|  |
| --- |
| Веб-сервис  «Получение информации о зарегистрированных уведомлениях клиента (пользователя) за период»  Версия 1.0 |
| Описание электронного сервиса |

Оглавление

[1. Термины и сокращения 3](#_Toc480449608)

[2. Концепция веб-сервиса 3](#_Toc480449609)

[2.1. Базовые принципы 3](#_Toc480449610)

[2.1.1. Описание документа 3](#_Toc480449611)

[2.1.2. Глоссарий 3](#_Toc480449612)

[2.1.3. Общее описание 3](#_Toc480449613)

[2.1.4. Концепция веб-сервиса 4](#_Toc480449614)

[2.1.5. Концептуальная схема взаимодействия 5](#_Toc480449615)

[2.2. Технические подробности 5](#_Toc480449616)

[2.2.1. Описание документа 5](#_Toc480449617)

[2.2.2. Взаимодействие с веб-сервисом 6](#_Toc480449618)

[2.2.3. Примеры вызова 8](#_Toc480449619)

[2.2.4. Требования к типам данных 9](#_Toc480449620)

[2.2.5. Системные ошибки сервиса 11](#_Toc480449621)

[2.2.6. Приложение 1 11](#_Toc480449622)

[3. Описание ресурсов 11](#_Toc480449623)

[3.1. Получение информации об остатке средств на счете пользователя 11](#_Toc480449624)

[3.1.1. Входные параметры HTTP запроса 11](#_Toc480449625)

[3.1.2. Результат выполнения HTTP запроса 12](#_Toc480449626)

[3.1.3 Описание ресурса 13](#_Toc480449627)

[Описание ошибок 13](#_Toc480449628)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 15](#_Toc480449629)

# 

# Термины и сокращения

JSON – текстовый формат обмена сообщениями, как описано в RFC 7159 (<http://www.rfc-editor.org/rfc/rfc7159.txt>) и стандарте ECMA-404 (<http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> )

# Концепция веб-сервиса

В данном разделе содержатся документы с описанием концепции веб-сервиса, глоссарием, описанием технологии отправки запросов, описанием возможных типов данных, а также с примерами запросов и результатов выполнения.

## Базовые принципы

### Описание документа

Данный документ содержит описание концепции веб-сервиса. В данном документе указаны базовые принципы построения веб-сервиса и концептуальная схема взаимодействия.

### Глоссарий

REST – (Representational State Transfer — «передача состояния представления») это стиль архитектуры программного обеспечения для распределенных систем, который, как правило, используется для построения веб-служб.

HTTP метод – служебные слова, определенные в RFC 7231, указывающие на требуемую операцию над ресурсом.

HTTP заголовки – строки в HTTP-сообщении, определенные в RFC 7231, содержащие разделённую двоеточием пару параметр-значение, несущие информацию для клиента или сервера о свойствах запроса или ответа.

Формат JSON – (JavaScript Object Notation) - простой формат обмена данными, определенный по стандарту ECMA-404, удобный для чтения и написания как человеком, так и компьютером.

URI – (Uniform Resource Identifier - унифицированный (единообразный) идентификатор ресурса) это последовательность символов, идентифицирующая абстрактный или физический ресурс.

### Общее описание

Веб-сервис предоставляет интерфейс для взаимодействия широкому кругу внутренних и внешних систем. Подобная возможность достигается благодаря использованию протокола HTTP, стандартизированного по RFC 7230 – 7235. В качестве базовой архитектуры применяется **REST Level 2**, который использует все преимущества нативных HTTP возможностей, таких как заголовки, коды статусов, определенные URI и другие. Однако, REST API второго уровня **не является RESTfull** приложением, т. к. не включает в себя применение HATEOAS (Hypermedia as the Engine of Application State) шаблона проектирования. Все входные параметры (за исключением запросов GET, где параметры передаются в URI) передаются в стандартном формате POST, результаты запросов представлены в формате JSON, что позволяет снизить объем трафика при передаче данных.

### Концепция веб-сервиса

Концепция веб-сервиса заключается в создании REST интерфейса второго уровня. При этом основным форматом данных является JSON.

Веб-сервис (REST API) принимает запросы от клиента, выполняет необходимые операции и возвращает результат.

