LiveTex Web API v2: Общие принципы

Версия: **2.0.116**

Дата: 29.07.2015

Таблица изменений

Версия	Дата	Описание
2.0.116	29.07.2015	1. Исправлены ссылки для доступа к api.
2.0.115	01.07.2015	1. Документация разделена на две независимые части. Подробное описание типов данных и ресурсов генерируется автоматически с реализационной схемы.

Оглавление

О пометках в документе

Документ является черновиком, в котором ведется проектирование Web API и который использоваться разработчиками в качестве спецификации.

Уровень готовности отдельных разделов помечен первым символом в названии раздела:

- «+» - проектирование завершено, готово к передаче в разработку.
- «±» - в целом проектирование завершено, раздел готов к обсуждению с разработчиками, но еще не готов для передачи в разработку.
- «~» - раздел находится в процессе проектирования, разработчикам на него еще рано обращать внимание.
- «-» – работы по разделу приостановлены, так как это пока неактуально для ЛК 2.0.
- Отсутствие пометки говорит о том, что раздел уже реализован разработчиками.

Если в уже реализованном раздел внесены изменения, то раздел помечается символом «+» (проектирование). При этом в таблице изменений в самом начале документа сверху добавляется запись о сути изменений.

Светло-серым цветом помечены пункты, реализация которых предполагает дополнительную доработку модели данных или подключение других команд. Без этих доработок помеченные пункты не могут быть реализованы.

1. Общие принципы

1.1. Аутентификация

Все запросы к LiveTex Web API требуют аутентификации.

В качестве параметров аутентификации при запросах к LiveTex Web API используются те же логин и пароль, что указываются при входе в Личный кабинет.

LiveTex Web API поддерживают 2 типа аутентификации:

1. Стандартная аутентификация **HTTP Basic Authentication**.
2. **OAuth 2.0** по сценарию «Resource Owner Password Credentials Grant». http://tools.ietf.org/html/rfc6749#section-4.3

1.1.1. HTTP Basic Authentication

При вызове методов достаточно посылать логин и пароль в режиме HTTP Basic Authentication в соответствии со спецификацией RFC 2617.
http://www.ietf.org/rfc/rfc2617.txt

Пример вызова с помощью утилиты cURL

```
curl -u USERNAME:PASSWORD https://apiv2.livetex.ru/v2/sites
```

Пример HTTP-запроса

```
GET /v2/sites
Host: apiv2.livetex.ru
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Поскольку логин и пароль в этом режиме передаются практически в открытом виде, настоятельно рекомендуется выполнять запросы с этим типом аутентификации по протоколу HTTPS.

В случае неуспеха сервер вернет стандартный ответ с кодом ошибки 401 Unauthorized.

Пример HTTP-ответа в случае неуспеха

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
WWW-Authenticate: Basic realm="LiveTex API"
Content-Length: 30

{
      "code": "401"
      "message": "Unauthorized"
}
```

В составе параметров метода вызова может быть указан параметр suppress_response_codes, который предписывает возвращать HTTP-код 200, для ситуации требующих кодов 4xx. В этом случае код и описание статуса ответа будет приведено в ключах code и message соответственно.

Пример вызова с параметром suppress_response_codes

```
curl -u USERNAME:PASSWORD "https://apiv2.livetex.ru/v2/sites?suppress_response_codes=true"
```

Пример HTTP-ответа в случае неуспеха и suppress_response_codes

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
WWW-Authenticate: Basic realm="LiveTex API"
Content-Length: 30

{
      "code": "401"
      "message": "Unauthorized"
}
```

1.1.2. OAuth 2.0. Resource Owner Password Credentials Grant

Спецификация OAuth 2.0 предполагает 4 сценария аутентификации.
http://tools.ietf.org/html/rfc6749

LiveTex Web API в данный момент поддерживает только сценарий «Resource Owner Password Credentials Grant». В дальнейшем планируется реализация и всех остальных сценариев.

Все сценарии аутентификации/авторизации OAuth 2.0 предполагают получение ключа доступа – access_token, который в дальнейшем передается при вызове методов API.

Для получения access_token необходимо отправить HTTP-запрос методом POST по адресу https://apiv2.livetex.ru/v2/oauth2/token.

Параметры

Название	+	Тип	Описание
grant_type	+	string	Тип OAuth-аутентификации. В данном случае – строка «password».
username	+	string	Email пользователя, зарегистрированного в Личном кабинете.
password	+	string	Пароль пользователя.
suppress_response_codes		string	Признак, предписывающий возвращать HTTP-код 200, для ситуации требующих кодов 4xx. По умолчанию – false. Если true, то в составе возвращаемых данных будут присутствовать поля "code" и "message".

Возвращаемые данные

Название	Тип	Описание
access_token	string	Ключ доступа.
token_type	string	Тип ключа доступа. В настоящее время всегда – "bearer".
expires_in	numeric	Время жизни ключа доступа в секундах с момента его получение. По прошествии этого времени следует повторить запрос ключа доступа.
code	numeric	Код ответа. Возвращается только, если указано suppress_response_codes = true.
message	string	Текстовое описание кода ответа. Возвращается только, если указано suppress_response_codes = true.

Пример oAuth 2.0 аутентификации с использованием утилиты cURL

