Пример использования JSON интерфейса на языке PHP

Пример использования класса

```
$gate = new prostor-sms_JsonGate('api_login', 'api_password');

   var_dump($gate->credits()); // узнаем текущий баланс
   var_dump($gate->senders()); // получаем список доступных подписей

   $messages = array(
               array(
               "clientId" => "1",
               "phone"=> "71234567890",
               "text"=> "first message",
               "sender"=> "TEST"
           ),
           array(
               "clientId" => "2",
               "phone"=> "71234567891",
               "text"=> "second message",
               "sender"=> "TEST",
           ),
           array(
               "clientId" => "3",
               "phone"=> "71234567892",
               "text"=> "third message",
               "sender"=> "TEST",
           ),
       );
   var_dump($gate->send($messages, 'testQueue')); // отправляем пакет sms

   $messages = array(
        array("clientId"=>"1","smscId"=>11255142),
        array("clientId"=>"2","smscId"=>11255143),
        array("clientId"=>"3","smscId"=>11255144),
   );
   var_dump($gate->status($messages)); // получаем статусы для пакета sms
   var_dump($gate->statusQueue('testQueue', 10)); // получаем статусы из
очереди 'testQueue'
```

Класс prostor-sms_JsonGate

```
class prostor-sms_JsonGate
{

    const ERROR_EMPTY_API_LOGIN = 'Empty api login not allowed';
    const ERROR_EMPTY_API_PASSWORD = 'Empty api password not allowed';
    const ERROR_EMPTY_RESPONSE = 'errorEmptyResponse';

        protected $_apiLogin = null;

        protected $_apiPassword = null;

    protected $_host = 'json.gate.prostor-sms.ru';

    protected $_packetSize = 200;

    protected $_results = array();

        public function __construct($apiLogin, $apiPassword)
        {
                $this->_setApiLogin($apiLogin);
                $this->_setApiPassword($apiPassword);
        }

        private function _setApiLogin($apiLogin)
        {
                if (empty($apiLogin)) {
                        throw new Exception(self::ERROR_EMPTY_API_LOGIN);
                }
                $this->_apiLogin = $apiLogin;
        }

    private function _setApiPassword($apiPassword)
    {
        if (empty($apiPassword)) {
            throw new Exception(self::ERROR_EMPTY_API_PASSWORD);
        }
        $this->_apiPassword = $apiPassword;
    }

    public function setHost($host)
    {
        $this->_host = $host;
    }

    public function getHost()
    {
        return $this->_host;
    }

    private function _sendRequest($uri, $params = null)
    {
        $url = $this->_getUrl($uri);
        $data = $this->_formPacket($params);

        $client = curl_init($url);
        curl_setopt_array($client, array(
                CURLOPT_RETURNTRANSFER => true,
                CURLOPT_POST => true,
                CURLOPT_HEADER => false,
                CURLOPT_HTTPHEADER => array('Host: ' . $this->getHost()),
                CURLOPT_POSTFIELDS => $data,
        ));

        $body = curl_exec($client);
                curl_close($client);
        if (empty($body)) {
                throw new Exception(self::ERROR_EMPTY_RESPONSE);
        }
        $decodedBody = json_decode($body, true);
        if (is_null($decodedBody)) {
                throw new Exception($body);
        }
        return $decodedBody;
    }

    private function _getUrl($uri)
    {
        return 'http://' . $this->getHost() . '/' . $uri . '/';
    }

    private function _formPacket($params = null)
    {
        $params['login'] = $this->_apiLogin;
        $params['password'] = $this->_apiPassword;
        foreach ($params as $key => $value) {
                if (empty($value)) {
                        unset($params[$key]);
                }
        }
        $packet = json_encode($params);
        return $packet;
    }

    public function getPacketSize()
    {
        return $this->_packetSize;
    }

    public function send($messages, $statusQueueName = null, $scheduleTime =
null)
    {
        $params = array(
            'messages' => $messages,
            'statusQueueName' => $statusQueueName,
            'scheduleTime' => $scheduleTime,
        );
        return $this->_sendRequest('send', $params);
    }

    public function status($messages)
    {
        return $this->_sendRequest('status', array('messages' => $messages));
    }

    public function statusQueue($name, $limit)
    {
        return $this->_sendRequest('statusQueue', array(
                'statusQueueName' => $name,
                'statusQueueLimit' => $limit,
        ));
    }

    public function credits()
    {
        return $this->_sendRequest('credits');
    }

    public function senders()
    {
        return $this->_sendRequest('senders');
    }

}
```