Сервис принимает запрос по протоколу HTTP в следующем формате:

1. HTTP\_метод URI\_ресурса{?дополнительные\_параметры} (для метода GET) HTTP/версия
2. HTTP\_заголовки\_запроса
3. HTTP\_тело\_запроса (для методов POST, PUT, DELETE)

Допустимо использование следующих HTTP методов:

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Описание** | **Безопасный** | **Идемпотентный** |
| GET | Получение ресурса | Да | Да |
| POST | Создание ресурса | Нет | Нет |
| PUT | Изменение ресурса | Нет | Да |
| DELETE | Удаление ресурса | Нет | Да |

Метод считается **безопасным**, если он не изменяет состояния ресурса, т. е. предназначены только для получения информации. **Идемпотентный** метод может изменять или не изменять состояние ресурса, но повторные запросы, следующие за первым, не должны оказывать влияния на состояние ресурса. В то время, как идемпотентные операции производят один и тот же результат на сервере, ответ сам по себе может не быть тем же самым (например, состояние ресурса может измениться между запросами). При использовании метода DELETE производится удаление определенного ресурса с возвратом HTTP кода 200 OK, последующее использование данного метода повлечет возврат HTTP кода 404 Not found. Состояние на сервере после каждого вызова DELETE то же самое, но ответы разные.

Ответ веб-сервиса представляет собой заголовок HTTP, содержащий HTTP код результата выполнения запроса, и тело запроса, содержащее JSON-строку в кодировке UTF-8, являющуюся объектом результата выполнения запроса.

Сервис отправляет ответ по протоколу HTTP в следующем формате:

1. HTTP/версия HTTP\_код
2. HTTP\_заголовки\_ответа
3. HTTP\_тело\_ответа

Допустимые HTTP коды для ответа:

|  |  |
| --- | --- |
| **Код** | **Описание** |
| 200 OK | Запрос выполнен успешно |
| 400 Bad Request | При выполнении запроса произошла ошибка |
| 404 Not Found | Запрашиваемый ресурс не найден |
| 500 Internal server error | Внутренняя (системная) ошибка веб-сервиса |

### Концептуальная схема взаимодействия

Клиентское приложение осуществляет запрос к веб-сервису в определенном формате с использованием нативных HTTP методов, передавая в теле запроса необходимые параметры в стандартном формате POST (за исключением запросов GET, где параметры передаются в URI). После обработки запроса сервером, клиенту возвращается ответ, содержащий HTTP код результата работы запроса и тело ответа в формате JSON, содержащее результат работы запроса (в случае успешного завершения) или ошибку (код и текст).



## Технические подробности

### Описание документа

Данный документ содержит описание технических подробностей веб-сервиса. В данном документе описано взаимодействие клиентского приложения с веб-сервисом, показаны форматы запросов и ответов, а также установлены требования к форматам типов данных.

### Взаимодействие с веб-сервисом

Для взаимодействия с веб-сервисом используются HTTP методы GET, POST, PUT и DELETE. Заголовки запросов и ответов должны формироваться, основываясь на спецификации протокола HTTP 1.1 **RFC 7230 – 7235**.

#### Запрос

Заголовки HTTP запроса должны иметь следующее содержимое (за исключением запросов с помощью метода GET):

* Content-Type: application/x-www-form-urlencoded
* Content-Length: вычисляемое\_значение\_длины\_тела\_запроса\_в\_байтах

Тело HTTP запроса должно иметь следующее содержимое (за исключением запросов с помощью метода GET):

* Content: параметры запроса (тело запроса представляет собой стандартную POST строку в кодировке UTF-8, содержащую параметры запроса)

Для метода GET использовать заголовки не требуется, достаточно использовать только URI ресурса, в нем же передаются дополнительные параметры запроса.

Пример запроса для создания клиента:

1. POST /client HTTP/1.1
2. Content-Type: x-form-url-encoded
3. Content-Length: 205
4. v\_legalentity=0&v\_sex=0&v\_namerussfull="IVANOV I.I"&
5. v\_namerussshort="IVANOV"&v\_addressrus="Moscow"&v\_addressfact="Moscow"&
6. v\_inn=12345678&v\_okpo=4565&v\_okph=5665&v\_kpp=22334455

В теле запроса так же могут передаваться двоичные данные, в этом случае используется запрос составного типа (Content-Type: multipart/form-data). Раздел такого запроса, в котором находятся двоичные данные, идентифицируется строками:

Content-Type: application/octet-stream

Content-Transfer-Encoding: binary

(допустимо использование конкретного типа данных, если он известен, например image/jpg). Пример запроса для добавления документа в хранилище клиента:

1. POST /client/12345/storage HTTP/1.1
2. Content-Type: multipart/form-data; boundary=1BEF0A57BE110FD467A
3. Content-Length: 311
4. --1BEF0A57BE110FD467A
5. Content-Disposition: form-data; name="v\_docname"
6. Тестовый документ
7. --1BEF0A57BE110FD467A
8. Content-Disposition: form-data; name="v\_document\_file"
9. Content-Type: application/octet-stream
10. Content-Transfet-Encoding: binary
11. <содержимое\_файла>
12. --1BEF0A57BE110FD467A--

В случае возникновения ошибки, изменения не сохраняются, в ответе возвращается соответствующая ошибка.

