3.3.2 ZIP komprimační knihovna

Dále PortalZP využívá knihovnu DelZip192.dll pro práci se ZIP archívy. Tato knihovna musí být distribuovaná společně s knihovnou PortalZP. Knihovna se neregistruje, jen musí být umístěna ve stejném adresáři, kde je knihovna PortalZP.dll.

3.3.3 OpenSSL knihovny

Do verze knihovny PortalZP 2.6 využívala knihovna ke komunikaci ActiveX objekt Internet Exploreru. Od verze 2.6 využívá k HTTPS komunikaci knihovny OpenSSL a je nutné s knihovnou distribuovat i knihovny libeay32.dll a ssleay32.dll. Obě knihovny musí být umístěny ve stejném adresáři, kde je knihovna PortalZP.dll.

4. Vlastnosti a metody třídy PZPDLL

K dispozici je 8 metod: **Init()**, **Run()**, **SMSAuth()**, **SMSRun()**, **RunNonVis()**, **SMSAuthNonVis()**, **SMSRunNonVis()** a **SchrOper()**.

Metoda **SchrOper()** slouží pro alternativní komunikaci s Portálem ZP s účelem (službou) SCHRANKA, kdy odstraňuje tyto nedostatky standardních metod **Run()** a **SMSRun()**, resp. **RunNonVis()** a **SMSRunNonVis()**:

V případě požadavku na stažení detailu zprávy schránky či označení zprávy jako přečtené či smazané je pro každý požadavek určen jeden vstupní fyzický soubor. Přečtení např. 10 zpráv znamená

vytvoření 10 dočasných souborů a předání tohoto seznamu jako vstupní parametr DataIn do daných metod. Standardní metody **Run()** a **SMSRun()**, resp. **RunNonVis()** a **SMSRunNonVis()** pracují pouze s jedním výstupem a při použití mnohonásobných požadavků schránek v jednom dotazu vrací Portál ZP odpovídající množství odpovědí v <DATA> elementu. Dané metody vracejí pouze první rozparsovanou odpověď a všechny odpovědi je nutné parsovat z výstupního parametru Debug_XML_odpoved.

Tyto nedostatky se týkají jen práce se schránkou a to jen v případě, že je požadováno zaslání více souborů, např. 10 souborů pro stažení 10 zpráv v rámci jediné komunikace.

4.1 function Init()

Metoda **Init()** vrací hodnotu **true**, pokud je COM objekt třídy PZPDLL řádně inicializován.

ID: 0x00000065

Definice metody

Pascal:

```
function Init: HRESULT;
```

C++:

```
HRESULT Init(void);
```

4.2 function Run()

Metoda **Run()** čte vstupní parametry, sestaví XML výzvu, nechá ji pomoci komponenty Signer podepsat osobním certifikátem klienta, komunikuje se serverem PZP a nakonec ověří podpis vrácených dat a plní výstupní parametry.

ID: 0x00000070

Definice metody

Pascal:

```
function Run(const ZP: WideString; var ZP_Cert_SN: WideString; var ZP_Cert_IDN: WideString; const Sluzba: WideString; const DataIn: WideString; Timeout: Smallint; SWInfo: SWInfoRec; var ID_Podani: Integer; var Chyba: WideString; var Varovani: WideString; var DataOut: WideString; var Debug_internal_warning: WideString; var Debug_XML_vyzva: WideString; var Debug_XML_odpoved: WideString): HRESULT;
```

C++:

```
HRESULT Run(BSTR ZP, BSTR* ZP_Cert_SN, BSTR* ZP_Cert_IDN, BSTR Sluzba, BSTR DataIn, short Timeout, SWInfoRec SWInfo, int* ID_Podani, BSTR* Chyba, BSTR* Varovani, BSTR* DataOut, BSTR* Debug_internal_warning, BSTR* Debug_XML_vyzva, BSTR* Debug_XML_odpoved);
```

Parametry

in	ZP	Zkratka ZP bez interpunkce (jedna z hodnot CPZP, OZP, RBP, ZPS, VoZP, PZP, PILOT, PILOT-PZP)
in, out	ZP_Cert_SN	Sériové číslo certifikátu příslušné ZP.
in, out	ZP_Cert_IDN	IDN certifikátu příslušné ZP.
in	Sluzba	Zkratka služby, o jejíž podání se jedná (jedna z hodnot VYU, HOZ, PPPZ, VERPOJ, VERZZ, ORL, PRESMER, SCHRANKA, EXE_PO, EXE_FO, P2_IMPORT, NOVY_CERT, SCHRANKA_PODANI, BZD, HDN)
in	DataIn	Plné cesty ke vstupním souborům oddělené od sebe znakem " "
in	Timeout	Timeout komunikace. Zadává se v sekundách.
in	SWInfo	Informace o SW, který používá knihovnu, uloženy a předány ve struktuře SWInfoRec – viz kapitola 6
out	ID_Podani	Číslo uloženého podání, došlo-li ke komunikaci s portálem
out	Chyba	Kód chyby; "0"=OK, více v seznamu chyb v kapitole 9 a v Technickém řešení č. 6
out	Varovani	Případné varování pro klienta (prázdné = bez varování)
out	DataOut	HTML protokol nebo popis chyby. Byla-li volána služba SCHRANKA (viz parametr Sluzba) a bylo-li zasláno více souborů

		(viz parametr DataIn), bude takto vyseparována odpověď jen na první z nich. Jedná se o nedostatek, který nelze opravit se zachováním zpětné kompatibility, nicméně týká se jen práce se schránkami a řeší jej metoda SchrOper()
out	Debug_internal_warning	Případné interní varování knihovny určené pro výrobce SW
out	Debug_XML_vyzva	Sestavené XML výzvy
out	Debug_XML_odpoved	XML odpověď vrácená ze serveru

4.3 function SMSAuth()

Metoda **SMSAuth()** čte vstupní parametry, sestaví XML žádost pro přihlášení klienta a vygenerování SMS kódu pro další komunikaci. Následně tuto žádost předá serveru PZP, ověří podpis odpovědi a plní výstupní parametry.

ID: 0x00000066

Definice metody

Pascal:

```
function SMSAuth(const ZP: WideString; var ZP_Cert_SN: WideString; var ZP_Cert_IDN: WideString; const Jmeno: WideString; const Heslo: WideString; Timeout: Smallint; SWInfo: SWInfoRec; var ID_Podani: Integer; var Chyba: WideString; var Varovani: WideString; var DataOut: WideString; var Debug_internal_warning: WideString; var Debug_XML_vyzva: WideString; var Debug_XML_odpoved: WideString): HRESULT;
```

C++:

```
HRESULT SMSAuth(BSTR ZP, BSTR* ZP_Cert_SN, BSTR* ZP_Cert_IDN, BSTR Jmeno, BSTR Heslo, short Timeout, SWInfoRec SWInfo, int* ID_Podani, BSTR* Chyba, BSTR* Varovani, BSTR* DataOut, BSTR* Debug_internal_warning, BSTR* Debug_XML_vyzva, BSTR* Debug_XML_odpoved);
```

