Wymiana danych o biletach w systemie P&R Tarnów

ver. dokumentu 0.1

data ostatniej aktualizacji: 2015-11-12

Zmiany :

v0.1 2015-11-12 - Inicjacja dokumentacji wersja 0.1

1. Wprowadzenie

Webservice do wymiany danych zwany dalej WS jest wykonany w technologii XMLRPC. Dane są replikowane pomiędzy terminalami parkingowymi, Serwerem Lokalnym oraz Kasą Parkingową (tzw. hosty aktywne parkingu). Dokumentacja opisuje stan z wersją 1035 i wyższą.

2. Obiekty replikowane w systemie parkingowym

Obiekty potrzebne do funkcjonowania systemy biletowego zamodelowane są w .. /telpark_unicard/tickets_app/models.py. Poszczególne pola obiektów zostały opisane w dokumentacji kodu źródłowego. Algorytm wyliczający cenę obiektu Tickets znajduje się w managerze TicketPricesManager.

Omówienie kluczowych obiektów.

- TicketBillTime - obiekt opisujący algorytm naliczania/takowania (jednostka czasu rozliczeniowa) rozliczenia biletu. **Replikacja obowiązkowa.**

- TicketPrices - obiekt opisujący cenę rozliczeniową za zdefiniowaną jednostkę rozliczeniową w zadanym okresie czasu. Jednostką atomową w systemie parkingowym jest minuta. **Replikacja obowiązkowa.**

- Tickets - obiekt opisujący bilet oraz jego kluczowe parametry jak cena, czas parkowania, miejsce rozliczenia etc. **Replikacja obowiązkowa.**

- TicketsHistory - obiekt relacyjnie powiązany z każdym obiektem Tickets przetrzymujący histarię zmian poszczególnego obiektu Tickets. Stosowany w celu prześledzenia zmian, ma zadanie czysto informacyjne. **Replikacja opcjonalna.**

- PaymentTerminalSettings - obiekty przetrzymujący informację o poziomie gotówki w hopperze i payoucie. Po przekroczeniu zdefionowanego poziomu następuje raz na dobę przepływ do kasy parkingowej. **Replikacja obowiązkowa** w przypadku zastosowania urządzeń Inovation Technology Smart Payout NV200, Smart Hopper.

- TicketsCardPayment - obiekt opisujący transakcję zawartą poprzez Terminal płatniczy UX300B Verifone zrealizowaną poprzez Elavon. Algorytm dokładnie jest opisany w Wincor Nixdorf mPOS Protocol - Generic Specification Version 1.2.8. Obiekt ma funkcję czysto informacyjną. **Replikacja obowiązkowa** w przypadku zastosowania urządzeń UX300B Verifone i transakcji karcianych realizowanych poprzez Elavon.

3. Format seriazlizacji danych i procedura synchronizacji.

3.1. Ramka synchronizacyjna jest oparta o format przenośny danych JSON.
Schemat ogólny:

```
{
   "table": nazwa tabeli w DB
   "kind": rodzaj synchronizacji I - insert/edit, D - Delete
   "data": dane poszczególnego rekordu zserializowane za pomocą mechanizmu
         django.core.serializers. Dokumentacja:
   https://docs.djangoproject.com/en/1.8/topics/serialization
    }
```

3.2. Funkcja odpowiedzialna za serializację danych (make_sync_data) znajduje się:
/telpark_unicard/syncer/utils.py

3.3. Tak przygotowaną ramkę danych wysyłamy przez metodę XMLRPC syncdata na każdym hoscie aktywnym parkingu. WS syncdata jest zdefiniowany w /telpark_unicard/syncer/__init__.py pod funkcją syncdata. Funkcja syncdata przyjmuje dwa parametry:
token - jego wartość jest zdefiniowana /telpark_unicard/telpark_unicard/settings_base.py pod zmienną RPC_TOKEN.

data - patrz pkt. 3.1 i przykłady poniżej 3.4.

