

# **Отчет по лабораторной работе №2**

Данила Андреевич Стариков

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
<b>3</b>	<b>Выводы</b>	<b>24</b>

# 1 Цель работы

Познакомиться с протоколом JSON-PRC на примере составления запросов к сервису .

## 2 Выполнение лабораторной работы

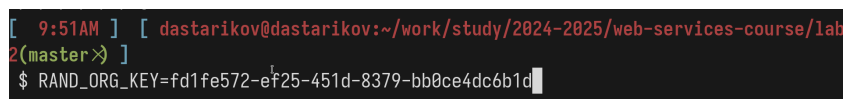
- Какие поля запроса и ответа фиксируются стандартом JSON-RPC?

При составлении запроса обязательно должны быть поля: “jsonrpc” — версия используемого протокола JSON-RPC, “method” — строка с указанием типа вызываемого метода, “params” — массив данных, которые передаются методу, “id” — значение любого типа, которое будет использоваться для идентификации ответа сервера.

При составлении ответа обязательно должны быть поля: “jsonrpc” — версия используемого протокола JSON-RPC, “result” — возвращаемые методом данные (null если при работе метода произошла ошибка), “error” — код ошибки (должен быть только при наличии ошибки), “id” — то же значение, что и в соответствующем запросе.

- Сохраняли ли вы ключ в переменную для дальнейшего использования или каждый раз вставляли в тело запроса?

Создана переменная окружения для API ключа, далее в теле запроса указывалось только имя этой переменной:



```
[ 9:51AM ] [ dastarikov@dastarikov:~/work/study/2024-2025/web-services-course/lab2(master) ]  
$ RAND_ORG_KEY=fd1fe572-ef25-451d-8379-bb0ce4dc6b1d
```

Рис. 2.1: Создание переменной окружения для хранения ключа.

- Сформируйте и выполните запрос к методу generateIntegers используя только обязательные параметры. Запросите 5 случайных чисел. Диапазон чисел выбирайте сами.

Выбран диапазон чисел от 1 до 10, запрос выглядит следующим образом:

```
$ http POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegers" \
params:='{
    "apiKey": "'$RAND_ORG_KEY'",
    "n": 5,
    "min": 1,
    "max": 10,
    "replacement": true
}' \
id:=1
```

Тело ответа выглядит так:

```
{

    "id": 1,
    "jsonrpc": "2.0",
    "result": {
        "advisoryDelay": 520,
        "bitsLeft": 249932,
        "bitsUsed": 17,
        "random": {
            "completionTime": "2025-04-14 06:55:51Z",
            "data": [
                10,
                6,
                10,
```

```

        3,
        2
    ]
},
"requestsLeft": 996
}
}

```

```

[ 9:55AM ] [ dastarikov@dastarikov:~/work/study/2024-2025/web-services-course/lab2(master)> ]
$ http POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegers" \
params='{
  "apiKey": "'$RAND_ORG_KEY'",
  "n": 5,
  "min": 1,
  "max": 10,
  "replacement": true
}' \
id:=1

```

Рис. 2.2: Запрос к методу generateIntegers.

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 520,
    "bitsLeft": 249932,
    "bitsUsed": 17,
    "random": {
      "completionTime": "2025-04-14 06:55:51Z",
      "data": [
        10,
        6,
        10,
        3,
        2
      ]
    },
    "requestsLeft": 996
  }
}
```

Рис. 2.3: Ответ метода generateIntegers.

- Какие поля в заголовке HTTP-запроса вы установили (отобразите заголовок вашего запроса)?

```
POST /json-rpc/2/invoke HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 169
Content-Type: application/json
Host: api.random.org
User-Agent: HTTPie/3.2.1
```

Рис. 2.4: Заголовок запроса метода generateIntegers.

- Каким HTTP-методом осуществляются запросы к random.org?

Запросы к random.org осуществляется с помощью метода POST.

- Отобразите тело ответа, который прислал сервер. Поясните все его поля.

В теле ответа содержатся поля:

- "id": 1 — идентификатор, совпадающий с id, указанным при запросе
- "jsonrpc": "2.0" — версия протокола JSON-RPC
- result — поле с данными, относящимися к ответу на запрос:
  - "advisoryDelay": 930 — рекомендуемая задержка перед отправкой следующего запроса (в мс)
  - "bitsLeft": 249864 — оставшееся количество битов для бесплатного пользования сервисом
  - "bitsUsed": 17 — количество битов использованное для выполнения этого запроса
  - "random" — поле, содержащее случайные данные ("data") и время выполнения запроса ("completionTime")
- "requestsLeft" — количество оставшихся запросов для бесплатного пользования сервисом



```
{
  "id": 1,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 520,
    "bitsLeft": 249932,
    "bitsUsed": 17,
    "random": {
      "completionTime": "2025-04-14 06:55:51Z",
      "data": [
        10,
        6,
        10,
        3,
        2
      ]
    },
    "requestsLeft": 996
  }
}
```

Рис. 2.5: Ответ метода generateIntegers.

- Отобразите поля заголовка HTTP-ответа.