Parametry

in	ZP	Zkratka ZP bez interpunkce (jedna z hodnot CPZP, OZP, RBP, ZPS, VoZP, PZP, PILOT, PILOT-PZP)
in, out	ZP_Cert_SN	Sériové číslo certifikátu příslušné ZP.
in, out	ZP_Cert_IDN	IDN certifikátu příslušné ZP.
in	Jmeno	Přihlašovací jméno (konto) do Portálu ZP.
in	Heslo	Přihlašovací heslo do Portálu ZP nebo prázdný řetězec, pokud klient heslo nepoužívá.
in	Timeout	Timeout komunikace. Zadává se v sekundách.
in	SWInfo	Informace o SW, který používá knihovnu, uloženy a předány ve struktuře SWInfoRec – viz kapitola 6
out	ID_Podani	Číslo uloženého podání, došlo-li ke komunikaci s portálem
out	Chyba	Kód chyby; "0"=OK, více v seznamu chyb v kapitole 9 a v Technickém řešení č. 6
out	Varovani	Případné varování pro klienta (prázdné = bez varování)
out	DataOut	HTML protokol nebo popis chyby. Byla-li volána služba SCHRANKA (viz parametr Sluzba) a bylo-li zasláno více souborů (viz parametr DataIn), bude takto vyseparována odpověď jen na první z nich. Jedná se o nedostatek, který nelze opravit se zachováním zpětné kompatibility, nicméně týká se jen práce se schránkami a řeší jej metoda SchrOper()
out	Debug_internal_warning	Případné interní varování knihovny určené pro výrobce SW
out	Debug_XML_vyzva	Sestavené XML výzvy
out	Debug_XML_odpoved	XML odpověď vrácená ze serveru

4.4 function SMSRun()

Metoda **SMSRun()** čte vstupní parametry, sestaví XML výzvu, komunikuje se serverem PZP a nakonec ověří podpis vrácených dat a plní výstupní parametry. Metoda **SMSRun()** je shodná

s metodou **Run()**, jen nepoužívá k ověření klienta podpis certifikátem, ale přihlašovací jméno klienta a SMS kód, který mu byl zaslán na mobilní telefon po úspěšném volání metody **SMSAuth()**.

ID: 0x00000071

Definice metody

Delphi:

```
function SMSRun(const ZP: WideString; var ZP_Cert_SN: WideString; var
ZP_Cert_IDN: WideString; const Sluzba: WideString; const DataIn:
WideString; const Jmeno: WideString; const SMSKod: WideString; Timeout:
Smallint; SWInfo: SWInfoRec; var ID_Podani: Integer; var Chyba: WideString;
var Varovani: WideString; var DataOut: WideString; var
Debug_internal_warning: WideString; var Debug_XML_vyzva: WideString; var
Debug_XML_odpoved: WideString): HRESULT;
```

C++:

```
HRESULT SMSRun(BSTR ZP, BSTR* ZP_Cert_SN, BSTR* ZP_Cert_IDN, BSTR Sluzba,
BSTR DataIn, BSTR Jmeno, BSTR SMSKod, short Timeout, SWInfoRec SWInfo, int*
ID_Podani, BSTR* Chyba, BSTR* Varovani, BSTR* DataOut, BSTR*
Debug_internal_warning, BSTR* Debug_XML_vyzva, BSTR* Debug_XML_odpoved);
```

Parametry

in	ZP	Zkratka ZP bez interpunkce (jedna z hodnot CPZP, OZP, RBP, ZPS, VoZP, PZP, PILOT, PILOT-PZP). Je možné použít i jinou ZP než byla použita u metody SMSAuth(), neboť voláním SMSAuth() dojde k přihlášení ke všem ZP ve stejném prostředí (ostrém nebo pilotním) a je tedy možné se přihlásit např. k ČPZP a stejným SMS kódem odeslat podání jak pro ČPZP, tak OZP, RBP apod. Pro odeslání podání do PILOT nebo PILOT-PZP je nutné se přihlásit do jedné z uvedených ZP.
in, out	ZP_Cert_SN	Sériové číslo certifikátu příslušné ZP.
in, out	ZP_Cert_IDN	IDN certifikátu příslušné ZP.
in	Sluzba	Zkratka služby, o jejíž podání se jedná (jedna z hodnot VYU, HOZ, PPPZ, VERPOJ, VERZZ, ORL, PRESMER, SCHRANKA, EXE_PO, EXE_FO, P2_IMPORT, NOVY_CERT, SCHRANKA_PODANI, BZD, HDN)
in	DataIn	Plné cesty ke vstupním souborům oddělené od sebe znakem " "
in	Jmeno	Přihlašovací jméno (konto) do Portálu ZP.
in	SMSKod	Platný SMS kód získaný z SMS zprávy, kterou klient obdrží po úspěšném přihlášení pomocí metody SMSAuth().
In	Timeout	Timeout komunikace. Žadává se v sekundách.
in	SWInfo	Informace o SW, který používá knihovnu, uloženy a předány ve struktuře SWInfoRec – viz kapitola 6
Out	ID_Podani	Číslo uloženého podání, došlo-li ke komunikaci s portálem
Out	Chyba	Kód chyby; "0"=OK, více v seznamu chyb v kapitole 9 a v Technickém řešení č. 6
Out	Varovani	Případné varování pro klienta (prázdné = bez varování)
Out	DataOut	HTML protokol nebo popis chyby. Byla-li volána služba SCHRANKA (viz parametr Sluzba) a bylo-li zasláno více souborů (viz parametr DataIn), bude takto vyseparována odpověď jen na první z nich. Jedná se o nedostatek, který nelze opravit se zachováním zpětné kompatibility, nicméně týká se jen práce se schránkami a řeší jej metoda SchrOper()
Out	Debug_internal_warning	Případné interní varování knihovny určené pro výrobce SW
Out	Debug_XML_vyzva	Sestavené XML výzvy
Out	Debug_XML_odpoved	XML odpověď vrácená ze serveru

4.5 function RunNonVis()

Metoda **RunNonVis()** je shodná s metodou **Run()** s jediným rozdílem: nezobrazuje vizuální okno s průběhem komunikace s Portálem ZP.

ID: 0x00000096

4.6 function SMSAuthNonVis()

Metoda **SMSAuthNonVis()** je shodná s metodou **SMSAuth()** s jediným rozdílem: nezobrazuje vizuální okno s průběhem komunikace s Portálem ZP.

ID: 0x00000097

4.7 function SMSRunNonVis()

Metoda **SMSRunNonVis()** je shodná s metodou **SMSRun()** s jediným rozdílem: nezobrazuje vizuální okno s průběhem komunikace s Portálem ZP.

ID: 0x00000098

4.8 function SchrOper()

Metoda **SchrOper()** slouží pro alternativní komunikaci s Portálem ZP s účelem (službou) SCHRANKA, kdy umožňuje jednodušeji předávat seznam požadovaných zpráv či schránek a zpřístupňuje ve výstupním parametru DataOut celý element se všemi odpovědi. Více viz podrobnější popis na začátku kapitoly 4.

Metoda na základě vstupních parametrů připraví odpovídající datové soubory dle rozhraní, nechá zpracovat serverem PZP, ověří podpis vrácených dat a plní výstupní parametry.

Metoda ve vstupních parametrech umožňuje rozlišit jak autentizaci pomocí certifikátu, tak SMS kódem a zároveň obsahuje parametr NonVis určující, zda se nemá zobrazovat vizuální okno s průběhem komunikace s Portálem ZP. Proto tato metoda nemá přepsané obdobné metody s prefixem SMS či postfixem NonVis a daná kombinace je řízena pouze vstupními parametry.