```
curl https://apiv2.livetex.ru/v2/oauth2/token \
      -d "grant_type=password" \
      -d "username=USERNAME" \
      -d "password=PASSWORD"
```

Пример эквивалентного HTTP-запроса

```
POST /v2/oauth2/token
Host: apiv2.livetex.ru

grant_type=password&username=USERNAME&password=PASSWORD
```

Пример HTTP-ответа при успешной аутентификации

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token": "2YotnFZFEjr1zCsicMWpAA",
    "token_type": "bearer",
    "expires_in": 3600
}
```

В случае неуспеха сервер вернет ответ с кодом ошибки 400 Bad Request в соответствии с разделом 5.2 спецификации OAuth 2.0.
http://tools.ietf.org/html/rfc6749#section-5.2

Возможные коды ошибок:

- invalid_request – некорректный запрос, например: не указано обязательное поле.
- invalid_grant – отказ в доступе, например: неверно указаны логин и/или пароль.

Пример ответ в случае неуспешной аутентификации

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "error":"invalid_grant"
}
```

Пример ответ при указании suppress_response_codes = true

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "code":"400",
    "message":"Bad Reguest",
    "error":"invalid_grant"
}
```

1.2. Вызов методов

Методы Web API вызываются путем посылки HTTP-запросов разных типов на адрес:

http://apiv2.livetex.ru/v2.

При авторизации в режиме OAuth 2.0 access_token может быть передан одним их следующих путей:

1. В системном параметре **access_token**.
2. В Cookies.
3. В заголовке «Authorization», например: «Authorization: Bearer **ACCESS_TOKEN**».

Система просматривает источники access_token в порядке их перечисления в указанном списке и использует первый найденный.

Параметры можно указывать как в URL, так и в теле запросов. Если параметр присутствует в обоих местах, то принимается во внимание значение, указанное в теле запроса.

Имена методов и параметров нечувствительны к регистру символов.

Вызовы LiveTex Web API возможны в двух стилях:

1. RESTful
2. RPC (Remote Procedure Call)

1.2.1. Вызов в стиле RESTful

LiveTex Web API существует в виде набора ресурсов, соответствующих основным сущностям системы:

- /v2/sites
- /v2/operators
- /v2/departments
- /v2/buttons
- И т.д.

К этим ресурсам можно обращаться в соответствии с принципами архитектурного стиля REST, который предполагает использование основных типов запросов HTTP для выполнения операций получения данных(GET), их добавления (POST), частичного изменения (PATCH) и удаления (DELETE).

Технически эти операции приводят к неявному вызову соответствующих методов сущности:

- GET – **list -** получение списка объектов сущности;
- GET с указание ID – **show -** получение данных указанного объекта;
- POST – **add** – добавление объекта;
- PATCH – **update** – изменение данных объекта (обязательно указание ID объекта);
- DELETE – **delete** – удаление объекта (обязательно указание ID объекта).