```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Origin: *
CF-RAY: 93015b4fad7f9db3-DME
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/json; charset=utf-8
Date: Mon, 14 Apr 2025 07:05:46 GMT
Server: cloudflare
Set-Cookie: __cfllb=0H28v41hJSkfmceVJ7ewzLrD5T9tbGCHCD63JhjW2f; SameSite=Lax; path=/; expires=Mon, 21-Apr-25 07:05:46 GMT; HttpOnly
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
cf-cache-status: DYNAMIC
```

Рис. 2.6: Заголовок ответа метода generateIntegers.

- Сформируйте запрос на 5 чисел в восьмеричной системе счисления. Какого типа данные будут в поле data ответа сервера? Под типом подразумеваются тип поддерживаемый форматом JSON.

```
POST /json-rpc/2/invoke HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 180
Content-Type: application/json
Host: api.random.org
User-Agent: HTTPie/3.2.1

{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "generateIntegers",
  "params": {
    "apiKey": "fd1fe572-ef25-451d-8379-bb0ce4dc6b1d",
    "base": 8,
    "max": 10,
    "min": 1,
    "n": 5,
    "replacement": true
  }
}
```

Рис. 2.7: Запрос метода generateIntegers с восьмеричной системой счисления.

```

{
  "id": 1,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 900,
    "bitsLeft": 249847,
    "bitsUsed": 17,
    "random": {
      "completionTime": "2025-04-14 07:17:20Z",
      "data": [
        "04",
        "05",
        "02",
        "05",
        "04"
      ]
    },
    "requestsLeft": 991
  }
}

```

Рис. 2.8: Ответ метода generateIntegers с восьмеричной системой счисления.

Заметим, что теперь массив "data" хранит данные в виде строк, причем на первом месте всегда стоит ноль.

- Как вы думаете, зачем нужен параметр advisoryDelay в ответе от сервера? Чему он равен в вашем случае? Менялся ли он от запроса к запросу?
  - "advisoryDelay" — рекомендуемая задержка перед отправкой следующего запроса (в мс). В ходе работы меняется, например: 900, 930, 730.
- Изучите список ошибок, которые обрабатывает сервер. Сформируйте два запроса, которые вызовут две разные ошибки на ваш выбор.
  - Ошибка -32601 Method not found:

```
[ 10:17AM ] [ dastarikov@dastarikov:~/work/study/2024-2025/web-services-course/lab2(master)> ]
$ http --verbose POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="badMethod" \
params='{
  "apiKey": "'$RAND_ORG_KEY'",
  "base": 8,
  "n": 5,
  "min": 1,
  "max": 10,
  "replacement": true
}' \
id:=1
```

Рис. 2.9: Запрос с ошибочным методом.

```
{
  "error": {
    "code": -32601,
    "data": null,
    "message": "Method not found"
  },
  "id": 1,
  "jsonrpc": "2.0"
}
```

Рис. 2.10: Ответ на запрос с ошибочным методом.

- Ошибка 200 Parameter is malformed:

```
[ 10:28AM ] [ dastarikov@dastarikov:~/work/study/2024-2025/web-services-course/lab2(master)> ]
$ http --verbose POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegers" \
params='{
  "apiKey": "'$RAND_ORG_KEY'",
  "base": 8,
  "n": "qqq",
  "min": 1,
  "max": 10,
  "replacement": true
}' \
id:=1
```

Рис. 2.11: Запрос с неправильным параметром.

```
{
  "error": {
    "code": 200,
    "data": [
      "n"
    ],
    "message": "Parameter 'n' is malformed"
  },
  "id": 1,
  "jsonrpc": "2.0"
}
```

Рис. 2.12: Ответ на запрос с неправильным параметром.

- Теперь сформируйте и отправьте запрос к методу `generateIntegerSequences`.

Сгенерировали один массив чисел от 1 до 30 длиной 20:

```
"apiKey": "$RAND_ORG_KEY",
"base": 10,
"n": 1,
"length": 20,
"min": 1,
"max": 30,
"replacement": true
}' \
id:=42

POST /json-rpc/2/invoke HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 204
```

Рис. 2.13: Запрос к методу `generateIntegerSequence`.

```

{
  "id": 42,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 940,
    "bitsLeft": 249609,
    "bitsUsed": 98,
    "random": {
      "completionTime": "2025-04-14 07:52:09Z",
      "data": [
        [
          13,
          28,
          14,
          16,
          26,
          4,
          29,
          13,
          8,
          2,
          19,
          15,
          23,
          19,
          13,
          28,
          15,
          25,
          17,
          2
        ]
      ]
    },
    "requestsLeft": 989
  }
}

```

Рис. 2.14: Ответ метода generateIntegersSequence.

- Покажите подробно исходный код вашего запроса, и исходный код ответа от сервера. Появились ли какие-то новые поля заголовка HTTP-сообщения?