Další podrobnosti jsou v dokumentaci „Datové rozhraní pro přístup k datům ve schránkách“ (P4ZP-PHP_APP_SCH_P01.doc)

ID: 0x000000A0

Definice metody

Delphi:

```
function SchrOper(const ZP: WideString; var ZP_Cert_SN: WideString; var
ZP_Cert_IDN: WideString; const Operace: WideString; const Identifikatory:
WideString; const SMSJmeno: WideString; const SMSKod: WideString; Timeout:
Smallint; SWInfo: SWInfoRec; NonVis: Smallint; var ID_Podani: SYSINT; var
Chyba: WideString; var Varovani: WideString; var DataOut: WideString; var
Debug_internal_warning: WideString; var Debug_XML_vyzva: WideString; var
Debug_XML_odpoved: WideString): HResult; stdcall;
```

C++:

```
HRESULT SMSRun(BSTR ZP, BSTR* ZP_Cert_SN, BSTR* ZP_Cert_IDN, BSTR Operace,
BSTR Identifikatory, BSTR SMSJmeno, BSTR SMSKod, short Timeout, SWInfoRec
SWInfo, short NonVis, int* ID_Podani, BSTR* Chyba, BSTR* Varovani, BSTR*
DataOut, BSTR* Debug_internal_warning, BSTR* Debug_XML_vyzva, BSTR*
Debug_XML_odpoved);
```

Parametry

in	ZP	Zkratka ZP bez interpunkce (jedna z hodnot CPZP, OZP, RBP, ZPS, VoZP, PZP, PILOT, PILOT-PZP).
in, out	ZP_Cert_SN	Sériové číslo certifikátu příslušné ZP.
in, out	ZP_Cert_IDN	IDN certifikátu příslušné ZP.
in	Operace	Název operace schránky, která má být použita. SCHRANKY – vrací seznam dostupných schránek ZPRAVY_VSE – vrací seznam všech zpráv z požadovaných schránek. Ve vstupních identifikátorech je předáváno ID schránky či seznam ID schránek oddělených znakem " " ZPRAVY_NOVE – vrací seznam nových zpráv z požadovaných schránek. Ve vstupních identifikátorech je předáváno ID schránky či seznam ID schránek oddělených znakem " " ZPRAVA_DETAIL – vrací detail požadovaných zpráv včetně případných příloh. Ve vstupních identifikátorech je předáváno ID

		zprávy či seznam ID požadovaných zpráv oddělených znakem " " ZPRAVA_PRECTI – označí požadované zprávy jako přečtené. Ve vstupních identifikátorech je předáváno ID zprávy či seznam ID požadovaných zpráv oddělených znakem " " ZPRAVA_SMAZ – označí požadované zprávy jako smazané. Ve vstupních identifikátorech je předáváno ID zprávy či seznam ID požadovaných zpráv oddělených znakem " "
in	Identifikatory	V případě operace SCHRANKY je zde očekáván prázdný řetězec. V ostatních operacích je zde očekáván seznam identifikátorů (ID schránek či ID zpráv) oddělených znakem " ". Co identifikátor, to jeden „virtuální“ vstupní soubor předávaný v požadavku a zároveň i odpověď ve formě <SOUBOR> elementu v DataOut.
in	SMSJmeno	Přihlašovací jméno (konto) do Portálu ZP v případě SMS autentizace. Pokud má být podání podepsáno certifikátem, předává se zde prázdný řetězec.
in	SMSKod	Platný SMS kód získaný z SMS zprávy, kterou klient obdrží po úspěšném přihlášení pomocí metody SMSAuth(). Pokud má být podání podepsáno certifikátem, předává se zde prázdný řetězec.
In	TimeOut	Timeout komunikace. Zadává se v sekundách.
in	SWInfo	Informace o SW, který používá knihovnu, uloženy a předány ve struktuře SWInfoRec – viz kapitola 6
In	NonVis	Hodnota 0 nebo 1. 0 – bude zobrazováno vizuální okno s průběhem komunikace 1 – nebude zobrazováno vizuální okno s průběhem komunikace
Out	ID_Podani	Číslo uloženého podání, došlo-li ke komunikaci s portálem
Out	Chyba	Kód chyby; "0"=OK, více v seznamu chyb v kapitole 9 a v Technickém řešení č. 6
Out	Varovani	Případné varování pro klienta (prázdné = bez varování)
Out	DataOut	V případě chyby (tj. Chyba není „0“) se chová stejně jako u metod Run či SMSRun (resp. RunNonVis a SMSRunNonVis). Tj. obsah DataOut obsahuje textový popis chyby či HTML stránku s chybou, tedy rozparovanou první odpověď. Pokud komunikace skončila v pořádku (Chyba="0"), potom obsahem je celý element <Data> se všemi odpovědi v podelementech <Soubor>. Pro získání odpovědi je tedy nutné toto vnitřní XML rozparovat a projít všechny <Soubor> elementy a z nich získat odpovědi.
Out	Debug_internal_warning	Případné interní varování knihovny určené pro výrobce SW
Out	Debug_XML_vyzva	Sestavené XML výzvy
Out	Debug_XML_odpoved	XML odpověď vrácená ze serveru

5. Vyexportované funkce z DLL knihovny

PortalZP.dll knihovna poskytuje kromě třídy PZPDLL také vyexportované funkce poskytující shodnou funkčnost jako metody zmíněné PZPDLL třídy kromě metody Init(). Funkce mají prefix DIIPZP_ a mají stejné parametry a funkčnost jako metody, které interně volají.

5.1 function DIIPZP_Run()

Funkce **DIIPZP_Run()** je shodná s metodou **PZPDLL.Run()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu Run().

5.2 function DIIPZP_SMSAuth()

Funkce **DIIPZP_SMSAuth()** je shodná s metodou **PZPDLL.SMSAuth()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu SMSAuth().

5.3 *function DIIPZP_SMSRun()*

Funkce **DIIPZP_SMSRun()** je shodná s metodou **PZPDLL.SMSRun()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu SMSRun().

5.4 *function DIIPZP_RunNonVis()*

Funkce **DIIPZP_RunNonVis()** je shodná s metodou **PZPDLL.RunNonVis()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu RunNonVis().

5.5 *function DIIPZP_SMSAuthNonVis()*

Funkce **DIIPZP_SMSAuthNonVis()** je shodná s metodou **PZPDLL.SMSAuthNonVis()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu SMSAuthNonVis().

5.6 *function DIIPZP_SMSRunNonVis()*

Funkce **DIIPZP_SMSRunNonVis()** je shodná s metodou **PZPDLL.SMSRunNonVis()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu SMSRunNonVis().

5.7 *function DIIPZP_SchrOper()*

Funkce **DIIPZP_SchrOper()** je shodná s metodou **PZPDLL.SchrOper()**, má shodné parametry a interně vytváří instanci třídy PZPDLL a volá její metodu SchrOper().

6. Informace o SW, který knihovnu využívá

Metody či vyexportované DLL funkce komunikující se serverem Portálu ZP používají ve svých volání vstupní parametr SWInfo pro předání informací o softwaru, který knihovnu PortalZP využívá.

Informace jsou využity k případnému řešení problému s klientem Portálu ZP. Stává se totiž, že je sice zalogována XML výzva pro komunikační bránu klienta, nicméně dotyčný klient pak vznese dotaz, po jehož analýze dospěje Asseco CE k závěru, že je to vlastně dotaz na technickou podporu výrobce příslušného SW, nicméně ne vždy je do XML výzvy vložen komentář s identifikací daného SW a / nebo příslušné technické podpory.