В случае прерывания соединения по таймауту, изменения не сохраняются, возвращается HTTP код 500 (см. раздел Системные ошибки сервиса). В веб сервисе имеется возможность устанавливать таймаут обработки для каждого отдельного ресурса. Значение таймаута по умолчанию – 60 секунд. Модифицированные значения таймаута должны оговариваться в требованиях каждого отдельного ресурса.

*Примечание 1: в случае, если заголовок Content-Length имеет значение больше реального размера тела запроса, то веб сервер будет ожидать оставшихся данных до истечения таймаута, после чего запрос будет выполнен с имеющимися данными.*

*Примечание 2: таймаут ожидания тела запроса устанавливается администратором веб сервера.*

*Примечание 3: в случае, если заголовок Content-Length имеет значение меньше реального размера тела запроса, то веб сервер будет учитывать только те данные, размер которых не превышает указанный.*

#### Ответ

Заголовки HTTP ответа должны иметь следующее содержимое:

* HTTP\_код\_результата\_выполнения\_запроса
* Сontent-Type: application/json
* Content-Length: вычисляемое\_значение\_длины\_тела\_ответа\_в\_байтах

Тело HTTP ответа должно иметь следующее содержимое:

* Сontent: результат выполнения запроса (тело ответа представляет собой JSON-строку в кодировке UTF-8, являющуюся объектом результата выполнения запроса)

Пример ответа для абстрактного ресурса создания клиента c успешным исходом:

1. HTTP/1.1 200 OK
2. Сontent-type: application/json
3. Content-Length: 36
4. {
5. "result":
6. {
7. "code":"4064",
8. "data":""
9. }
10. }

Тело ответа содержит объект result, в котором находится результат выполнения запрошенного метода:

* code: код, получаемый в результате выполнения запроса (в данном случае код созданного клиента). Значение кода >= 0.
* data: объект, содержащий данные, полученные в ответ на запрос или пустая строка, если данных нет.

Пример ответа для изменения услуги с ошибкой:

1. HTTP/1.1 400 Bad request
2. Сontent-type: application/json
3. Content-Length: 212
4. {
5. "result":
6. {
7. "code":-1251,
8. "data":"Произошла ошибка при отключении услуги. Дата отключения услуги меньше даты ее подключения."
9. }
10. }

Тело ответа содержит объект result, в котором находится результат выполнения запрошенного метода:

* code: код, получаемый в результате выполнения запроса (в данном случае код ошибки). Значение кода < 0.
* data: строка, содержащая описание ошибки, возникшей при выполнении запрошенного метода.

### Примеры вызова

1. Запрос на создание клиента (POST запрос с параметрами)

curl -X POST -H "Content-Type: x-www-form-url-encoded" -d 'v\_legalentity=0&v\_sex=0&v\_namerussfull="IVANOV I.I"&v\_namerussshort="IVANOV"&v\_addressrus="Moscow"&v\_addressfact="Moscow"&v\_inn=12345678&v\_okpo=4565&v\_okph=5665&v\_kpp=22334455' http://localhost/api/client

{"result":{"code":"4065", "data":""}}

1. Запрос на возврат массива данных (информации о ТП) (GET запрос с динамически формируемым URI)

curl -X GET http://localhost/api/price\_list/1

{"result":{"code":"0","data":[{"COSERVICETYPE":"1000","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 640\*480, \u0431\u0435\u0437 \u0430\u0440\u0445\u0438\u0432\u0430","COST":"50"},{"COSERVICETYPE":"1001","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 640\*480, \u0430\u0440\u0445\u0438\u0432 7 \u0434\u043d\u0435\u0439","COST":"60"},{"COSERVICETYPE":"1002","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 640\*480, \u0430\u0440\u0445\u0438\u0432 14 \u0434\u043d\u0435\u0439","COST":"70"},{"COSERVICETYPE":"1003","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 1280\*800, \u0431\u0435\u0437 \u0430\u0440\u0445\u0438\u0432\u0430","COST":"80"},{"COSERVICETYPE":"1004","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 1280\*800, \u0430\u0440\u0445\u0438\u0432 7 \u0434\u043d\u0435\u0439","COST":"90"},{"COSERVICETYPE":"1005","NMSERVICETYPE":"\u0432. \u043d\u0430\u0431\u043b\u044e\u0434\u0435\u043d\u0438\u0435 1280\*800, \u0430\u0440\u0445\u0438\u0432 14 \u0434\u043d\u0435\u0439","COST":"100"}]}}