```

http --verbose POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegerSequences" \
params:='{
    "apiKey": "'$RAND_ORG_KEY'",
    "base" : 10,
    "n": 1,
    "length" : 20,
    "min": 1,
    "max": 30,
    "replacement": true
}' \
id:=42

{
    "id": 42,
    "jsonrpc": "2.0",
    "result": {
        "advisoryDelay": 940,
        "bitsLeft": 249609,
        "bitsUsed": 98,
        "random": {
            "completionTime": "2025-04-14 07:52:09Z",
            "data": [
                [
                    13,
                    28,
                    14,
                    16,

```

```

        26,
        4,
        29,
        13,
        8,
        2,
        19,
        15,
        23,
        19,
        13,
        28,
        15,
        25,
        17,
        2
    ]
]
},
    "requestsLeft": 989
}
}

```

Новые поля в HTTP запросе не появились.

- В чем отличия поля random в случае метода generateIntegerSequences?

Этот метод генерирует данные в виде последовательности, поэтому элементом массива data тоже будет массив.

- Как с помощью random.org просимулировать подкидывание монеты или



шестигранной игровой кости (кубика)? Подумайте, какие параметры нужно указать в запросе.

Пусть 0 - ребро, 1 - решка. Тогда нужно построить запрос, генерирующий последовательность нулей и единиц:

```
http --verbose POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegerSequences" \
params:='{
    "apiKey": "'$RAND_ORG_KEY'",
    "base" : 10,
    "n": 1,
    "length" : 20,
    "min": 0,
    "max": 1,
    "replacement": true
}' \
id:=42
```

```

{
  "id": 42,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 610,
    "bitsLeft": 249569,
    "bitsUsed": 20,
    "random": {
      "completionTime": "2025-04-14 07:59:50Z",
      "data": [
        [
          1,
          1,
          0,
          0,
          1,
          0,
          0,
          1,
          1,
          0,
          1,
          1,
          1,
          0,
          1,
          1,
          0,
          1,
          0,
          1,
          0,
          1
        ]
      ]
    },
    "requestsLeft": 987
  }
}

```

Рис. 2.15: Симуляция подбрасывания монетки.

- Представьте, что вам нужно сгенерировать случайные IP-адреса (5 штук).

Какой запрос нужно послать к random.org?

Один IP-адрес можно представить в виде последовательности 4 чисел, принимающих значения от 0 до 255.

```
http --verbose POST https://api.random.org/json-rpc/2/invoke \
Content-Type:application/json \
jsonrpc="2.0" \
method="generateIntegerSequences" \
params:='{
    "apiKey": "'$RAND_ORG_KEY'",
    "base" : 10,
    "n": 5,
    "length" : 4,
    "min": 0,
    "max": 255,
    "replacement": true
}' \
id:=42

{
    "id": 42,
    "jsonrpc": "2.0",
    "result": {
        "advisoryDelay": 520,
        "bitsLeft": 249409,
        "bitsUsed": 160,
        "random": {
            "completionTime": "2025-04-14 08:03:22Z",
            "data": [
                [
```

```
        100,  
        20,  
        228,  
        237  
    ],  
    [  
        194,  
        77,  
        29,  
        47  
    ],  
    [  
        14,  
        62,  
        3,  
        188  
    ],  
    [  
        101,  
        182,  
        66,  
        170  
    ],  
    [  
        186,  
        175,  
        250,  
        213  
    ]  
]
```

```
    ]  
  },  
  "requestsLeft": 986  
}  
}
```

```

{
  "id": 42,
  "jsonrpc": "2.0",
  "result": {
    "advisoryDelay": 520,
    "bitsLeft": 249409,
    "bitsUsed": 160,
    "random": {
      "completionTime": "2025-04-14 08:03:22Z",
      "data": [
        [
          100,
          20,
          228,
          237
        ],
        [
          194,
          77,
          29,
          47
        ],
        [
          14,
          62,
          3,
          188
        ],
        [
          101,
          182,
          66,
          170
        ],
        [
          186,
          175,
          250,
          213
        ]
      ]
    },
    "requestsLeft": 986
  }
}

```

Рис. 2.16: Генерация 5 IP-адресов.

- Подумайте, зачем может понадобиться `generateIntegerSequences` и почему нельзя обойтись многократным вызовом `generateIntegers`?

Использование метода `generateIntegerSequences` позволяет уменьшить количество запросов к сервису, уменьшить генерации последовательностей и расходы на отправку запросов.

## **3 Выводы**

В результате выполнения лабораторной работы познакомились с протоколом JSON-RPC на примере составления запросов к сервису