Informace o SW jsou uloženy ve struktuře SWInfoRec, obsahující 4 nepovinné atributy (mohou být předány prázdné řetězce):

- **Nazev** (string): Název software
- **Verze** (string): Verze software
- **Vyrobce** (string): Výrobce software
- **Helpline** (string): Kontakt na helpline výrobce SW

Definice struktury

Delphi:

```
SWInfoRec = packed record  
    Nazev: WideString;  
    Verze: WideString;  
    Vyrobce: WideString;  
    Helpline: WideString;
```

end;

C++:

```
struct SWInfoRec {  
    BSTR Nazev;  
    BSTR Verze;  
    BSTR Vyrobce;  
    BSTR Helpline;  
};
```

7. Průběh komunikace

Při komunikaci s Portálem ZP proběhne ověření klienta a to buď:

- a) na základě podpisu vytvořeného podepsáním podání certifikátem klienta
- b) nebo pomocí přihlašovacího jména a hesla.

V případě podepsání certifikátem dochází na serveru k jednorázovému ověření klienta, jeho přihlášení, předání podání a následnému odhlášení klienta z Portálu ZP.

V případě přihlášení pomocí jména a hesla dojde na serveru k ověření uživatele a následně k jeho přihlášení do všech zdravotních pojišťoven v ostrém či pilotním prostředí – podle toho, která ZP byla adresována. Poté je klientovi odeslána SMS zpráva na jeho mobilní telefon, kde je uveden SMS kód, který společně s přihlašovacím jménem slouží k ověření identity klienta. Platnost tohoto SMS kódu je časově omezena a po vypršení platnosti kódu je nutné se znovu přihlásit pomocí SMSAuth či SMSAuthNonVis. Během platnosti SMS kódu pro dané přihlašovací jméno odmítne Portál ZP poslat další SMS kód stejnému klientovi (přihlašovacímu jménu) a je nutné při volání metody SMSRun/SMSRunNonVis/SchrOper použít již dříve zasláný SMS kód.

Celá komunikace s portálem ZP probíhá pomocí metody Run/RunNonVis/SchrOper (v případě použití certifikátu) nebo SMSRun/SMSRunNonVis/SchrOper (v případě ověření pomocí SMS zprávy), kde je potřeba předat správně vstupní parametry ZP, ZP_Cert_SN, ZP_Cert_IDN, Sluzba, DataIn a TimeOut. Pokud proběhnou v pořádku testy vstupních parametrů, začíná vlastní komunikace s klientskou bránou zdravotní pojišťovny předané v parametru ZP. Pokud jsou data odesílaná na bránu zdravotní pojišťovny větší než 10kB, dojde k automatické kompresi za účelem zmenšení objemu přenášených dat.

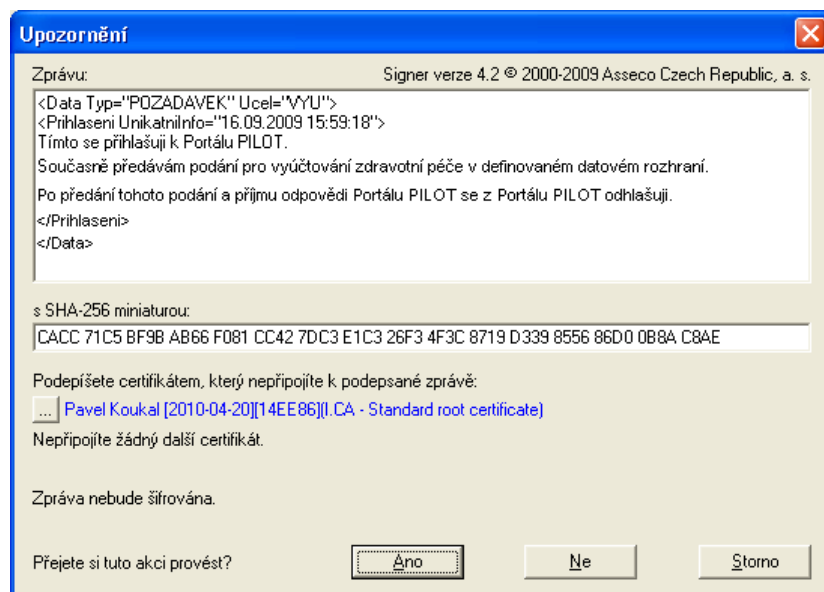
7.1 Sestavení a podepsání XML výzvy

Z předaných vstupních parametrů ZP, Sluzba a DataIn je sestavena XML výzva, která se doplní buď podpisem nebo identifikací uživatele formou jména a SMS kódu. To záleží již na volané metodě:

a) volána metoda Run() či RunNonVis() či volána metoda SchrOper() s použitím certifikátu

V případě volání metody Run() či RunNonVis(), případně při volání metody SchrOper() s použitím certifikátu musí klient podepsat XML výzvu certifikátem, který má zaveden v Portálu ZP u svého konta. XML výzva dále obsahuje přihlašovací řetězec, který je součástí předávaných dat a podepsáním celé XML výzvy dává klient souhlas k přihlášení se k Portálu ZP za účelem předání požadovaného podání (souborů).

Pokud operační systém podporuje hashovací funkci SHA-256, je podpis vytvořen s SHA-256 miniaturou, v opačném případě s SHA-1 miniaturou. Pokud je použita SHA-1, je tato informace vrácena ve výstupním parametru Debug_internal_warning. Hashovací funkce SHA-2 rodiny (SHA-224, SHA-256, SHA-384 a SHA-512) jsou dostupné pouze na operačních systémech Windows XP SP3, Windows Vista, Windows Server 2008 SP1 a Windows 7.



Pokud podepsání proběhne v pořádku (uživatel nestornuje podepsání), je celá XML výzva, která se posílá na server, uložena ve výstupním parametru `Debug_XML_vyzva`.

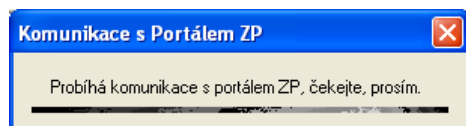
b) volána metoda `SMSRun()` či `SMSRunNonVis` či volána metoda `SchrOper()` s použitím SMS

V případě volání metody `SMSRun()` či `SMSRunNonVis()`, případně při volání metody `SchrOper()` s použitím SMS již musí být klient přihlášen k Portálu ZP a musí mít platný SMS kód. Pokud ještě není klient přihlášen (nemá SMS kód), je nutné jej přihlásit voláním metody **`SMSAuth()` nebo `SMSAuthNonVis()`**, kde předá svoje přihlašovací jméno a heslo a po úspěšném ověření mu bude zaslána SMS zpráva s ověřovacím SMS kódem. Tento SMS kód a své přihlašovací jméno potom předá společně s ostatními vstupními parametry metodě **`SMSRun()` nebo `SMSRunNonVis()` nebo `SchrOper()`**, která je přidá za XML výzvu sestavenou ze vstupních parametrů.

7.2 Odeslání XML výzvy na klientskou bránu ZP

Po sestavení a případném podepsání XML výzvy je tato výzva odeslána na klientskou bránu zdravotní pojišťovny uvedené ve vstupním parametru `ZP`. Jedná se o anonymní HTTPS komunikaci, server nepožaduje ověření klienta certifikátem.