### Требования к типам данных

Для корректного взаимодействия клиентского приложения с веб-сервисом определяются форматы (в соответствии со спецификацией **ECMA-404**) для следующих типов данных:

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Описание** | **Пример** |
| Object | Объект JSON. Содержит разделенные запятой пары <ключ>:<значение> | {"v\_namerussshort":"IVANOV", "v\_addressrus":"Moscow,\\nRussia"} |
| Array | Массив JSON. Содержит перечисление данных любых типов | ["12345","\u0430\u0431\u043b \u044e",{"v\_namerussshort":"IVANOV", "v\_addressrus":"Moscow,\\nRussia"}] |
| String | Строковый формат данных. Специальные символы (кавычки, слеши, переносы строк и пр.) должны быть экранированы. Любые символы национальных алфавитов должны быть закодированы как Unicode-последовательности | "IVANOV"  "Moscow,\\nRussia" "\u043d\u0430\u0431\u043b \u044e\u0434 \u0435\u043d \u0438\u0435" |
| Number | Числовой формат данных. Используется для представления положительных, отрицательных целых и дробных чисел. Десятичным разделителем является «**точка**» | “12345” (целое положительное)  “-12345” (целое отрицательное)  “123.45” (дробное) |
| DateTime | Формат данных типа «дата и время». Данные всегда находятся в формате “dd.mm.yyyy hh24:mi:ss”. В случае отсутствия значения времени, будет возвращено “dd.mm.yyyy 00:00:00”. В случае отсутствия значения даты, будет возвращено “01.01.1900 hh24:mi:ss”.  Все значения указываются в часовом поясе БС | “24.03.1989 06:42:15” (дата со временем)  “15.11.2002 00:00:00” (дата без времени)  “01.01.1900 01:23:07” (время без даты) |

Все типы данных должны передаваться как строки (включая числовые типы) – обрамленные в двойные кавычки. Экранирование специальных символов будет осуществляться в соответствии со спецификацией **ECMA-404**.

*Примечание: для определенного ресурса формат данных по умолчанию может быть изменен. При этом в документации к данному ресурсу в ОБЯЗАТЕЛЬНОМ порядке указывается актуальный формат.*

### Системные ошибки сервиса

Системными ошибками сервиса будут являться: сбой программного обеспечения веб сервера, критическая ошибка в коде сервиса, нештатное завершение выполнения кода сервиса (таймаут). В случае возникновения системной ошибки сервиса будет возвращен HTTP код 500 Internal Server Error.

### Приложение 1

#### Переопределение HTTP методов.

В случае, если целевой браузер не поддерживает некоторые HTTP методы, веб-сервис предоставляет возможность использования метода POST с указанием в скрытом поле формы необходимого неподдерживаемого метода.

Пример использования для метода PUT:

1. <form action="/books/1" method="post">
2. ... other form fields here...
3. <input type="hidden" name="\_METHOD" value="PUT"/>
4. <input type="submit" value="Update Book"/>
5. </form>

Так же для переопределения HTTP метода допустимо использовать HTTP заголовок **X-HTTP-Method-Override**.

# Описание ресурсов

3.1 Получение информации о зарегистрированных уведомлениях клиента (пользователя) за период

Ресурс **GET** /client/ notifications

Адрес тестового экземпляра, доступного через сеть VipNet №4995: [http://10.251.201.31/atlant\_api/client/notifications](http://10.251.201.31/atlant_api/client/notifications%20)

Адрес промышленного экземпляра, доступного через сеть VipNet №4995: http://10.251.84.49/atlant\_api/client/notifications

Ресурс «Получение информации о зарегистрированных уведомлениях клиента за период» предназначен для получения данных о номерах, датах регистрации и ГУИД уведомлений, зарегистрированных клиентом, к которому привязан переданный во входных параметрах УИП за период переданных во входных параметрах дат. Ресурс поддерживает постраничную навигацию.