Получение списка сайтов:

```
curl https://apiv2.livetex.ru/v2/sites \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

Получение данных сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/sites/12345 \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

Создание нового сайта:

```
curl https://apiv2.livetex.ru/v2/sites \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "url=www.mysite.com"
```

Изменение сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/sites/12345 \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "is_embedded_chat=false" \
      –X PATCH
```

Удаление сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/sites/12345 \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –X DELETE
```

Сущности системы могут иметь специализированные бизнес-методы. Доступ к этим методам в стиле RESTful предоставляется по следующей схеме:

https://apiv2.livetex.ru/v2/**{ENTITY}**/**{METHOD}**

Например:

https://apiv2.livetex.ru/v2/leads/setCompleted

При этом параметры метода могут передаваться как в строке адреса, так и в теле запроса.

В таком режиме методы получения данных (list, show и т.д.) допускается вызвать HTTP-запросами типа GET и POST. Методы, добавляющие, изменяющие и удаляющие данные – только HTTP-запросами типа POST.

1.2.2. Вызов в стиле RPC

Вызовы в стиле RPC предполагают явное указание вызываемого метода в системном параметре «method». При этом методы получения данных допускается выполнять HTTP-запросами типа GET и POST, а методы добавления, изменения и удаления данных – только HTTP-запросами типа POST.

Получение списка сайтов:

```
curl https://apiv2.livetex.ru/v2/ \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "method=sites.list"
```

Получение данных сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/ \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "method=sites.show" \
      –d "id=12345"
```

Создание нового сайта:

```
curl https://apiv2.livetex.ru/v2/ \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "method=sites.add" \
      –d "url=www.mysite.com"
```

Изменение сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/ \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –d "method=sites.update" \
      –d "is_embedded_chat=true"
```

Удаление сайта с id=12345:

```
curl https://apiv2.livetex.ru/v2/ \
   -H "Authorization: Bearer ACCESS_TOKEN" \
   –d "method=sites.delete" \
   –d "id=12345"
```

1.2.3. Системные параметры вызовов

При вызове методов LiveTex Web API, кроме параметров вызываемого метода, можно указать ряд дополнительных системных параметров.

Таблица 1. Системные параметры

Название	Тип	Описание
format	string	Формат возвращаемых результатов. Возможные значения: • **json** (по умолчанию)

		• **xml** Значение нечувствительно к регистру символов.
callback	string	Имя JavaScript-функции для «упаковки» возвращаемого результата (JSONP). Принимается во внимание только для формата JSON.
method	string	Значение нечувствительно к регистру символов. **RESTful** Если указан для HTTP-запроса типа POST при обращении к API в стиле RESTful, переопределяет тип запроса для обработчика вызова. Возможные значения: PATCH, DELETE. Например: method=PATCH Используется в случаях, когда клиентская среда по каким-то причинам не может выполнять HTTP-запросы отличные от GET или POST. **RPC** При обращении к API в стиле RPC содержит имя вызываемого метода, например: method=sites.list
access_token	string	Ключ доступа. Может быть использован в случаях, когда клиентская среда по каким-то причинам не может установить значение HTTP-заголовка.
suppress_response_codes	boolean	Признак, предписывающий возвращать HTTP-код 200, для ситуации требующих кодов 4xx. По умолчанию – false. Если true, то в составе возвращаемых данных будут присутствовать поля "code" и "message", отражающие истинный статус ответа, а также дополнительные поля, которые в обычном обычно передаются в заголовке ответа.
include_deleted	boolean	Если true, то методы list и show будут возвращать данные по удаленным объектам. В полях, содержащих массивы объектов связанных сущностей, также будут возвращены удаленные объекты. **ВНИМАНИЕ!** За редким исключением удаленные объекты нельзя указать в качестве значений полей сложных типов. См. также раздел «Работа с удаленными данными».

1.2.4. Структура и форматы результатов

Ответ сервера с результатами выполнения запроса в зависимости от ситуации может содержать следующие ключи:

- **results** – результаты выполнения метода. Возвращается всегда. Если в описании метода указано «Метод ничего не возвращает», то значением results будет null.
- Ключи с метаданными, специфичными для конкретного метода. Например, ключ «total», возвращаемый методами «list».
- **code** – код ответа (см. нижеследующий раздел).
  Возвращается только, если в параметрах вызова указано suppress_response_codes = true.
- **message** – текстовое описание кода ответа. Если все в порядке – OK.
  Возвращается только, если указано suppress_response_codes = true.
- Ключи с данными, специфичными для конкретного типа ошибки, например: errors для кода ответа 422 или reply-after для кода 429.

LiveTex Web API поддерживает 2 формата результатов: **JSON** (по умолчанию) и **XML**.

Требуемый формат результата можно указать с помощью системного параметра «**format**» при вызове метода.

Получение списка сайтов в JSON

```
curl "https://apiv2.livetex.ru/v2/sites" \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

```
{
    "total": 2,
    "results": [
        {
            "id":10000087,
            "url":"mysite1.com"
        },
        {
            "id":10000012,
            "url":"mysite2.com"
        }
    ]
}
```

Получение списка сайтов в JSON с использованием callback (JSONP)

```
curl "https://apiv2.livetex.ru/v2/sites?callback=myFunction" \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

```
myFunction({
    "total": 2,
    "results": [
        {
            "id":10000087,
            "url":"mysite1.com"
        },
        {
            "id":10000012,
            "url":"mysite2.com"
        }
    ]
});
```

Получение списка сайтов в XML

```
curl "https://apiv2.livetex.ru/v2/sites?format=xml" \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
      <total>2</total>
      <results>
       <item>
             <id>10000087</id>
             <url>mysite1.com</url>
       </item>
       <item>
             <id>10000012</id>
             <url>mysite2.com</url>
       </item>
      </results>
</response>
```

Получение данных категории быстрых сообщений с id=12345 в JSON

```
curl "https://apiv2.livetex.ru/v2/quickmessagecategories/12345" \
      -H "Authorization: Bearer ACCESS_TOKEN"
```

```
{
    "results": {
        "id": 12345,
        "title": "Приветствия",
        "is_global": true,
        "site": {
            "id": 10000087,
            "url": "mysite1.com"
        },
        "created_at": "2013-12-21T12:43:05+04:00",
        "updated_at": "2013-12-21T12:43:05+04:00"
    }
}
```

Удаление сайта с id=12345 в JSON

```
curl "https://apiv2.livetex.ru/v2/sites/12345" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -X "DELETE"
```

```
{
      "results": null
}
```

Удаление сайта с id=12345 в XML

```
curl "https://apiv2.livetex.ru/v2/sites/12345?format=xml" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -X "DELETE"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
      <results/>
</response>
```

Если при вызове метода указан параметр suppress_response_codes = true, то в возвращаемом объекте будут также ключи code и message.

Удаление сайта с id=12345 и suppress_response_codes=true

```
curl "https://apiv2.livetex.ru/v2/sites/12345?suppress_response_codes=true" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -X DELETE
```

```
{
      "code": 200,
      "message": "OK",
      "results": null
}
```

1.2.5. Работа со связанными сущностями

Связанные сущности – это атрибуты сложного типа. См. раздел «Сложные типы данных». Например, поле employee (Сотрудник) у чата.

Методы типа list и show, возвращают поля сложных типов в виде объекта со свойствами соответствующего связанного типа.

Методы типа list по умолчанию возвращают ограниченное количество полей связанных сущностей – так называемые, стандартные поля. Однако в параметре fields можно перечислить поля, которые требуется вернуть и для связанной сущности. Подробнее о параметре fields можно узнать в разделе «1.3.2 Параметр «fields».

Метод show также имеет параметр fields, но если он не указан, то возвращается полный набор полей основной сущности. Для полей связанных сущностей при этом, как и в случае методов типа list, возвращаются только стандартные поля. «Дозаказать» дополнительные поля для связанных сущностей можно с помощью все того же параметра fields.

В этом механизме предусмотрено ограничение на вложенность сущностей. Возвращаются только поля непосредственно связанных сущностей. Для полей сущностей, связанных со связанными, возвращается только поле id.

1.2.6. Коды ответов

В обычном режиме работы LiveTex Web API сигнализирует о статусе обработки вызова через HTTP Status Code в заголовке HTTP-ответа.
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

- **200** – все в порядке, результаты смотри в теле ответа. Структура результатов будет соответствовать спецификации конкретного метода.
- **4xx** – клиентская ошибка, подробности, если применимо, приведены в структурированном виде в теле ответа.
- **5xx** – серверная ошибка, повторить запрос чуть позже. На тело ответа обращать внимания не следует.

Исключением является указание системного параметра вызова suppress_response_codes=true.

В этом случает этих ситуациях HTTP Status Code 200 будет возвращаться и для ситуаций 4xx, а в теле ответа будут дополнительно присутствовать ключи «code» и «message.

Таблица 2. Коды ответов 4xx

Код	Сообщение	Описание
400	Bad Request	Некорректный HTTP-запрос.
401	Unauthorized	В доступен отказано, требуется аутентификация.
403	Forbidden	Аутентифицированный пользователь не имеет прав на выполнение указанного метода или доступа к указанному объекту. Этот код означает, что указанный метод или объект существует, но текущему пользователю доступ к нему запрещен.
404	Not Found	Метод или объект не найден. Возвращается также при попытке получить доступ к объекту другого аккаунта. То есть «чужие» объекты в принципе не существуют в

		контексте запроса.
405	Method Not Allowed	Указанный тип HTTP-запроса не поддерживается.
422	Validation Failed	Возникла ошибка при проверки значений параметров.
429	Too Many Requests	См. раздел "1.2.9 Ограничения на частоту вызовов".

Таблица 3. Дополнительные поля ответа для кода 422 Validation Failed

Название	Тип	Описание
errors	Object	Объект с полями - именам параметров. В каждом таком ключе содержится массив с кодами ошибок проверки данных. См. раздел "1.2.7. Проверка корректности вводимых данных".

Таблица 4. Дополнительные поля ответа для кода 429 Too Many Requests

Название	Тип	Описание
retry-after	numeric	Количество секунд, через которые можно повторить запрос. См. раздел "1.2.9 Ограничения на частоту вызовов".

Пример тела ответа для кода 403

```
{
    "code": 403,
    "message": "Unauthorized"
}
```

1.2.7. Проверка корректности вводимых данных

Если при выполнении проверки корректности значение параметров вызова метода обнаружены ошибки, то система вернет код 422.

При этом в структуре данных об ошибке будет присутствовать ключ «errors», содержащий перечень параметров и обнаруженные ошибки.

Пример ответа для кода 422

```
{
    "code": 422,
    "message": "Validation Failed",
    "errors": {
        "fields": {
             "error": "invalid",
             "message": "Указаны несуществующие поля: tile"
        }
    }
}
```

Таблица 5. Перечень ошибок проверки корректности вводимых данных

Ошибка	Описание
missing	Не указано или указано пустое значение обязательного поля.
already_exists	В системе уже есть запись с указанным значением. Применимо только к полям, предполагающим уникальность.
out_of_range	Значение выходим за рамки допустимого диапазона или не входит в набор допустимых значений. Для простых строковых полей означает, что строка по длине меньше или больше, чем разрешено.
invalid	Некорректное значение. См. Документацию на соответствующее поле.

1.2.8. Передача параметров сложных типов

Некоторые методы Web API имеют параметры сложных типов, например:

1. Методы HoldRules.add и HoldRules.update имеют параметр hold_messages типа Array.<HoldMessage>.
2. Метод Payers.add имеет параметр requisites типа Requisites.
3. Методы InvitationRules.add и InvitationRules.update имеют параметр locations типа Array.<Location>.
4. И т.д.

Значения полей таких параметров передаются в запросе в квадратных скобках:

param_name[param_field_name]=value

здесь:

- param_name – название параметра,
- param_field_name – название поля сложного параметра.

Поля сложных типов параметров также могут быть сложными и иметь свои поля. Уровень вложенности полей не ограничен.

При описании массивов индекс элемента также указывается в квадратных скобках. Нумерация начинается с 0.

Пример вызова метода HoldRules.add

```
curl "https://apiv2.livetex.ru/v2/holdrules" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -d "title=Одно сообщение и перевод" \
      -d "hold_messages[0][text]=Оператор скоро вам ответит" \
      -d "hold_messages[0][send_after]=15" \
      -d "is_transfer=true" \
      -d "transfer_after=30"
```

Пример вызова метода Payers.add

```
curl "https://apiv2.livetex.ru/v2/payers" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -d "payer_type=ru_person" \
      -d "currency=RUB" \
      -d "requisites[payment_type]=emoney" \
      -d "requisites[first_name]=Григорий" \
      -d "requisites[last_name]=Красов" \
      -d "requisites[patronymic]=Васильевич"
```

Пример вызова метода InvitationRules.add

```
curl "https://apiv2.livetex.ru/v2/invitationrules" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -d "title=Первое посещение из России" \
      -d "action=chat" \
      -d "welcome=Могу ли я вам чем-то помочь?" \
      -d "is_active=true" \
      -d "is_new_visitor=true" \
      -d "locations[0][country][id]=123" \
      -d "locations[0][city][id]=3245"
```

В некоторых случаях допустимо пустое значение сложного типа. Для указания пустого значения в методах типа add и update следует указать параметр с пустыми квадратными скобками и пустым значением.

Пример вызова метода InvitationRules.update, сбрасывающий набор locations

```
curl "https://apiv2.livetex.ru/v2/invitationrules" \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      –X PATCH \
      -d "locations[]="
```

1.2.9. Ограничения на частоту вызовов

Частота вызовов подсчитывается для access_token. Максимум **10 запросов за 10 секунд**.

Если превышено, то вернется ответ с кодом **429 Too Many Requests**. При этом в поле **Retry-After** будет указано количество секунд, через которые можно повторить запрос.

Некоторые методы имеют более жесткие ограничения по частоте вызова. Мы оставляем за собой право менять эти ограничения по своему усмотрению. Разработчикам предлагается ориентироваться на код ответа 429 и значение Retry-After.

1.2.10. Ролевая модель

Набор функциональных возможностей пользователя сервиса LiveTex, а также область видимости данных, зависят от назначенной ему роли. В Web API это отражается на возможности вызвать какой-то метод, или получить/изменить определенные данные.

Каждый вызов Web API производится в контексте конкретного пользователя, которому назначена определенная роль. Если настройки роли не дают соответствующих полномочий, то метод либо вернет только те данные, которые доступны пользователю, либо вернет сообщение об ошибке с кодом 403 Forbidden, если в рамках данной роли вызов метода не разрешен или запрашиваемый объект не входит в область видимости данных роли.

Набор возможных ролей задается системно и не может быть отредактирован пользователями.

В нижеследующей таблице приведен перечень системных ролей на момент подготовки данного документа.

Таблица 6. Текущий перечень ролей пользователей

Название роли	Код роли
Администратор	admin
Администратор	admin_partner
Управляющий	chief
Управляющий	chief_partner
Руководитель операторов	manager
Оператор	operator

В конкретном аккаунте доступно подмножество этого списка ролей или даже другие роли со специально настроенными полномочиями. Получить перечень действующих ролей можно с помощью метода **Roles.list**.

Одним из фундаментальных понятий ролевой модели является область видимости, которая определяет подмножество объектов, с которым допустимо выполнение конкретного метода.

Для упрощения описания области видимости вводятся некоторые служебные термины:

- **Свой сайт** - это сайт, указанный в поле managed_sites сотрудника, вызывающего метод.
- **Свой отдел** - это отдел, указанный в поле managed_departments сотрудника, вызывающего метод.
- **Свой сотрудник** – в данный момент, это любой сотрудник.
- **Свой сценарий вовлечения** - это сценарий вовлечения не связанный ни с одним сайтом, либо связанный только со своими сайтами.
- **Свой сценарий удержания** - это сценарий удержания не связанный ни с одним сайтом, либо связанный только со своими сайтами.
- **Свой диалог (чат, звонок, лид)** – это диалог, для которого верно любое из следующих утверждений:
  1) диалог состоялся на своем сайте (в том числе, удаленном) И отдел диалога не указан;
  2) диалог состоялся на своем сайте (в том числе, удаленном) И адресован в свой отдел (в том числе удаленный);
  3) диалог распределен на сотрудника, вызывающего метод.
- **Свой тикет** – это тикет, созданный сотрудников, вызывающим метод.
- **Связанный сайт** – это сайт, с которым сотрудник связан через механизмы маршрутизации диалогов: сотрудник указан в поле employees сайта ИЛИ сотрудник указан в поле employees отдела, который связан с сайтом с признаком связи is_visible = true.

- **Связанный отдел** - это отдел, с которым сотрудник связан через механизмы маршрутизации диалогов: сотрудник указан в поле employees отдела.
- **Адресованный диалог (чат, звонок, лид)** – это завершенный диалог, обработанный текущим сотрудником, либо еще не завершенный диалог, для которого верно любое из следующих утверждений:
    1) сотрудник-участник диалога еще не определен И диалог инициирован на связанном сайте И отдел диалога не указан;
    2) сотрудник-участник диалога еще не определен И диалог инициирован на связанном сайте И отдел диалога входит в число связанных отделов;
    3) сотрудник-участник диалога определен и это сотрудник, вызывающий метод.

ПРИМЕЧАНИЕ
Для сотрудников с ролью, имеющей признак is_admin = true (см. раздел Error: Reference source not found. Error: Reference source not found), «своими» являются все объекты системы вне зависимости от значения полей managed_sites и managed_departments, участия сотрудника в диалоге и т.д.
Для таких сотрудников соответствующие признаки is_managed всегда будут true для всех объектов.

1.2.11. Работа с удаленными данными

Объекты некоторых сущностей никогда полностью не удаляются из системы. В таком случае метод delete просто помечает объект как удаленный. В составе полей таких сущностей присутствует поле **is_deleted**, которое имеет значение true у удаленных объектов.

По умолчанию удаленные объекты нельзя получить с помощью методов list и show. Метод list их не возвращает, а метод show выдаст ошибку 404. Данные удаленных объектов возвращаются методами Web API только в следующих случаях:

1. **Если удаленный объект является значением поля связанной сущности.**
   Например, поле site в сущности Chat. Даже если сайт удален, при запросе данных чата в поле сайт будут данные удаленного сайта.
   Однако если значение поля является массивом объектов (например, поле Employee.managed_sites), то удаленные объекты НЕ будут присутствовать среди его элементов.
2. **Если это явно предполагается методом.**
   Например, методы Stats.employees, Stats.departments, Stats.sites.

Для прямого получения данных удаленных объектов через методы list и show можно воспользоваться системным параметром вызова **include_deleted**. Этот параметр предписывает методу расширить область видимости, в том числе, и на удаленные объекты. Это также оказывает влияние на выборку объектов в массиве, являющимся значением поля (например, поле Employee.managed_sites). При include_deleted=true там будут присутствовать и удаленные объекты.

Параметр include_deleted оказывает свое действие на выборку данных, но не на методы их изменения. Указать id удаленного объекта в качестве значения поля при вызове методов add или update НЕ получится даже при использовании include_deleted. Исключение составляют лишь некоторые поля, в описании которых явно указано на такую возможность.

1.3. Выборка данных

У большинства методов, возвращающих списки (массивы) записей есть параметры, позволяющие настроить выборку:

- **q** - критерий выборки.
- **fields** - определяет возвращаемые поля.
- **sort** - определяет порядок сортировки.
- **offset** и **limit** - позволяют организовать постраничную выборку данных.

1.3.1. Поиск. Параметр «q»

Методы list оснащены параметром q, который определяет критерий выборки данных.

Значение q - это строка, в которой можно использовать поисковые выражения вида:

first_name = Олег is_active = true age > 20 age < 30

Каждый метод list поддерживает свой собственный набор поле, которые можно использовать в таких выражениях (см. спецификацию конкретного метода «list»).

Критерий поиска в общем случае строится по маске:

{field_name}{comparison_operator}{value}

где

{field_name} - имя поля.

{comparison_operator} - оператор сравнения, например: =, !=, <, > и т.д.

{value} - значение критерия поиска.

Если значение критерия выборки содержит пробелы, то его надо заключить в двойные кавычки.

Разделителем условий является символ пробела. В рамках одного условия пробелы игнорируются.

Все компоненты критерия (field_name, comparison_operator, value) обязательны. Если какого-то компонента в одном из критериев не будет, то все условие выборки будет сочтено некорректным и метод вернет сообщение об ошибке.

Все критерии поиска связаны между собой условием "И".

Допускается использовать до 3 условий для одного и того же параметра. Все последующие условия будут проигнорированы.

Возможные значения оператора сравнения зависят от типа поля. Соответствующий перечень приведен в нижеследующей таблице.

Таблица 7. Допустимые операторы сравнения для различных типов данных

Тип данных	Оператор сравнения	Примеры
numeric	=, >, >=, <, <=, !=	• msg_count>5 • msg_count>=6 msg_count<=10
boolean	=	• is_paid=true • is_paid=0
datetime	=, >, <	• created_at>2014-01-11 created_at<2014-01-15 • created_at=lastmonth • answered_at=today Если дата указана без времени, то предполагается 00:00:00.
time	=, >, <	• period>09:00 period<18:00 Вариант "period > 23:59:59", несмотря на некоторую нелогичность, определяет последнюю секунду суток.
string	=	• first_name=Петр • title="Отдел доставки"

		Поиск выполняется без учета регистра символов.
enum	=	• action=lead,vary Этот типа критерия выборки характерен для полей сущностей, предполагающих ограниченный набор возможных строковых значений. При поиске элементы в списке указанных значений связываются по ИЛИ. То есть в приведенном примере будут записи с action = lead и action = vary.
idlist	=, !=	• site_ids=321654,193675 Этот тип критерия, как правило, используется для выборки по системным идентификаторам объектов. Если значение соответствующего поля не является обязательным, может потребоваться выборка записей, у которых соответствующее поле не указано. Для обозначения такого варианта следует использовать ключевое слово "**undefined**".

1.3.2. Параметр «fields»

Параметр fields присутствует во всех методах типа list и show и предоставляет возможность указать поля сущностей, которые требуется вернуть в результатах.

Поля связанных сущностей перечисляются в круглых скобках после названия сущности, например: **fields=id,title,created_at,employee(id,first_name,last_name)**.

Если в значении параметра fields указано несуществующее поле, то метод вернет ошибку с кодом 422.

Пример