V případě volání `Run`, `SMSRun` nebo `SMSAuth` metod, tedy metod bez postfixu `NonVis` je celou dobu zobrazeno následující informační okno:



V případě volání `*NonVis` metod okno zobrazeno není a volající aplikace by měla zobrazovat vlastní informační okno.

Celá komunikace je také časově omezena počtem sekund uvedeným v parametru `TimeOut`. Toto omezení je nastavitelné a pokud není uvedeno (hodnota 0) je `TimeOut` přednastaven na 300 sekund. Pokud komunikace z nějakého důvodu nedoběhne do tohoto časového limitu, komunikace končí chybou a podání není doručeno do ZP. Odpočítávání timeoutu probíhá ihned od korektního podepsání dat klientem, kdy začíná samotná HTTPS komunikace. Délku timeoutu je dobré volit s ohledem na velikost přenášených dat. Defaultních 300 sekund stačí na bezproblémový přenos do řádu několika stovek KB dat. Pokud bude klient přenášet několika MB podání, je třeba úměrně prodloužit timeout, aby bylo možné objem dat na portál ZP přenést.

7.3 Přijetí XML odpovědi a její zpracování

Po skončení komunikace očekává knihovna na výstupu odpověď ve formě XML a pokud ji neobdrží, například z důvodu stornování podpisu XML výzvy, potom je na výstupu textové chybové hlášení. Obecně ať komunikace skončí jakkoliv, tak jsou naplněny výstupní parametry `ID_Podani`, `Chyba`, `Varovani`, `DataOut`, `Debug_internal_warning`, `Debug_XML_vyzva` a `Debug_XML_odpoved`.

7.3.1 Komunikace skončila chybou

V případě, že se vyskytne jakákoliv chyba, která způsobí, že komunikace s portálem ZP neproběhne korektně (uživatelské storno, chyba síťového připojení, timeout apod.) budou výstupní parametry naplněny takto:

<code>ID_podani</code>	0
<code>Chyba</code>	řetězec s číselnou hodnotou chyby od -1001 do -1099.
<code>Varovani</code>	prázdný řetězec
<code>DataOut</code>	chybové hlášení nebo HTML chybová stránka
<code>Debug_internal_warning</code>	prázdný řetězec
<code>Debug_XML_vyzva</code>	sestavené XML výzvy nebo prázdný řetězec v případě storna při podepisování vstupních dat
<code>Debug_XML_odpoved</code>	prázdný řetězec nebo chybové hlášení

7.3.2 Komunikace skončila v pořádku, zpracování na serveru vrátilo chybu

V případě, že komunikace proběhne v pořádku a bude vrácena XML odpověď s chybovým stavem budou výstupní parametry naplněny takto:

ID_podani	číslo uloženého podání
Chyba	řetězec určující chybu při zpracování na serveru
Varovani	prázdný řetězec nebo případné varování pro klienta – viz. 7.3.4
DataOut	chybové hlášení nebo výsledný HTML protokol s chybovým hlášením
Debug_internal_warning	prázdný řetězec nebo interní varování knihovny určené pro výrobce SW – viz. 7.3.4
Debug_XML_vyzva	sestavené XML výzvy
Debug_XML_odpoved	vrácená XML odpověď

I když zpracování na serveru skončilo chybou, parametr ID_Podani obsahuje číslo podání, protože podání bylo uloženo na serveru z důvodu případného ručního dohledání. Toto podání nebude předáno dále do pojišťovny.

7.3.3 Komunikace i zpracování na serveru skončilo v pořádku

V případě, že komunikace proběhne v pořádku a vstupní data budou na serveru korektně zpracovány, tj. nevyskytne se při zpracování žádná chyba, budou výstupní parametry naplněny takto:

ID_podani	číslo uloženého podání
Chyba	"0"
Varovani	prázdný řetězec nebo případné varování pro klienta – viz. 7.3.4
DataOut	výsledný HTML protokol
Debug_internal_warning	prázdný řetězec nebo interní varování knihovny určené pro výrobce SW – viz. 7.3.4
Debug_XML_vyzva	sestavené XML výzvy
Debug_XML_odpoved	vrácená XML odpověď

7.3.4 Ověření podpisu XML odpovědi

Pokud dojde k validní komunikaci a server vrátí XML odpověď (bod 7.3.2 a 7.3.3), XML odpověď je podepsána certifikátem, aby byla zaručena důvěryhodnost vrácených dat. Tento podpis knihovna ověří a stav ověření se nachází ve výstupních atributech **Varovani** a **Debug_internal_warning**. Ověření podpisu a následné plnění výstupních parametrů je ovlivněno vstupními/výstupními parametry **ZP_Cert_SN** a **ZP_Cert_IDN**.

Popis ověření podpisu:

Knihovna ověří podpis přijatých dat proti certifikátu serveru, s kterým proběhla komunikace. Tento certifikát automaticky získá stažením ze serveru, s kterým komunikuje, z jeho https adresy https://.../portal_sign.pem.

Poté zkontroluje, zda sériové číslo certifikátu je shodné s parametrem **ZP_Cert_SN** a zda vystavitel certifikátu je shodný s parametrem **ZP_Cert_IDN**. Dále plní parametry **ZP_Cert_SN** a **ZP_Cert_IDN** údaji, které byly zjištěny z certifikátu serveru, proti kterému proběhlo ověření podpisu.

Během tohoto ověřování se zjištěnými nesrovnalostmi nebo informacemi plní výstupní parametry **Varovani** a **Debug_internal_warning**.

Ideální stav nastává tehdy, když jsou oba tyto parametry prázdné – nejsou žádná varování, ať interní nebo pro klienta.

Možné hodnoty parametrů a odstranění jejich příčin:

1.	Varovani = 'Nepodařilo se ověřit podpis serveru:' + chybové hlášení
<p>Kontrola podpisu proti přijatým datům neproběhla korektně. Data byla zřejmě změněna nebo není k dispozici správný certifikát.</p> <p>Odstranění varování: Kontaktujte administrátory portálu na pzp.admin@asseco-ce.com a uveďte číslo podání a název zdravotní pojišťovny, které se podání týkalo pro vyřešení problému. Toto chybové hlášení by se nemělo vyskytnout.</p>	
2.	Varovani = 'Kontrola podpisu odpovědi provedena proti tomuto certifikátu staženého z Portálu <ZP>: SN: <SN1> , IDN: <IDN1> . Byl však očekáván certifikát s touto identifikací:

	SN: <SN2> , IDN: <IDN2> . Možná byl vydán nový certifikát serveru.'
	<p>Kontrola podpisu proti přijatým datům proběhla korektně, ale certifikát, proti kterému bylo ověření provedeno je jiný než požadovaný certifikát.</p> <p>Aplikace využívající DLL knihovnu požadovala ověření podpisu proti certifikátu se SN a IDN předanými v parametrech ZP_Cert_SN a ZP_Cert_IDN, ale ověření proběhlo proti certifikátu se <SN1> a <IDN1> (kde <SN1> a <IDN1> jsou konkrétní hodnoty čtené z certifikátu serveru).</p> <p>Odstranění varování:</p> <p>Zřejmě se změnil certifikát serveru, při příští komunikaci můžete použít SN a IDN vrácené v parametrech ZP_Cert_SN a ZP_Cert_IDN, kde po skončení komunikace jsou uloženy hodnoty čtené z certifikátu serveru. Je však vhodné si nezávisle ověřit, zda tyto údaje odpovídají certifikátu, který byl danému portálu opravdu vydán.</p>
3.	<p>Debug_internal_warning = 'Kontrola podpisu odpovědi provedena proti tomuto certifikátu staženého z Portálu <ZP>: SN: <SN> , IDN: <IDN> . Obecně se nemusí jednat o certifikát Portálu <ZP>'</p> <p>Kontrola podpisu proti přijatým datům proběhla korektně, parametr Debug_internal_warning nese informaci o použitém certifikátu, proti kterému bylo ověření provedeno.</p> <p>Odstranění varování:</p> <p>Při příští komunikaci můžete naplnit parametry ZP_Cert_SN a ZP_Cert_IDN hodnotami vrácenými v parametrech ZP_Cert_SN a ZP_Cert_IDN po skončení komunikace. Tyto hodnoty jsou čtené z certifikátu serveru. Je však vhodné si nezávisle ověřit, zda tyto údaje odpovídají certifikátu, který byl danému portálu opravdu vydán.</p>
4.	<p>Debug_internal_warning = 'Windows CryptoAPI stávajícího operačního systému neumožňuje při podepisování dat použít hashovací funkci SHA-256. Byla proto použita SHA-1.'</p> <p>Při vytváření podpisu dat v XML dokumentu je vypočítáván hash (miniatura) dat pomocí hashovací funkce SHA-256. K tomuto účelu je využito CryptoAPI funkcí systému Windows a pouze novější verze operačních systémů podporují funkce rodiny SHA-2, kam zmíněná SHA-256 patří. Pokud operační systém funkci SHA-256 nepodporuje, je v parametru Debug_internal_warning vráceno toto varování a k výpočtu použita starší funkce SHA-1.</p> <p>Odstranění varování:</p> <p>Použít novější operační systém nebo nainstalovat service pack či aktualizaci, která tyto funkce do systému přidá – je-li pro aktuálně používaný systém k dispozici. Operační systémy podporující funkce SHA-2: Windows XP SP3, Windows Vista, Windows Server 2008 SP1 a Windows 7</p>
5.	<p>Debug_internal_warning = ' Podpis není validní, nelze z něho získat SN a IDN!'</p> <p>Vrácené XML zřejmě neobsahuje podpis dat nebo je podpis porušen. Pravděpodobně se klient přihlásil certifikátem, který nemá zaveden u svého klientského konta v portálu ZP a v tomto případě je komunikace odmítnuta na nejnižší úrovni a není podepsána. V tomto případě by parametr Chyba nikdy neměl obsahovat hodnotu "0"</p> <p>Odstranění varování:</p> <p>Nelze odstranit</p>

8. Použití knihovny ve vývojovém prostředí

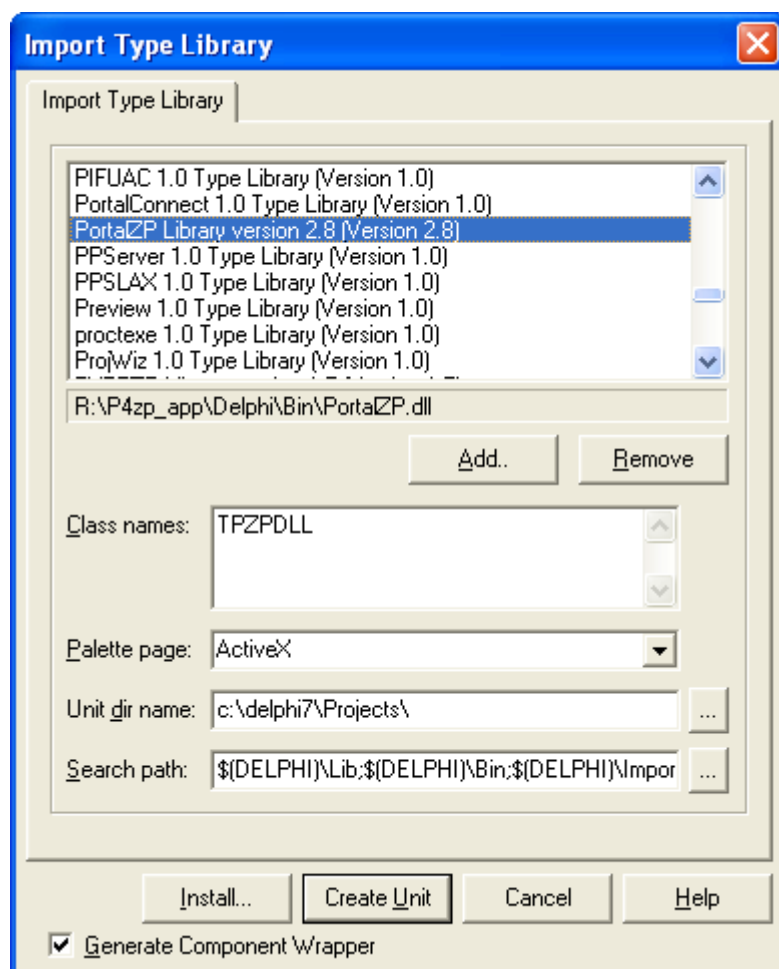
Knihovna je použitelná v jakémkoliv vývojovém nástroji, který podporuje technologii COM (Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual C#, Active Server Pages, Delphi atd.) nebo umožňuje importovat a volat funkce z DLL knihovny.

Jakmile je knihovna nainstalována (registrována) v systému, vývojové nástroje nemají problémy ji naimportovat (jako Type Library) a používat.

8.1 Návod pro importování COM komponenty do Delphi

Před samotným importem COM komponenty je nutné, aby DLL knihovna byla již zaregistrována v systému – viz. kapitola 3.1.

Naimportování komponenty do vývojového prostředí Borland (například Delphi nebo C++ Builder) provedete pomocí menu "Project | Import Type Library". Je zobrazeno následující okno:



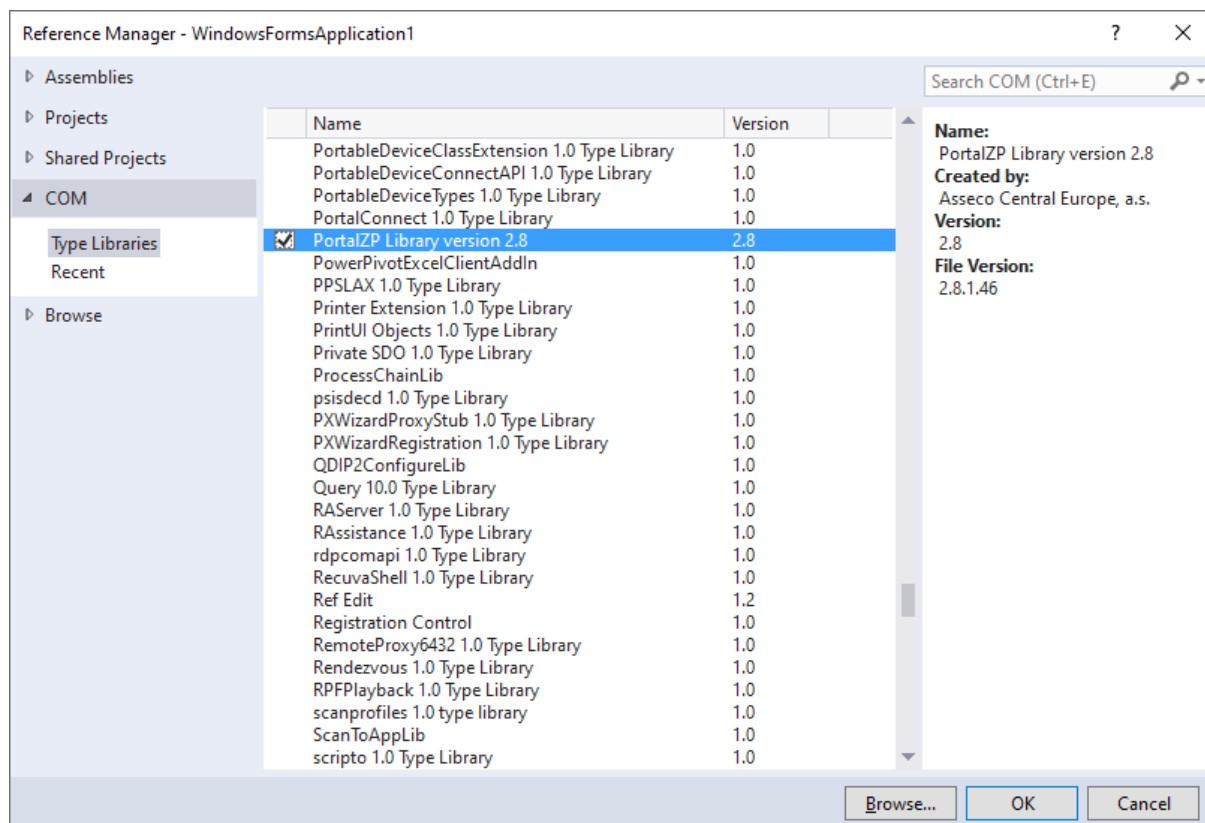
Vybere se položka "PortalZP Library version 2.8", která obsahuje třídu PZPDLL a buď se nainstaluje na lištu s komponentami pomocí tlačítka "Install" nebo jen se nechá vytvořit zdrojové soubory pomocí tlačítka "Create Unit". Soubory se ukládají v obou případech do adresáře "Unit dir name".

Po zaregistrování třídy PZPDLL se již s touto třídou pracuje naprosto běžně jako s jinou třídou.

8.2 Návod pro importování COM komponenty do Visual Studia

Před samotným importem COM komponenty je nutné, aby DLL knihovna byla již zaregistrována v systému – viz. kapitola 3.1.

Naimportování komponenty do aktuálního projektu Visual Studia provedete pomocí hlavního menu „PROJECT“ => „Add reference ...“. Je zobrazeno následující okno ve kterém se přepnete na záložku „COM“:



Dohleďte záznam „PortalZP library version ...“, zaškrtněte jej a pomocí tlačítka OK bude přidána reference na COM objekt do projektu.

Pokud chcete používat knihovnu v .NET projektech, můžete využít verzi knihovny určenou přímo pro .NET projekty, která je nabízena v balíčku pod jménem knihovny PortalZPCsNet.

9. Příklady použití třídy PZPDLL

9.1 COM verze v Borland Delphi 7

Následující zdrojový kód ukazuje základní použití třídy PZPDLL v Borland Delphi 7, kdy byla naimportována jako třída TPZPDLL v případě přístupu přes COM (viz. kapitola 8.1):

```
type
  SWInfoRec = packed record
    Nazev: WideString;
    Verze: WideString;
    Vyrobce: WideString;
    Helpline: WideString;
  end;

procedure TForm1.Button1Click(Sender: TObject);
var PZPDLL: TPZPDLL;
    ID_Podani: Integer;
    ZP, ZP_Cert_SN, ZP_Cert_IDN, Sluzba, DataIn, Chyba, Varovani, DataOut,
    XML_IN, XML_OUT, InternalWarning: WideString;
    Timeout: Smallint;
    SWInfo: SWInfoRec;

const cZP: array[0..7] of WideString = ('CPZP', 'OZP', 'RBP', 'ZPS', 'VoZP',
    'PZP', 'PILOT', 'PILOT-PZP');
const cSluzba: array[0..13] of WideString = ('VYU', 'HOZ', 'PPPZ', 'VERPOJ',
    'VERZZ', 'ORL', 'PRESMER', 'SCHRANKA', 'EXE_PO', 'EXE_FO', 'P2_IMPORT',
```

```
'NOVY_CERT', 'SCHRANKA_PODANI', 'BZD', 'HDN');

begin
  try
    PZPDLL:=TPZPDLL.Create(Self);
    if PZPDLL.Init<>S_OK then
      EAbort.Create('ERROR');
    except
      Application.MessageBox(PChar('Pravdepodobně nemáte nainstalovanou
knihovnu PortalZP.DLL do systému! '#13'Instalaci provedete pomocí příkazu
regsvr32 (a plnou cestou ke knihovně PortalZP.DLL): '#13'regsvr32
PortalZP.DLL'), PChar('Chyba'), MB_OK or MB_ICONERROR);
      exit;
    end;
  try
    // Vstupni data
    SWInfo.Nazev:='PortalZP_Demo';
    SWInfo.Verze:='2.7.2.45';
    SWInfo.Vyrobce:='Asseco Central Europe, a.s.';
    SWInfo.Helpline:='pomoc@portalzp.cz';
    ZP:=cZP[0]; // CPZP
    Sluzba:=cSluzba[0]; // VYU
    DataIn:='C:\Soubor1.dat|C:\Soubor2.dat';
    ZP_Cert_SN:='90A31D91402A5AEE7D0A5D0DC7AD027';
    ZP_Cert_IDN:='C=US, O=DigiCert Inc, OU=www.digicert.com, CN=RapidSSL
RSA CA 2018';
    Timeout:=300;
    // Volani knihovny
    PZPDLL.Run(ZP, ZP_Cert_SN, ZP_Cert_IDN, Sluzba, DataIn, Timeout,
SWInfo, ID_Podani, Chyba, Varovani, DataOut, InternalWarning, XML_In,
XML_Out);
    // Vystup
    If Chyba='0' then
      begin
        ShowMessage('OK. Data mam ulozena v DataOut');
      end else
      begin
        ShowMessage('Nastala chyba '+Chyba+'. Její text je uložen v
DataOut');
      end;
    finally
      PZPDLL.Free;
    end;
  end;
end;
```