### Входные параметры HTTP запроса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | **Тип параметра** | **Обязательный параметр** | **Название параметра** | **Описание параметра** |
| 1 | String(25) | Да | uip | Уникальный идентификатор плательщика или выставленного счета, по которому определяется пользователь. |
| 2 | DateTime | Да | startdate | Дата и время начала периода (включительно). |
| 3 | DateTime | Нет | enddate | Дата и время окончания периода (включительно).  Если не задано, то за дату окончания периода принимается текущая дата. |
| **4** | Number(10) | Нет | batchindex | Уникальный идентификатор запроса.  Если при запросе этот параметр не заполнен, то происходит запрос к таблицам БД, данные кэшируются и выдается нужная страница данных; значение данного параметра для получения следующей порции данных возвращается в выходных параметрах. Если параметр заполнен, то происходит запрос только к кэшу и выдается нужная страница данных. |
| **5** | Number(10) | Да | pagenumber | Номер страницы |
| **6** | Number(3) | Да | recordscount | Количество строк на одну страницу |

### Результат выполнения HTTP запроса

#### HTTP-код 200 OK

В результате успешного выполнения запроса возвращается объект result, содержащий параметры data и page.

В параметре data находится содержимое выводимой страницы, массив объектов со следующими полями:

|  |  |  |  |
| --- | --- | --- | --- |
| **№№** | **Наименование** | **Тип** | **Описание** |
| **в1** | NOTIFICATIONNUMBER | String(23) | Регистрационный номер уведомления |
| **22** | NOTIFICATIONDATE | DateTime | Дата и время регистрации уведомления |
| **33** | NOTIFICATIONID | String(36) | ГУИД уведомлеения |

В параметре page находится информация, необходимая для работы постраничной навигации, массив объектов со следующими полями:

|  |  |  |  |
| --- | --- | --- | --- |
| **№№** | **Наименование** | **Тип** | **Описание** |
| **11** | batchindex | Number(10) | Уникальный идентификатор запроса |
| **22** | pagenumber | Number(10) | Номер страницы |
| **33** | pagestotal | Number(10) | Общее количество страниц |
| **44** | recordscount | Number(3) | Количество строк на одну страницу |
| **55** | recordstotal | Number(10) | Общее количество строк |

#### HTTP-код 400 Bad Request

В случае ошибки в параметре code возвращается код ошибки, в параметре data находится строка с текстом ошибки, параметр page не возвращается.

### 3.1.3 Описание ресурса

Ресурс выполняет следующие действия:

* + - 1. Поиск и определение типа УИП. 1-й тип УИП – УИП, привязанный к счету по заявке (заказ через Портал), 2-й тип УИП – УИП, привязанный к лицевому счету (УИП указывается при заказе через веб-сервис). Если УИП не найден, возвращается массив из 0 записей.
      2. Для УИП 1-го типа.
         1. Определяется пакет уведомлений, привязанный к заявке и статус регистрации (зарегистрировано/не зарегистрировано) каждого уведомления пакета. Период, указанный в параметрах запроса, не анализируется. Возвращается информация по каждому зарегистрированному уведомлению. Если нет зарегистрированных уведомлений, возвращается массив из 0 записей.
      3. Для УИП 2-го типа.
         1. Определяются пакеты уведомлений, поданные в указанный в параметрах запроса период. Для каждого найденного пакета определяется статус регистрации (зарегистрировано/не зарегистрировано) каждого уведомления. Возвращается информация по каждому зарегистрированному уведомлению. Если нет зарегистрированных уведомлений, возвращается массив из 0 записей.

# Описание ошибок

Возможные значения кодов ошибок, возвращаемых в ответах с HTTP-кодом 400 Bad Request, и их описания:

|  |  |  |
| --- | --- | --- |
| **Код** | **Текст** | **Описание** |
| **Функциональные ошибки** | | |
| -101 | Переданный УИП не найден в системе | Пользователь с переданным уникальным идентификатором плательщика не найден в системе  Проверка выполняется если параметр УИП передан в запросе |
| **Технические ошибки** | | |
| -401 | Не заполнены все обязательные поля | В переданном массиве данных присутствуют не заполненные обязательные поля |
| < -1000 | Внутренняя ошибка системы | Диапазон внутренних ошибок системы (необработанных исключений)  Текст и код ошибки зависит от конкретного компонента системы |

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |
| --- | --- | --- | --- |
| Номер версии документа | Дата изменения (дд.мм.гггг) | Автор изменения  (Ф.И.О.) | Комментарии по изменениям |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |