www.areality.sk

RealitySKWS_1 - rozhraní pro přenos zakázek

Technická dokumentace
Verze 1.1
16.9.2010 – Ing. Jiří Fořt, Diadema Software s.r.o.

Rozhraní www.areality.sk pro přenos zakázek je založeno na standardu Web Services, pro volání vzdálených metod je použit standard SOAP nad protokolem HTTP.

Formální specifikace rozhraní ve standardu WSDL je k dispozici na URL http://www.areality.sk/WS/RealitySKWS_1.asmx?wsdl

Rozhraní se sestává z následujících metod (prototypy metod jsou zapsány v syntaxi jazyka C#). Většina metod je určena pro dávkové zpracování, je nutné je volat 1x v rámci relace (!).

`string About()`

- popis
  - vrací informace o serverovém software
- vstup
  -
- výstup
  - řetězec obsahující popis aktuální verze serverového software a údaje o copyrightu
- počet volání
  - libovolný
- přihlášení
  - není nutné
- poznámka
  - lze použít pro jednoduchý test spojení se serverem (ping)

`byte[] PreLogin (string UserAgent)`

- popis
  - generuje autorizační token pro přihlášení
- vstup
  - **`string UserAgent`** – identifikátor klientské aplikace (každému tvůrci je provozovatelem přiděleno jednoznačné ID)
- výstup
  - pole byte s autorizačním tokenem (40B)
- počet volání
  - libovolný
- přihlášení
  - uživatel nesmí být přihlášen
- poznámka

- detailní popis přihlašovacího algoritmu viz příloha

`bool Login (string LoginName, byte[] Password, byte[] AgentAuthCode)`

- popis
  - přihlásí uživatele
- vstup
  - **`string LoginName`** – přihlašovací jméno RK do systému areality.sk
  - **`byte[] Password`** – autorizačním tokenem zašifrované heslo
  - **`byte[] AgentAuthCode`** – autorizačním tokenem zašifrovaný agent
- výstup
  - true při úspěšném přihlášení, false při chybném jménu/heslu/AgentAuthCode
- počet volání
  - libovolný
- přihlášení
  - uživatel nesmí být přihlášen
- poznámka
  - detailní popis přihlašovacího algoritmu viz příloha

`void Logout()`

- popis
  - odhlásí přihlášeného uživatele
- vstup
  -
- výstup
  - true při úspěšném přihlášení, false při chybném jménu/heslu/AgentAuthCode
- počet volání
  - 1x
- přihlášení
  - nutné

`TRealitySKZakazkaStav_1[] CtiStavZakazek ()`

- zjišťuje seznam a stav zakázek včetně stavu obrázků k jednotlivým zakázkám přihlášené RK
- vstup
  -
- výstup
  - pole informačních záznamů `TRealitySKZakazkaStav_1`. Pro každou existující zakázku 1 prvek pole.
- počet volání
  - 1x
- přihlášení
  - nutné
- poznámka
  - detailní popis záznamu `TRealitySKZakazkaStav_1` viz příloha

`TRealitySKRemotableRKInfo_1[] CtiRK (string[] IDRK)`

- dávkově čte informace o realitních kancelářích
- vstup
  - **`string[] IDRK`** - pole IDRK o kterých mají být vráceny informace
    - pokud není parametr předán, vrátí aktuálně přihlášenou RK. **To je po přihlášení velmi vhodné pro zjištění IDRK, které je nutné předávat dalším funkcím.**
- výstup
  - pole informačních záznamů `TRealitySKRemotableRKInfo_1`. Pro každou čtenou realitní kancelář je vrácen 1 prvek pole pozičně odpovídající předanému IDRK
- počet volání
  - 1x – dávková operace
- přihlášení
  - nutné
- poznámka
  - detailní popis záznamu `TRealitySKRemotableRKInfo_1` viz příloha

`TRealitySKRemotableZakazkaData_1[] CtiZakazky (string[] IDZak)`

- dávkově čte informace o zakázkách
- vstup
  - **`string[] IDZak`** - pole IDZak, které mají být čteny
- výstup
  - pole informačních záznamů `TRealitySKRemotableZakazkaData_1`. Pro každou čtenou zakázku je vrácen 1 prvek pole pozičně odpovídající předanému IDZak
- počet volání
  - 1x
- přihlášení
  - nutné
- poznámka
  - detailní popis záznamu `TRealitySKRemotableZakazkaData_1` viz příloha

`public TRealitySKOpResult_1[] ZapisZakazky (TRealitySKRemotableZakazkaData_1[] ZakazkyZapis)`

- dávkový zápis / aktualizace zakázek
- vstup
  - **`ZakazkyZapis`** – pole zakázek, které které mají být zapsány. Pokud položka `TRealitySKRemotableZakazkaData_1.IDZak` není vyplněna, bude přidána nová zakázka, jinak bude příslušná zakázka přepsána. **Doporučená maximální velikost pole je 10 zakázek.** Vzhledem k tomu, že zápis může nějakou dobu trvat, třeba při volání mít dostatečně nastaven timeout, případně předávat méně nemovitostí v jednom volání.
- výstup
  - pole informačních záznamů `TRealitySKOpResult_1`, indikující výsledek zápisu jednotlivé zakázky. Pro každou zapisovanou zakázku je vrácen 1 prvek pole

pozičně odpovídající předanému záznamu zakázky pro zápis. Nově přidané zakázky mají vyplněno pole `TRealitySKOpResult_1.IDZak` na hodnotu, kterou jim přidělil server.

- počet volání
  - libovolný

- přihlášení
  - nutné
- poznámka
  - detailní popis `TRealitySKRemotableZakazkaData_1` viz příloha
  - zakázky starší více než 24 hodin je možné modifikovat maximálně jednou za 2 hodiny. Při porušení tohoto mezení nebude změna zakázky zapsána a zpracování bude penalizováno 10ti vteřinovou prodlevou.

`public int ZapisObrazek(string IDZak, string Popis, byte[] data)`

- přidá jeden obrázek k existující zakázce
- vstup
  - `string IDZak` – identifikace zakázky, ke které má být přidán obrázek
  - `string Popis` – volitelný popis obrázku
  - `byte[] data` – vlastní data obrázku ve formátu JPEG, max. velikost 200KB.
- výstup
  - ID obrázku, které přidělil server.
- počet volání
  - libovolný
- přihlášení
  - nutné

`public TRealitySKOpResult_1[] VymazZakazky (string[] IDZak)`

- dávkově vymazává existující zakázky
- vstup
  - **`string[] IDZak`** - pole IDZak, které mají být vymazány
- výstup
  - pole informačních záznamů `TRealitySKOpResult_1`, indikující výsledek mazání jednotlivé zakázky. Pro každou mazanou zakázku je vrácen 1 prvek pole pozičně odpovídající předanému IDZak.
- počet volání
  - libovolný
- přihlášení
  - nutné

`public TRealitySKOpResult_1[] VymazObrazky (int[] IDObrazek)`

- dávkově vymazává existující obrázky
- vstup
  - **`int[] IDObrazek`** - pole IDObrazek, které mají být vymazány
- výstup

    - pole informačních záznamů `TRealitySKOpResult_1`, indikující výsledek mazání jednotlivého obrázku. Pro každý mazaný obrázek je vrácen 1 prvek pole pozičně odpovídající předanému IDObrazek.
- počet volání
    - Nx
- přihlášení
    - nutné