9.2 DLL verze v Borland Delphi 7

Následující zdrojový kód ukazuje základní použití s importovanou DLL funkcí v Borland Delphi 7. V tomto případě nemusí být DLL knihovna zaregistrována v systému a ani nainportována do Delphi, stačí pouze, aby DLL knihovna byla dostupná výsledné binární aplikaci, tj. všechny potřebné DLL byly ve stejném adresáři nebo v adresáři dostupném pomocí PATH systémové proměnné.

```
type
  SWInfoRec = packed record
    Nazev: WideString;
    Verze: WideString;
    Vyrobce: WideString;
    Helpline: WideString;
  end;
```

```
function DllPZP_Run(const ZP: WideString; var ZP_Cert_SN, ZP_Cert_IDN:
WideString;
    const Sluzba, DataIn: WideString;
    Timeout: Smallint; SWInfo: SWInfoRec;
    var ID_Podani: SYSINT; var Chyba, Varovani, DataOut,
    Debug_internal_warning, Debug_XML_vyzva,
    Debug_XML_odpoved: WideString): HRESULT; stdcall;

function DllPZP_Run; external 'PortalZP.dll' name 'DllPZP_Run';

procedure TForm1.Button1Click(Sender: TObject);
Var ID_Podani:Integer;
    ZP, ZP_Cert_SN, ZP_Cert_IDN, Sluzba, DataIn, Chyba, Varovani, DataOut,
XML_IN, XML_OUT, InternalWarning: WideString;
    Timeout:Smallint;

Const cZP:array[0..7] of WideString = ('CPZP', 'OZP', 'RBP', 'ZPS', 'VoZP',
'PZP', 'PILOT', 'PILOT-PZP');
Const cSluzba:array[0..13] of WideString = ('VYU', 'HOZ', 'PPPZ', 'VERPOJ',
'VEZZ', 'ORL', 'PRESMER', 'SCHRANKA', 'EXE_PO', 'EXE_FO', 'P2_IMPORT',
'NOVY_CERT','SCHRANKA_PODANI','BZD','HDN');

begin
    // Vstupni data
    SWInfo.Nazev:='PortalZP_Demo';
    SWInfo.Verze:='2.7.2.45';
    SWInfo.Vyrobc:='Asseco Central Europe, a.s.';
    SWInfo.Helpline:='pomoc@portalzp.cz';
    ZP:=cZP[0]; // CPZP
    Sluzba:=cSluzba[0]; // VYU
    DataIn:='C:\Soubor1.dat|C:\Soubor2.dat';
    ZP_Cert_SN:='90A31D91402A5AEE7D0A5D0DC7AD027';
    ZP_Cert_IDN:='C=US, O=DigiCert Inc, OU=www.digicert.com, CN=RapidSSL RSA
CA 2018';
    Timeout:=300;
    // Volani funkce DLL knihovny
    DllPZP_Run(ZP, ZP_Cert_SN, ZP_Cert_IDN, Sluzba, DataIn, Timeout, SWInfo,
ID_Podani, Chyba, Varovani, DataOut, InternalWarning, XML_In, XML_Out);
    // Vystup
    If Chyba='0' then
    begin
        ShowMessage('OK. Data mam ulozena v DataOut');
    end else
    begin
        ShowMessage('Nastala chyba '+Chyba+'. Její text je uložen v DataOut');
    end;
end;
```

9.3 COM verze ve Visual Studiu a C# Windows Forms Application projektu

Následující zdrojový kód ukazuje základní použití třídy PZPDLL ve Visual Studiu, kdy byla do projektu naimportována pomocí návodu v kapitole 8.2:

```
using PortalZP_Library;

private void button1_Click(object sender, EventArgs e)
{
    PZPDLL MyKOM;
    int ID_Podani;
    string Chyba, Varovani, DataOut, DBG_Warn, XMLIn, XMLOut;
    SWInfoRec SWInfo = new SWInfoRec() {
```

```
Nazev = "PortalZP_Demo",  
Verze = "2.7.2.45",  
Vyrobce = "Asseco Central Europe, a.s.",  
Helpline = "pomoc@portalzp.cz"  
};  
ID_Podani = 0;  
Chyba = Varovani = DataOut = DBG_Warn = XMLIn = XMLOut = "";  
MyKOM = new PZPDLL();  
MyKOM.Run("PILOT", "", "", "VYU", "", 300, SWInfo, ref ID_Podani, ref  
Chyba, ref Varovani, ref DataOut, ref DBG_Warn, ref XMLIn, ref XMLOut);  
MessageBox.Show(DataOut);  
MyKOM = null;  
}
```

10. Chybové stavy

Chybové stavy jsou rozděleny do dvou skupin.

Jednu skupinu chyb vrací knihovna a jsou to chyby, které se staly při kontrole vstupních dat, podepisování či přímo při komunikaci s portálem ZP. Druhá skupina chyb jsou chyby, které vrací portál ZP při zpracování požadavku.

Popis chyb:

-1001	Neznámá zkratka ZP. ZP může být pouze jedna z hodnot CPZP, OZP, RBP, ZPS, VoZP, PZP, PILOT, PILOT-PZP
-1002	Neznámá služba. Služba může být pouze jedna z hodnot VYU, HOZ, PPPZ, VERPOJ, VERZZ, ORL, PRESMER, SCHRANKA, EXE_PO, EXE_FO, P2_IMPORT, NOVY_CERT, SCHRANKA_PODANI, BZD, HDN
-1003	Nenalezen předaný soubor ... Soubor předaný v parametru DataIn nebyl nalezen.
-1004	Vypršel časový limit XXX sekund.
-1005	Nastala chyba podepisovací komponenty Signer.
-1006	Nebylo přijato validní XML, obsahem výstupních parametrů DataOut a Debug_XML_odpoved je blíže neurčená chyba (může být HTML např. o odstávce portálu, prostý text nebo nevalidní XML)
-1007	Chyba XML parseru při zpracovávání odpovědi
-1010	HTTP kód odpovědi nebyl 200, nejednalo se o validní odpověď XML brány
-1011	Server Portálu ZP není dostupný
-1020	Komunikační vlákno nebylo vytvořeno
-1090	Interní chyba knihovny #1
-1099	Interní chyba knihovny #2
-1031	Neznámá operace schránek. Operace musí být jedna z hodnot SCHRANKY, ZPRAVY_VSE, ZPRAVY_NOVE, ZPRAVA_DETAIL, ZPRAVA_PRECTI, ZPRAVA_SMAZ
-1032	Byla použita operace schránek SCHRANKY a byl předán seznam identifikátorů. Pro tuto operaci musí být seznam identifikátorů prázdný.
-1033	Metodě SchrOper byl předán prázdný seznam identifikátorů. Všechny operace kromě operace SCHRANKY vyžadují předat seznam identifikátorů ať ve formě jednoho ID nebo seznamu ID oddělených znakem " "
-1034	Metodě SchrOper byl předán nevalidní seznam identifikátorů. Jedná se o operaci schránek ZPRAVA_DETAIL, ZPRAVA_PRECTI nebo ZPRAVA_SMAZ. Tyto operace očekávají v seznamu identifikátorů předané číselné identifikátory oddělené znakem " ". Zřejmě byl předán nečíselný identifikátor, který je přípustný pouze u operace ZPRAVY_VSE nebo ZPRAVY_NOVE.
0	Zpracování skončilo v pořádku, výstupní data byla zpracována a uložena do výstupních parametrů. Podání je uloženo na serveru a v parametru ID_Podani je jeho číslo.
ostatní	Chybové stavy při zpracování podání na portálu ZP – validní odpověď stejně jako při nulové chybě, akorát nedojde ke správnému zpracování podání vlivem nesrovnalostí údajů ve vstupních datech (není oprávnění, chybná struktura dat, nebyly předány všechny informace, chyba kontrol vstupních dat apod.). Chyba není pouze číselného charakteru ale jedná se o jakýkoliv alfanumerický kód jiný než v předchozích případech.

11. Debug výstup knihovny

Knihovna od verze 2.6.2.39 obsahuje podporu pro logování běhu při nutnosti detekování problémů na straně klienta. Logování výstupu se provádí vytvořením adresáře c:\PortalZP_Log s RW právy. Pokud knihovna nalezne adresář c:\PortalZP_Log, vytvoří do něj soubor PzpDll.log v Win1250 kódování, kam bude ukládat logy o průběhu komunikace. Tento logový soubor lze použít jako podklad pro hledání případných problémů.