```
curl "https://apiv2.livetex.ru/v2/departments" -G \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -d "fields=id,title,employees(id,first_name,last_name,email)"
```

Результат

```
{
    "total": 2,
    "results": [
        {
            "id":1234,
            "title":"Отдел доставки"
            "employees": [
                {
                    "id":12345,
                    "first_name":"Иван",
                    "last_name":"Голубев"
                    "email":"ivan@golubev.com"
                },
                {
                    "id":10986,
                    "first_name":"Григорий",
                    "last_name":"Петров"
                    "email":"greg@petrov.com"
                }
            ]
        },
        {
            "id": 1235,
            "title": "Отдел поддержки",
            "employees": [
                {
                    "id":1024,
                    "first_name":"Ольга",
                    "last_name":"Иванова"
                    "email":"olga@ivanova.com"
                },
                {
                    "id": 10986,
                    "first_name":"Григорий",
                    "last_name":"Петров"
                    "email":"greg@petrov.com"
                }
        }
    ]
}
```

1.3.3. Параметр «sort»

При вызове методов типа list можно указать порядок сортировки результатов с помощью параметра sort. Например:

sort=first_name:a,last_name:a,created_at:d

":a" – указывает на прямой порядок сортировки, ":d" – на обратный.

Сортировка возможна далеко не по всем полям сущностей. Перечень доступных полей и порядков сортировки приведены в спецификации параметра sort в описании соответствующего метода list.

Пример

```
curl "https://apiv2.livetex.ru/v2/departments" -G \
      -H "Authorization: Bearer ACCESS_TOKEN" \
      -d "sort=title:a"
```

Результат

```
{
    "total": 2,
    "data": [
        {
            "id":1234,
            "title":"Отдел доставки"
        },
        {
            "id": 1235,
            "title":"Отдел поддержки",
        }
    ]
}
```

1.3.4. Параметры «limit» и «offset»

В LiveTex Web API существует ограничение на количество возвращаемых данных в одном запросе в методах list. Оно определяется значением параметра **limit**. Значение по умолчанию – 50. **Максимальное значение – 100**.

Чтобы получить остальные записи, можно использовать параметр **offset** – количество записей, которое необходимо пропустить. Значение по умолчанию – 0. Максимальное значение не ограничивается.

В результатах вызова метода list всегда присутствует поле **total** - общее количество записей, удовлетворяющих критерию поиска. Если значение этого поля больше limit, то это означает, что есть смысл воспользоваться параметром offset, чтобы получить остальные записи.

1.4. Загрузка файлов

Некоторые сущности системы имеют поля типа "file". В качестве значения такого атрибута в методах add и update следует указывать ключ загрузки – **file_token**.

Ключ загрузки возвращается при загрузке файла методом POST в формате «multipart/form-data».

Адрес загрузки: http://apiv2.livetex.ru/v2/upload

Таблица 8. Параметры, передаваемые при загрузке файла

Название	+	Тип	Описание
field	+	string	Наименование поля, для которого загружается файл в формате entity.field. Пример: employee.photo
file	+	file	Загружаемый файл.

Параметр «field» позволяет выполнить проверку загружаемого файла на соответствие требованиям соответствующего поля. Если проверка завершится неуспешно, то сервер вернет соответствующий код ошибки.

Если в описании поля не сказано иное, то максимальный размер файла не должен превышать **5MB**.

Пример ответа при успешной загрузке

```
{
    "results":{
        "file_token": "afe147e45bb1448eb55d0a834ade3124"
    }
}
```

Пример ответа при ошибке проверки данных

```
{
    "code": 422,
    "message": "Validation Failed",
    "errors": {
        "file": [
             "invalid"
        ]
    }
}
```

Ошибка 422 invalid генерируется также при попытке загрузить файл нулевого размера.