Datové struktury

```
public class TRealitySKOpResult_1
{
      public bool OK;               // výsledek operace
      public string ErrDesc;        // popis chyby
      public string ID;             // ID (zakázky, obrázku, ...)
}

public class TRealitySKZakazkaStav_1
{
      public string IDZak;          // IDZak
      public int[] IDObrazek;       // seznam obrázků k zakázce
}

public class TRealitySKRemotableZakazkaData_1
      {
      public string IDZak;
      public string OnLine;         // rezervováno, false
      public string IDRK;           // ID realitní kanceláře
      public string Slogan;
      public string IDReferent;     // ID referenta
      public string UzivIDZak;      // uživatelské (volitelné) ID zak.
      public string IDDruh;
      public string IDTyp;
      public string Velikost;       // FLOAT: velikost (m2)
      public string Cena;           // FLOAT: cena
      public string IDJednotkaCena;
      public string PoznamkaCena;
      public string IDStat;
      public string IDKraj;
      public string IDOkres;
      public string IDObec;
      public string IDKatUzemi;
      public string Lokalita;
      public string Popis;
      public string [][] Charakteristiky;      // dvojice {IDCharakteristika,
                                                  IDHodnota nebo hodnota}
      public string IDStav;
      public string KDispoziciOd;  //DATE
      public string ProdanoDne;    //DATE
      public string ZrusitDne;     //DATE
      public string ProdanoZaCenu; //INT
      public bool   Exkluzivni;
      public bool   KeSpolupraci;
      public string URL;
      public string DrazbaMisto;
      public string DrazbaDatum;
      public string DrazbaPredmet;
      public string DrazbaJistina;
      public bool   DrazbaDobrovolna;
      public string DrazbaZverejnitDne; //DATE
      public string VlozenoDne;     //DATE
}

Povinná pole pro zápis jsou podtržena.
FLOAT - string konvertovatelný na float (povolena desetinná tečka)
DATE  - string ve tvaru DD.MM.YYYY
INT   - string konvertovatelný na int (bez desetinné tečky)
```

```
public class TRealitySKRemotableRKInfo_1
      {
            public TRealitySKRemotableRealitniKancelar_1 RK;
            public TRealitySKRemotableReferent_1[] Referenti;
      }

      public class TRealitySKRemotableRealitniKancelar_1
      {
            public string IDRK;
            public string Nazev;
            public string Ulice;
            public string Mesto;
            public string IDOkres;
            public string PSC;
            public string Tel1;
            public string Tel2;
            public string Fax;
            public string WWW;
            public string EMail;
            public string Kontakt;
            public bool   ARKCMS;
            public string ICO;
            public string DIC;
      }

      public class TRealitySKRemotableReferent_1
      {
            public string IDReferent;
            public string Prijmeni;
            public string Jmeno;
            public string Titul;
            public string EMail;
            public string Tel;
            public string Fax;
      }
```

www.areality.sk

RealitySKWS_2 - rozhraní pro přenos zakázek

Technická dokumentace
Verze 1.0

Rozhraní www.areality.sk pro přenos zakázek je založeno na standardu Web Services, pro volání vzdálených metod je použit standard SOAP nad protokolem HTTP.

Formální specifikace rozhraní ve standardu WSDL je k dispozici na URL http://www.areality.sk/WS/RealitySKWS_2.asmx?wsdl

Rozhraní se sestává ze stejných metod jako rozhraní RealitySKWS_1. Přibyla jen jedna metoda, která bude popsaná dále.

`TRealitySKZakazkaStav_2[] CtiStavZakazek2 ()`

- funkce metody je stejná jako funkce metody **`CtiStavZakazek,`** jen je výstup doplněn o uživatelské číslo zakázky a o datum poslední změny.

Datové struktury

```
public class TRealitySKZakazkaStav_2 : TRealitySKZakazkaStav_1
{
      public string UzivIDZak;   //uživatelské číslo zakázky
      public string ZmenenoDne;  //datum poslední změny
}
```

www.areality.sk

RealitySKWS_3 - rozhraní pro přenos zakázek

Technická dokumentace
Verze 1.0

Rozhraní www.areality.sk pro přenos zakázek je založeno na standardu Web Services, pro volání vzdálených metod je použit standard SOAP nad protokolem HTTP.

Formální specifikace rozhraní ve standardu WSDL je k dispozici na URL http://www.areality.sk/WS/RealitySKWS_3.asmx?wsdl

Rozhraní se sestává ze stejných metod jako rozhraní RealitySKWS_2. Nové jsou metody pro správu makléřů a pro čtení statistik zakázek.

`public TRealitySKOpResult_1[] ZapisReferenty(TRealitySKRemotableReferent_1[] ReferentiZapis)`

- zapíše informace o makléřích
- vstup
  - **ReferentiZapis** – pole záznamů o makléřích
- výstup
  - pole informačních záznamů `TRealitySKOpResult_1`, indikující výsledek zápisu jednotlivé zakázky. Pro každou zapisovanou zakázku je vrácen 1 prvek pole pozičně odpovídající předanému záznamu zakázky pro zápis. Nově přidané položky mají vyplněno pole `TRealitySKOpResult_1.IDZak` na hodnotu, kterou jim přidělil server.
- počet volání
  - 1x
- přihlášení
  - nutné

`public TRealitySKOpResult_1[] VymazReferenty(string[] IDReferent)`

- zapíše informace o makléřích
- vstup
  - **IDReferent** – pole identifikátorů jednotlivých makléřů, které je třeba vymazat
- výstup
  - pole informačních záznamů `TRealitySKOpResult_1`
- počet volání
  - 1x
- přihlášení
  - nutné