3.4. Przykłady serializacji wszystkich obiektów kluczowych

Przykład serializacji obiektu TicketBillTime:


```
{
   "table": "ticket_billtime",
   "kind": "I",
   "data": [
      {
         "fields": {
            "name": "Minutowe",
            "factor": "1.00"
         },
         "model": "ticket_app.ticketbilltime",
         "pk": "1571db90-614d-4c11-a172-8ffb7ab20dc3"
      }
   ]
}
```

Przykład serializacji obiektu TicketPrices:


```
{
   "table": "ticket_prices",
   "kind": "I",
   "data": [
      {
         "fields": {
            "skip_prices_before": false,
            "enable": true,
            "price_const": "0.00",
            "ticket_type": "D",
            "bill_time": "1571db90-614d-4c11-a172-8ffb7ab20dc3",
            "price_modul": "embeded",
            "stop": 999999999,
            "start": 0,
            "day_type": "W",
            "price_per_bill_time": "0.00"
         },
         "model": "ticket_app.ticketprices",
         "pk": "5a4ba475-ed07-4f6f-ad1c-6f7f1ecfe3ec"
      }
   ]
}
```


Przykład serializacji obiektu Tickets:


```
{
   "table": "ticket_ticket",
   "kind": "I",
   "data": [
      {
         "fields": {
            "payout": "0.00",
            "opens_count": 1,
            "start": "2015-11-05T10:25:01",
            "ticket_type": "D",
            "opens_count_used": 0,
            "price": "0.00",
            "barcode": "pr658f013473",
            "cashier": "kiosk",
            "comment": "",
            "stop": "2015-11-05T10:25:01",
            "credit": "0.00",
            "payment_place": "K",
            "duration": "0",
```

```
            "leave_time": 15,
            "billed": false
        },
        "model": "ticket_app.tickets",
        "pk": "aaa09386-ba97-4e21-b5e7-963206761f1a"
    }
  ]
}
```


Przykład serializacji obiektu TicketsHistory:


```
{
  "table": "ticket_ticket_history",
  "kind": "I",
  "data": [
    {
      "fields": {
        "ticket": "99d332c0-932e-4e98-ab88-06985b2fd473",
        "date_sync": "2015-10-30T12:34:00",
        "sync_data": "Terminal wyjazdowy \r\n\r\n {   '_state':
<django.db.models.base.ModelState object at 0x315ff10>,\r\n    'barcode':
u'pra6dda26e5b',\r\n    'billed': True,\r\n    'cashier': u'terminal_wyjazdowy',\r\n    'comment':
u'',\r\n    'credit': Decimal('0.00'),\r\n    'duration': u'0:00:16',\r\n    'id': UUID('99d332c0-932e-
4e98-ab88-06985b2fd473'),\r\n    'leave_time': 15,\r\n    'opens_count': 1,\r\n
'opens_count_used': 1,\r\n    'payment_place': u'K',\r\n    'payout': Decimal('0.00'),\r\n    'price':
Decimal('0.00'),\r\n    'start': datetime.datetime(2015, 10, 30, 12, 33, 35, 562000),\r\n    'stop':
datetime.datetime(2015, 10, 30, 12, 33, 52, 455989),\r\n    'ticket_type': u'D'} \r\n\r\n {
'_state': <django.db.models.base.ModelState object at 0x315ff10>,\r\n    'barcode':
u'pra6dda26e5b',\r\n    'billed': True,\r\n    'cashier': u'terminal_wyjazdowy',\r\n    'comment':
u'',\r\n    'credit': Decimal('0.00'),\r\n    'duration': u'0:00:16',\r\n    'id': UUID('99d332c0-932e-
4e98-ab88-06985b2fd473'),\r\n    'leave_time': 15,\r\n    'opens_count': 1,\r\n
'opens_count_used': 1,\r\n    'payment_place': u'K',\r\n    'payout': Decimal('0.00'),\r\n    'price':
Decimal('0.00'),\r\n    'start': datetime.datetime(2015, 10, 30, 12, 33, 35, 562000),\r\n    'stop':
datetime.datetime(2015, 10, 30, 12, 33, 52, 455989),\r\n    'ticket_type': u'D'}"
      },
      "model": "ticket_app.ticketshistory",
      "pk": "010040d9-a0f6-4fca-94f7-f61ac357c2bd"
    }
  ]
}
```


Przykład serializacji obiektu PaymentTerminalSettings:


```
{
  "table": "ticket_ticket_float_value",
  "kind": "I",
  "data": [
    {
      "fields": {
```

```
            "c_02": 0,
            "c_20": 4,
            "name": "Kasa1",
            "c_05": 0,
            "ip": "10.0.0.43",
            "enabled": true,
            "c_10": 15,
            "c_100": 0,
            "c_1": 30,
            "c_50": 1,
            "c_2": 20,
            "c_5": 20,
            "c_200": 0
         },
         "model": "ticket_app.paymentterminalsettings",
         "pk": "dfb7e6c3-ae0a-4ad9-bbe6-a2cc00720a6f"
      }
   ]
}
```


Przykład serializacji obiektu TicketsCardPayment:


```
{
   "table": "ticket_ticket_card_payment",
   "kind": "I",
   "data": [
      {
         "fields": {
            "application_transaction_counter": "026B",
            "terminal_id": "24200012",
            "transaction_time": "134027",
            "cvm_results_descr": "No cardholder verification",
            "sync_data": "{u'application_transaction_counter': u'026B', u'terminal_id':
u'24200012', u'transaction_time': u'134027', u'cvm_results_descr': u'No cardholder
verification', u'pos_entry_mode': u'07', u'cash_register_id': u'POS00001',
u'pos_entry_mode_desc': u'Contactless (ICC)', u'response_cause_descr': u'Transaction
performed successfully', u'application_label': u'PPC MCD 01  v20',
u'dedicated_file_name_df': u'A0000000041010', u'transaction_date': u'151112',
u'application_card_preferred_name': None, u'application_cryptogram':
u'AA64EC66EB8DF9C0', u'status_message': 'Transakcja zaakceptowana',
u'cryptogram_information_data_print': None, u'authorization_code': u'017788',
u'pci_dss_compliant_pan_part': u'541333001016', 'barcode': u'prc38bab766f', u'rodzaj_ramki':
u'9990', u'cryptogram_information_data_descr': None, u'invoice_number': u'000207',
u'merchant_id': u'     1000011111', u'cvm_results_value_descr': u'Successful', u'cvm_results':
u'1F0302', u'response_code': u'0000', u'batch_number': u'000001', u'amount': u'0.5',
u'rrn_number': u'040000000138', u'aid': u'A0000000041010', u'cryptogram_information_data':
None}",
            "merchant_id": "     1000011111",
```

```
            "pos_entry_mode": "07",
            "cash_register_id": "POS00001",
            "pos_entry_mode_desc": "Contactless (ICC)",
            "date_sync": "2015-11-12T13:40:34.796",
            "application_label": "PPC MCD 01  v20",
            "dedicated_file_name_df": "A0000000041010",
            "transaction_date": "151112",
            "application_card_preferred_name": null,
            "application_cryptogram": "AA64EC66EB8DF9C0",
            "status_message": "Transakcja zaakceptowana",
            "cryptogram_information_data_print": null,
            "authorization_code": "017788",
            "pci_dss_compliant_pan_part": "541333001016",
            "rodzaj_ramki": "9990",
            "cryptogram_information_data_descr": null,
            "invoice_number": "000207",
            "ticket": "41e860b2-c9b6-4718-916f-f50806a2ab70",
            "response_cause_descr": "Transaction performed successfully",
            "cvm_results_value_descr": "Successful",
            "cvm_results": "1F0302",
            "response_code": "0000",
            "batch_number": "000001",
            "amount": "0.5",
            "rrn_number": "040000000138",
            "aid": "A0000000041010",
            "cryptogram_information_data": null
        },
        "model": "ticket_app.ticketscardpayment",
        "pk": "8df9cdd2-7650-49ea-a34b-a1a2e1bfca47"
    }
  ]
}
```


4. Wytyczne

- Aby zachować poprawną strukturę zaleca się buforować dane replikacyjne, w przypadku utraty połączenia z hostami aktywnymi parkingu, aby móc wykonać synchronizację w okresie późniejszym.

5. Appendinx

5.1 Hosty aktywne - obiekt Tarnów.

10.80.2.11 - terminal wjazd
10.80.2.21 - terminal wyjazd
10.80.2.32 - kasa parkingowa
10.80.2.101 - Serwer Lokalny

5.2 Metody XMLRPC sterujące urządzeniami Inovation Technology Smart Payout NV200, Smart Hopper wywoływane na hoscie 10.80.2.32. Wszystkie poniższe metody przyjmują jako parametr zmienną token (RPC_TOKEN zdefioniowaną w settings_base.py)

- hopper_payout_empty - opróżnienie zawartości hoppera i payouta do kasety.

- hopper_payout_float_mode - rozkaz przepływu środków do kasety wg stany zdefiniowanego w obiekcie **PaymentTerminalSettings.**

- hopper_payout_amount - poziom środków w hopperze i payoucie z rozbiciem na nominały.

- hopper_payout_load_mode - wywołanie w kasie parkingowej trybu załadunku aby wprowadzić środki za pomocą żetonirki i payouta.

- hopper_payout_reset - restart sprzętowy urządzeń hopper i payout.