`public TRealitySKZakazkaStatistikaResult_3[] CtiStatistiky(string[] IDZak)`

- čte statistiky jednotlivých zakázek. Jsou uchovávány statistiky pouze cca 60 dnů staré
- vstup
  - **IDZak** – pole čísel zakázek, které chcete přečíst
- výstup
  - pole `TRealitySKZakazkaStatistikaResult_3` informačních záznamů se statistikami
- počet volání
  - 1x
- přihlášení
  - nutné

Datové struktury

```
public class TRealitySKZakazkaStatistika_3
{
    public int Den;
    public int Mesic;
    public int Rok;
    public int Nahledy;
    public int Maily;
    public int Zajmy;
}

public class TRealitySKZakazkaStatistikaResult_3
{
    public string IDZak;
    public TRealitySKZakazkaStatistika_3[] Stat;
}
```

Algoritmus ověřování přihlašovacích údajů serveru www.areality.sk

1. Každý klientský SW má provozovatelem serveru přiděleno veřejné ID (UserAgent) a tajné heslo (AgentPassword).

2. Klient zavolá metodu "PreLogin(UserAgent)", která vrátí náhodně vygenerovaný autorizační token (Token) o délce 40b (délka může být změněna, je však vždy dělitelná číslem 4).

3. Klient vygeneruje autorizační kód (AgentAuthCode) a tento předá spolu s přihlašovací jménem a heslem při volání metody Login. Heslo je zakódováno stejným způsobem, jako se generuje AgentAuthCode. Pokud nesouhlasí AgentAuthCode, server generuje vyjímku "Invalid AgentAuthCode". Pokud je kód v pořádku server vrací true, když jsou předány správné přihlašovací údaje, jinak vrací hodnotu false.

Postup generování autorizačního kódu (AgentAuthCode) a kódování hesla

1. Heslo (nebo AgentPassword) je okopírováno cyklicky za sebe do pole o stejné délce, jako je velikost Tokenu (tj. hesloheslohesloheslo….). Jako Heslo se nyní bude uvažovat toto pole.

2. Na VŠECHNY jednotlivé byty Tokenu se aplikuje operace XOR tak, že se x-tý byte Tokenu XORuje x-tým byte Hesla.

3. Token se rozdělí na polovinu a na první polovinu Tokenu se aplikuje operace XOR tak, že se x-tý byte Tokenu XORuje x-tým bytem z druhé poloviny tokenu. Token se tak zkrátí na polovinu oproti původní délce („přeložení na ½“)

4. Token se opět rozdělí na polovinu a na první polovinu Tokenu se aplikuje operace XOR tak, že se x-tý byte Tokenu XORuje x-tým bytem z druhé poloviny tokenu. Token se tak zkrátí na čtvrtinu oproti původní délce (další „přeložení na ½")

5. Tento Token je požadovaný AgentAuthCode.

01234567890123456789012345678901234567890123456789 -> 01234567890123456789 -> 0123456789

Vývoj a testování aplikace pro přenos dat na server

1. vývojář aplikace kontaktuje správce serveru na adrese (info@diadema.cz) a vyžádá si přidělení testovacího účtu. Pro vytvoření je třeba sdělit správci název SW, který bude export používat.
2. správce serveru vytvoří
    a. UserAgentID + heslo
    b. email a heslo pro přihlášení k testovacímu účtu
3. testování aplikace se provádí na serveru devel.areality.sk. Je možné použít WSDL http://devel.areality.sk/WS/RealitySKWS_3.asmx?wsdl. Tento server je vyhrazen pouze pro účely testování. Je možné zde libovolně testovat. RK nebude účtován žádný kredit. Na adrese http://devel.areality.sk běží kopie webové aplikace serveru, kde je možné ověřit stav databáze.
4. po odladění je nutné otestovat aplikaci spolu se správcem serveru. Termín je třeba si nejprve dohodnout telefonicky nebo mailem. Celý test trvá cca 10 minut, pokud je export správně naprogramován.
5. pokud test proběhne bez chyb, přidá správce na server areality.sk UserAgentID s heslem, aby bylo možné export provozovat na tomto serveru.
6. pro funkční export musí mít RK zapnutou službu „Importovací SW tretích strán“. Cena viz ceník služeb.