Stalker API v2

revision 87 (2015-07-28)

Полезные источники

1. Формат запроса и ответа

API middleware Stalker построен по архитектуре REST и предоставляет доступ ко всем основным ресурсам системы. Ресурс может обрабатывать четыре операции, каждая из которых соответствует определенному HTTP методу:

HTTP method	Действие
GET	Получение коллекции или информации об элементе коллекции
PUT	Обновление элемента коллекции или замена всей коллекции
POST	Добавление элемента в коллекцию
DELETE	Удаление коллекции или элемента коллекции

Таблица 1.1. Соответствие HTTP метода действию в API.

1.1 Запрос

Запрос к API должен содержать название ресурса и, если необходимо, один или несколько идентификатора элементов.

```
<HTTP METHOD> /<RESOURCE>/<IDENTIFIER>
```

1.1.1 Постраничный вывод

Если необходимо работать с постраничными данными при работе с коллекцией, то нужно указать смещение (o*ffset*) и количество элементов (*limit*). Для получения числа элементов в коллекции нужно запросить элемент *count* у необходимого ресурса.

Пример 1.1. Запрос количества элементов в коллекции users:

```
GET /users/count
```

Пример 1.2. Запрос 20 элементов коллекции users начиная с 41:

```
GET /users?offset=40&limit=20
```

1.1.2 Поиск

Некоторые коллекции поддерживают поиск. Строка, по которой осуществляется поиск, должна передаваться в параметре *q*.

Пример 1.3. Поиск по видео по названию:

```
GET /video?q=Terminator
```

1.1.3 Локализация

Управление локализацией ответа осуществляется через HTTP заголовок *Accept-Language.*

Заголовок *Accept-Language*	Язык локализации
en-US	Английский
ru-RU	Русский

1.1.4 Поддержка экранов с разными разрешениями

Данная возможность позволяет получать изображения, максимально адаптированные к используемому разрешению экрана. Требуемое разрешение передается в дополнительном HTTP заголовке (*UA-resolution)* запроса к API. Поддерживаются четыре разрешения.

ldpi (120 dpi)	mdpi (160 dpi)	hdpi (240 dpi)	xhdpi (320 dpi)

Таблица 1.1.4 Поддерживаемые разрешения для изображений

В заголовке *UA-resolution* должно передаваться значение dpi.

Пример 1.1.4 Запрос списка каналов с иконками для разрешения 320 dpi

```
-> GET /users/1/tv-channels
-> UA-resolution: 320
```

1.2 Ответ

Ответ может передаваться в формате JSON, JSONP, XML или PLAIN TEXT в зависимости от заголовка *Accept* в запросе HTTP. Кодировка ответа - *UTF-8*.

Заголовок *Accept*	Формат ответа
application/json	JSON
application/javascript	JSONP
text/xml	XML
text/plain	PLAIN TEXT

Таблица 1.2. Соответствие HTTP заголовка Accept и формата ответа

Структура ответа:

```
{
    "status" : "OK" || "ERROR",
    "results" : [],
    "error" : "" // Поле содержит строку сообщения об ошибке.
}
```

1.3 Обработка HTTP статуса ответа

HTTP Статус	Описание
200 OK	Успешная обработка запроса к API
201 Created	Успешное создание коллекции или элемента коллекции
204 No Content	Успешное удаление, создание или обновление, либо ресурс существует, но нет данных
400 Bad Request	Ошибка в запросе
401 Unauthorized	Требуется авторизация
403 Forbidden	Нет прав для доступа к ресурсу
404 Not Found	Ресурс не найден
405 Method Not Allowed	Ресурс не поддерживает данный метод. разрешенные методы перечисляются в заголовке Allow
406 Not Acceptable	Не поддерживается переданный в заголовке Accept media type (см табл. 1.2)
500 Internal Server Error	Ошибка сервера

Таблица 1.3. Описание возможных статусов ответа

1.4 Примеры

Пример 1.4. Использование HTTP заголовков

```
GET /users/1/settings
Accept: application/json
Accept-Language: ru-RU
Authorization: OAuth oauth_token="ad180jjd733klru7"
...
HTTP/1.1 200 OK
Content-Type: application/json

{"status":"OK", "results":{"parent_password":"0000",...}}
```

2. Авторизация и аутентификация

Для авторизации используется протокол OAuth 2.0 - авторизация по логину и паролю (Resource Owner Password Credentials). Возможны две схемы аутентификации - MAC (каждый запрос к серверу должен иметь цифровую подпись - сигнатуру) и Bearer (в каждом запросе передается *oauth_token*).
По умолчанию middleware использует Bearer схему аутентификации.

При не авторизованном обращении к API возвращается соответствующий HTTP статус и заголовок с параметрами авторизации:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer
...
```

После успешной авторизации сервер возвращает oauth_token, который должен передаваться в заголовке каждого запроса к API.

```
GET /users/1/settings
Authorization: Bearer SLDJd4mg43cjQfElUs3Qub4L6xE
...
```

С течением времени данный token может стать не валидным, тогда необходимо заново произвести процедуру авторизации.

2.1 Протокол авторизации

При регистрации приложения разработчику выдается *client_id* и *auth_url*.

- *client_id* - идентификатор клиента (приложения);
- *auth_url* - url, страницы авторизации (https).

2.1.1 Авторизация клиентских приложений (Implicit Grant)

1. открытие встроенного браузера со страницей авторизации сервиса (https);
2. пользователь вводит логин и пароль и подтверждает выдачу прав приложению;
3. браузер редиректится на страничку (заглушку), в url которой (после #) содержится access_token и другие атрибуты;
4. приложение перехватывает переадресацию и получает access_token;

Пример 2.1. Открытие страницы авторизации

```
-> GET /authorize?response_type=token&client_id=s6BhdRkqt3 HTTP/1.1
-> Host: example.com
```

- *response_type* - должно быть token
- *client_id* - идентификатор клиента

Пример 2.2. Редирект после успешной авторизации пользователя (получение *user_id*). Тип доступа - MAC

```
<- HTTP/1.1 302 Found
<- Location: https://example.com/rd#access_token=FJQbwq9&token_type=mac&
mac_key=adijq39jdlaska9asud&mac_algorithm=hmac-sha-256&expires_in=3600&user_id=1
```

Пример 2.3. Редирект после успешной авторизации пользователя. Тип доступа - Bearer

```
<- HTTP/1.1 302 Found
<- Location:
https://example.com/rd#access_token=FJQbwq9&token_type=bearer&expires_in=3600&user_id=
```

```
1
```

Пример 2.4. Редирект после неудачной авторизации пользователя

```
<- HTTP/1.1 302 Found
<- Location: https://example.com/cb#error=access_denied
```

2.1.2 Авторизация по логину и паролю (Resource Owner Password Credentials)

Данный способ рекомендуется применять только тогда, когда нет возможности реализовать вариант, описанный в п. 2.1.1. Суть его заключается в том, что пользователь вводит свой логин и пароль не в браузере на странице авторизации сервера, а напрямую в приложении. Приложение отсылает *POST* запрос с необходимыми реквизитами на страницу авторизации, а в ответ получает *oauth_token* или информацию об ошибке.

Параметр	**Описание**
grant_type	тип, значение - password
username	логин
password	пароль

Таблица 2.1. Обязательные параметры авторизации

Параметр	**Описание**
mac	MAC адрес устройства
serial_number	серийный номер устройства
model	название модели устройства
client_type	тип клиента (STB, Android, Android_STB, iOS, SmartTV)
version	строка с версией прошивки устройства, версией портала, версией плеера. Пример: ImageVersion: …; ImageDescription: ...; ImageDate: ...; Hardware version: …; PORTAL version: ...; API Version: ...
device_id	идентификатор устройства
device_id2	второй идентификатор устойства
hw_version	строка формата hash.random.timestamp

Таблица 2.2. Специфические параметры для авторизации

Пример 2.5. Отправка данных авторизации

```
->POST /token HTTP/1.1
->Host: server.example.com
->Content-Type: application/x-www-form-urlencoded

->grant_type=password&username=<username>&password=<password>
```

Пример 2.6.1. Ответ при успешной авторизации (получение user_id). Bearer схема (**по умолчанию**).

```
<-HTTP/1.1 200 OK
<-Content-Type: application/json
```

```
<-Cache-Control: no-store

<-{
<-    "access_token":"SlAV32hkKG",
<-    "token_type":"bearer",
<-    "expires_in":3600,
<-    "refresh_token":"40521a5647587290a18f0ba0920a5e0c",
<-    "user_id":1
<-}
```

Пример 2.6.2. Ответ при успешной авторизации (получение user_id). MAC схема.

```
<-HTTP/1.1 200 OK
<-Content-Type: application/json
<-Cache-Control: no-store

<-{
<-    "access_token":"SlAV32hkKG",
<-    "token_type":"mac",
<-    "mac_key":"adijq39jdlaska9asud",
<-    "mac_algorithm":"hmac-sha-256",
<-    "expires_in":3600,
<-    "refresh_token":"40521a5647587290a18f0ba0920a5e0c",
<-    "user_id":1
<-}
```

Пример 2.7. Ответ при неудачной авторизации

```
<-HTTP/1.1 200 OK
<-Content-Type: application/json
<-Cache-Control: no-store

<-{
<-    "error":"access_denied"
<-}
```

2.1.3 Обновление авторизации

По истечении срока действия ключа *access_token*, длительность жизни которого указывается в *expires_in* (в секундах), сервер на любой запрос возвращает ответ со статусом 401, означающий необходимость обновления ключа. Однако для клиента поле *expires_in* является чисто информационным, процедуру обновления токена необходимо производить исключительно при получении ошибки авторизации во время работы API (статус 401). Для незаметного для пользователя обновления авторизации используется параметр *refresh_token*, с помощью которого можно получить новый ключ.

```
->POST /token HTTP/1.1
->Host: server.example.com
->Content-Type: application/x-www-form-urlencoded

->grant_type=refresh_token&refresh_token=40521a5647587290a18f0ba0920a5e0c
```

```
<-HTTP/1.1 200 OK
<-Content-Type: application/json
<-Cache-Control: no-store

<-{
<-    "access_token":"SlAV32hkKG",
<-    "token_type":"bearer",
<-    "expires_in":3600,
<-    "refresh_token":"40521a5647587290a18f0ba0920a5e0c",
<-    "user_id":1
<-}
```

2.1.4 Разработка и отладка

Для упрощения разработки и отладки Stalker API введены временные API ключи (*api_key*). Разработчик может запросить такой ключ и использовать его в каждом запросе к API, при этом проходить OAuth авторизацию уже не требуется.

Пример 2.8. Использование *api_key*

```
GET /users/1/settings?api_key=c2e07700a1dd67743c0efe5b8b7d6903
…
```

В реальном приложении использование *api_key* недопустимо.

2.2 Схемы аутентификации

2.2.1 Цифровая подпись каждого обращения к API (MAC)

Для предотвращения подделки запросов к API каждое обращение должно подписываться цифровой подписью (сигнатурой). Используется MAC схема аутентифиации доступа. Рекомендуется использовать только в при необходимости повышенной безопасности.

Пример 2.9. Подпись обращения к API сигнатурой.

```
GET /resource/1?b=1&a=2 HTTP/1.1
Host: example.com
Authorization: MAC id="h480djs93hd8",
                   ts="1336363200",
                   nonce="dj83hs9s",
                   mac="SLDJd4mg43cjQfElUs3Qub4L6xE="
```

- *id* - *access_token*, получаемый при успешной OAuth авторизации
- *ts* - unix timestamp клиента
- *nonce* - уникальная строка, генерируемая клиентом.

- *mac* - сигнатура, которая высчитывается по алгоритму *mac_algorithm* из нормализованных параметров HTTP запроса и секретного ключа *mac_key.*

2.2.2 Токен на предъявителя (Bearer)

Упрощенная схема, используемая по умолчению в Stalker middleware, в которой достаточно передавать oauth_token в каждом запросе. В связи со сложностью работы MAC схемы рекомендуется использовать именно Bearer.

Пример 2.10. Bearer схема

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer 7Fjfp0ZBr1KtDRbnfVdmIw
```

3. Ресурсы

Набор ресурсов и действий, с которыми можно производить операции, зависит от привилегий, полученных при авторизации. В данном разделе описаны все возможные действия.

3.1 Ресурс USERS

Ресурс предоставляет доступ к коллекции пользователей и персонализированным данным конкретного пользователя.

Формат: /users/<user_id>/<resource>
Формат: /users/current

- *user_id* - ID пользователя (получаемый при авторизации)
- *resource*:
    a. *settings* - хранилище настроек
    b. *tv-channels* - персонализированная список ТВ каналов
    c. *tv-favorites* - список ID избранных каналов
    d. *video* - персонализированный список видео
    e. *video-favorites* - список ID избранных video
    f. *ping* - контроллер для обновления последнего времени доступа. Необходимо для того, чтобы знать состояние устройства online/offline. Рекомендуется вызывать данный контроллер раз в 120 секунд.
    g. *media-info* - контроллер для обновления информации о текущем проигрываемом контенте (ID, тип, конец проигрывания). Поддерживаемые типы: *tv-channel*, *video*, *karaoke, tv-archive*
    h. *message* - получение сообщений от middleware
    i. *modules* - список доступных модулей

j. message-history - история системных сообщений

Идентификатор	*ID пользователя*

Таблица 3.1. Описание ресурса users

Название поля	**Описание**
id	id пользователя
account	номер лицевого счета
status	административный статус (1 - включена, 0 - выключена)
mac	MAC адрес устройства
fname	Полное имя пользователя (опционально)
phone	Телефон (опционально)
tariff_plan	Тарифный план (опционально)
end_date	Дата окончания предоставления услуг (опционально)
account_balance	Баланс на счету пользователя (опционально)

Таблица 3.2. Описание полей ресурса users

Пример 3.2. Получение информации о пользователе по ID пользователя

```
-> GET /users/1
<- {"status":"OK", "results":{"id":1, "mac":"00:1A:79:00:15:B3", "status":0,
"account":1553, "logo":"\/stalker_portal\/c\/template\/logo.png",
"background":"\/stalker_portal\/new\/launcher\/img\/1080\/bg.jpg"}}
```

Пример 3.3. Получение настроек по ID пользователя

```
-> GET /users/1/settings
<- {"status":"OK", "results":{"parent_password":"0000"}}
```

Пример 3.4. Сохранение настроек

```
-> PUT /users/1/settings
-> parent_password=1111
...
```

Пример 3.4.2. Обновление времени доступа

```
-> GET /users/1/ping
...
```

Пример 3.4.3. Обновление информации о проигрываемом контенте (начало проигрывания канала)

```
-> POST /users/1/media-info
-> type=tv-channel&media_id=12
```

Пример 3.4.4. Обновление информации о проигрываемом контенте, удаление информации (остановка проигрывания)

```
-> DELETE /users/1/media-info
```

…

Пример 3.4.5. Получение информации об оплате

```
-> GET /users/1/payment-info
…
```

Пример 3.4.6. Получение пользовательского соглашения

```
-> GET /users/1/agreement
…
```

Пример 3.4.7. Получение условий использования

```
-> GET /users/1/tos
…
```

Пример 3.4.8. Получение списка доступных модулей

```
-> GET /users/1/modules
<- {"status":"OK","results":["media_browser","tv","apps","dvb"]}
```

3.2 Ресурс TV-CHANNELS

Ресурс предоставляет доступ к списку ТВ каналов.

Формат:
*(список)***: /tv-channels?mark=<favorite|default:>**

(персонализированный список): **/users/<user_id>/tv-channels?mark=<favorite|default:>**

- *mark* - метка для фильтрации, при значении *favorite* в персонализированном списке позвоялет выбрать только избранные каналы.
-

*(элемент)***: /tv-channels/<ch_id>**

- *ch_id* - ID канала

(*хранилище id последнего канала*): **/users/<user_id>/tv-channels/last**

Идентификатор	*id канала*

Таблица 3.3. Описание ресурса tv-channels

Название поля	**Описание**
id (int)	id канала
name (string)	название канала
genre_id (string)	id жанра
number (int)	номер канала
url (string)	url для проигрывания
archive (int 0-1)	флаг наличия ТВ архива
archive_range (int, hours)	глубина записи архива/time shift (с текущего момента)
pvr (int 0-1)	флаг разрешения записи ТВ канала
censored (int 0-1)	флаг ограничения по возрасту

favorite (int 0-1)	флаг избранного (для персонализированного списка)
logo (string)	url картинки логотипа канала согласно указанному разрешению или пустая строка
monitoring_status (int 0-1)	статус мониторинга. 0 - канал временно не доступен

Таблица 3.4. Описание полей ресурса tv-channels

Дополнительно: Ресурс поддерживает media type *audio/x-mpegurl*, который возвращает m3u плейлист.

Пример 3.5. Получение персонализированного списка всех ТВ каналов

```
-> GET /users/1/tv-channels
<- {"status":"OK", "results":[{"id":1, "name":"\u041c-TV
\u0423\u043a\u0440\u0430\u0438\u043d\u0430", "number":10,
"url":"rtp://239.1.1.60:1234", "archive":1, "censored":0, "favorite":1},...]}
```

Примечание: Если url канала пустой (пустая строка - `""`), то перед началом проигрывания необходимо произвести процедуру получения ссылки (ресурс tvchannel-link).

Пример 3.6. Получение списка избранных ТВ каналов (фильтрация общего списка)

```
-> GET /users/1/tv-channels?mark=favorite
```

Пример 3.6.1. Сохранение id последнего просмотренного канала для дальнейшего доступа к нему

```
-> PUT /users/1/tv-channels/last
-> ch_id=123
```

Пример 3.6.2. Получение id последнего просмотренного канала

```
-> GET /users/1/tv-channels/last
-> {"status":"OK","results":123}
```

3.3 Ресурс TV-FAVORITES

Данный ресурс позволяет управлять списком избранных каналов.

Формат: /users/<user_id>/tv-favorites

- *user_id* - ID пользователя

Пример 3.7. Получение списка ID избранных ТВ каналов

```
-> GET /users/1/tv-favorites
<- {"status":"OK", "results":[3,4,1,6,2]}
```

Пример 3.8. Сохранение списка ID избранных ТВ каналов

```
-> PUT /users/1/tv-favorites
-> ch_id=3,1,8,11
```

Пример 3.8.1. Очистка списка ID избранных ТВ каналов

```
-> PUT /users/1/tv-favorites
-> ch_id=
```

Пример 3.9. Добавление ТВ канала в избранное (добавляется в конец списка)

```
-> POST /users/1/tv-favorites
-> ch_id=3
```

Пример 3.9.1 Удаление ТВ канала из избранного

```
-> DELETE /users/1/tv-favorites/3
```

3.4 Ресурс TV-CHANNELS LINK

Ресурс позволяет получить прямую ссылку для проигрывания ТВ канала.

Формат: /tv-channels/<ch_id>/link

- *ch_id* - ID канала

Идентификатор	*id ТВ канала*

Таблица 3.5. Описание ресурса tv-channel link

Пример 3.10. Получение ссылки на канал с ID=3

```
-> GET /users/1/tv-channels/3/link
<- {"status":"OK", "results":"
http://192.168.1.165:1935/itv/ch3161.stream/playlist.m3u8?e57f8e65591ada28f9bc88ba5e2acbc0"}
```

Если рабочих ссылок для данного канала нет (например, временно отключены мониторингом), то возвращается ответ с HTTP статусом 404 и в теле ответа содержится описание ошибки.

Пример. Ошибка при запросе ссылки для проигрывания

```
-> GET /users/1/tv-channels/3/link
<- HTTP/1.1 404 Not Found
<- {"status":"ERROR","results":null,"error":"temporary_unavailable"}
```

Примечание: Если элемент коллекции TV имеет пустое поле url, то перед каждым проигрыванием этого канала необходимо проходить процедуру получения ссылки.

Пример 3.11. Получение ссылки на канал с ID=3 и указанием времени начала (time shift)

```
-> GET /users/1/tv-channels/3/link?start=1329133000
<- {"status":"OK", "results":"
```

http://192.168.1.165:1935/itv/ch3161.stream/playlist.m3u8?e57f8e65591ada28f9bc88ba5e2acbc0"}

3.5 Ресурс TV-GENRES

Предоставляет доступ к списку ТВ жанров и каналам, соответствующим указанному жанру.

Формат: /tv-genres/<tvgenre_id>/tv-channels

- *tvgenre_id* - ID жанра

Идентификатор	*id категории*
Поля для обновления	
Поля для добавления	

Таблица 3.6. Описание ресурса tv-genres

Название поля	**Описание**
id (string)	id жанра
title (string)	название жанра

Таблица 3.7. Описание полей ресурса tv-genres

Пример 3.12. Получение списка жанров ТВ

```
-> GET /tv-genres
<- {"status":"OK", "results":[{"id":"sport", "title":"sports"},...]}
```

Пример 3.13. Получение списка видео по ID ТВ жанра

```
-> GET /tv-genres/sport/tv-channels
...
```

3.6 Ресурс VIDEO

Ресурс предоставляет доступ в списку видео.

Формат:

(список): **/video?q=<search>**

- *q* - строка, по которой осуществляется поиск в списке. Поиск осу

(элемент): **/video/<video_id>**

(персонализированный список):
/users/<user_id>/video?mark=<favorite|not_ended|default:>&sortby=<name|default:>&so

rtby_desc=<added|default:added>

- *user_id* - ID пользователя
- *mark* - метка для фильтрации. При значении *favorite* в персонализированном списке позвоялет выбрать только избранные видео. При значении *not_ended* в персонализированном списке позволяет выбрать видео, просмотр которых не закончился.
- sortby - поле, по которому производится сортировка по возрастанию.
- sortby_desc - поле, по которому производится сортировка по убыванию

Идентификатор	*id видео*
Поля для обновления	
Поля для добавления	

Таблица 3.8. Описание ресурса video

Название поля	**Описание**
id (int)	id видео
name (string)	название
original_name (string)	оригинальное название
description (string)	описание
director (string)	режиссер
actors (string)	актеры
year (int)	год
genres (string)	жанры
genres_ids (array<string>)	id жанров
censored (int 0-1)	флаг ограничения по возрасту
cover (string)	url обложки
series (array)	номера серий
url (string)	url для проигрывания. Если url пустой, то необходимо использовать процедуру запроса url.
added (int, timestamp)	дата добавления
hd (int, 0-1)	качество
favorite (int 0-1)	флаг нахождения в списке избранного*
rating_kinopoisk (float)	рейтинг фильма Kinopoisk
rating_imdb (float)	рейтинг фильма IMDB
not_ended (int 0-1)	флаг недосмотренного видео
not_ended_episode (int)	номер недосмотренной серии (при наличии такой информации)
end_time (int)	время в секундах, на котором остановился просмотр (в случае not_ended=1)
downloadable (int, 0-1)	флаг доступности для скачивания

Таблица 3.9. Описание полей ресурса video

Примечание: Данный ресурс поддерживает поиск, который осуществляется одновременно по полям *name, original_name, description, actors, year, genres.*

Пример 3.14. Поиск по видео

```
-> GET /video?q=Терминатор
…
```

Пример 3.15. Получение персонализорованного списка видео (значащее поле *favorite*)

```
-> GET /users/1/video
…
```

Пример 3.16. Получение списка избранных видео

```
-> GET /users/1/video?mark=favorite
…
```

Пример 3.17. Получение количества видео

```
-> GET /video/count
…
```

Пример 3.18. Получение 20 элементов коллекции video начиная с 40-го элемента

```
-> GET /video?offset=40&limit=20
…
```

Пример 3.19. Пример сортировки по имени (в алфавитном порядке)

```
-> GET /video?sortby=name
…
```

Пример 3.20. Пример сортировки по времени добавления (новые выше)

```
-> GET /video?sortby_desc=added
…
```

3.7 Ресурс VIDEO-CATEGORIES

Ресурс предоставляет доступ к списку видео категорий.

Формат:

(список категорий): **/video-categories**

(список видео в категории): **/video-categories/<vcategory_id>/video**

- *vcategory_id* - ID категории

Идентификатор	*id категории*
Поля для обновления	
Поля для добавления	

Таблица 3.10. Описание ресурса video-categories

Название поля	**Описание**

id (string)	id категории
title (string)	название категории

Таблица 3.11. Описание полей ресурса video-categories

Пример 3.21. Получение списка категорий видео

```
-> GET /video-categories
<- {"status":"OK", "results":[{"id":"cartoon", "title":"Cartoon"},...]}
```

Пример 3.22. Получение списка видео по ID видео категории

```
-> GET /video-categories/cartoon/video
...
```

3.8 Ресурс VIDEO-GENRES

Ресурс предоставляет доступ к списку жанров видео.

Формат

(список жанров): **/video-genres**

(список жанров в категории): **/video-categories/<vcategory_id>/video-genres**

(список видео в жанре):

/video-categories/<vcategory_id>/video-genres/<vgenre_id>/video

- *vcategory_id* - ID видео категории
- *vgenre_id* - ID видео жанра

Примечание: Каждая категория может содержать уникальный список жанров.

Идентификатор	*id категории*
Поля для обновления	
Поля для добавления	

Таблица 3.12. Описание ресурса video-genres

Название поля	**Описание**
id (string)	id категории
title (string)	название жанра

Таблица 3.13. Описание полей ресурса video-genres

Пример 3.23. Получение списка жанров видео

```
-> GET /video-genres
<- {"status":"OK", "results":[{"id":"comedy", "title":"comedy"},...]}
```

Пример 3.24. Получение списка видео по ID видео жанра

```
-> GET /video-genres/comedy/video
```

```
...
```

Пример 3.25. Получение списка жанров для категории и затем получение видео по ID видео жанра из видео категории

```
-> GET /video-categories/cartoon/video-genres/
...
-> GET /video-categories/cartoon/video-genres/comedy/video
...
```

3.9 Ресурс VIDEO LINK

Ресурс позволяет получить прямую ссылку для проигрывания видео.

Формат
(фильм): **/video/<video_id>/link**
(сериал): **/video/<video_id>/episodes/<episode_number>/link**

- *video_id* - ID видео
- *episode_number* - номер серии

Идентификатор	*id ТВ канала*

Таблица 3.14. Описание ресурса video-link

Пример 3.26. Получение ссылки для видео с ID=1

```
-> GET /video/1/link
<- {"status":"OK",
"results":"http://192.168.1.165:1935/vclub/the_terminator.mpg/playlist.m3u8?397fa9a3e4
360fd539e1d83493235542"}
```

Пример 3.27. Получение ссылки для второй серии видео с ID=3

```
-> GET /video/3/episodes/2/link
<- {"status":"OK",
"results":"http://192.168.1.165:1935/vclub/house_md_s01e03.mpg/playlist.m3u8?397fa9a3e
4360fd539e1d83493235542"}
```

Примечание: Если элемент коллекции VIDEO имеет пустое поле url, то перед каждым проигрыванием этого канала необходимо проходить процедуру получения ссылки.

3.10 Ресурс VIDEO-FAVORITES

Данный ресурс позволяет управлять списком избранных видео.

Формат: /users/<user_id>/video-favorites

- *user_id* - ID пользователя

Пример 3.28. Получение списка ID избранных видео

```
-> GET /users/1/video-favorites
<- {"status":"OK", "results":[3,4,1,6,2]}
```

Пример 3.29. Сохранение списка ID избранных видео

```
-> PUT /users/1/video-favorites
-> video_id=3,1
```

Пример 3.29.1. Очистка списка ID избранных видео

```
-> PUT /users/1/video-favorites
-> video_id=
```

Пример 3.30. Добавление видео ролика в избранное (добавляется в конец списка)

```
-> POST /users/1/video-favorites
-> video_id=3
```

Пример 3.30.1. Очистка списка ID избранных видео

```
-> DELETE /users/1/video-favorites/3
```

3.11 Ресурс VIDEO NOT-ENDED

Ресурс позволяет управлять списком недосмотренного видео.

Формат: /users/<user_id>/video/<video_id>/not-ended

- *user_id* - ID пользователя
- *video_id* - ID видео

Идентификатор	*id видео*
Поля для обновления	*end_time, episode*
Поля для добавления	*end_time, episode*

Таблица 3.15. Описание ресурса video not-ended

Название поля	**Описание**
id (int)	id категории
end_time (int)	время, на котором остановлен просмотр (секунды)
episode (int)	серия сериала, по умолчанию 0

Таблица 3.16. Описание полей ресурса video not-ended

Пример 3.31. Добавление/обновление информации о недосмотренном видео

```
-> PUT /users/1/video/12/not-ended
```

```
-> end_time=147
```

Пример 3.32. Добавление/обновление информации о недосмотренной серии сериала

```
-> PUT /users/1/video/15/not-ended
-> end_time=400&episode=1
```

Пример 3.33. Удаление информации о недосмотренно фильме (фильм просмотрен)

```
-> DELETE /users/1/video/15/not-ended
```

3.12 Ресурс EPG

Ресурс предоставляет доступ к программе ТВ передач.

Формат

(за промежуток времени): **/tv-channels/<ch_id>/epg?from=<from_time>&to=<to_time>**

(фиксированное кол-во передач): **/tv-channels/<ch_id>/epg?next=5**

- *ch_id* - ID ТВ канала или список ID каналов через запятую
- *from* - начало промежутка (unix timestamp, секунды)
- *to* - конец промежутка (unix timestamp, секунды)
- *next* - количество передач (включая текущую)

Название поля	**Описание**
id (int)	ID передачи
start (int timestamp)	время начала передачи
end (int timestamp)	время окончания передачи
name (string)	название передачи
in_archive (int, 0-1)	находится в ТВ архиве, доступен для проигрывания (см. EPG-LINK)
downloadable (int, 0-1)	флаг доступности для скачивания

Таблица 3.17. Описание ресурса EPG

Пример 3.34. Получение программы передач на ТВ канал за указанный промежуток

```
-> GET /tv-channels/1/epg?from=1329133000&to=1329134000
```

…

Пример 3.35. Получение текущей программы и следующих четырех

```
-> GET /tv-channels/1/epg?next=5
```

…

3.13 Ресурс EPG-LINK

Ресурс позволяет получить прямую ссылку на записанную ТВ передачу (тв архив).

Формат: /epg/<program_id>/link

- *program_id* - ID ТВ передачи

Идентификатор	*id передачи (см. EPG)*

Таблица 3.18. Описание ресурса epg-link

Пример 3.36. Получение ссылки для программу передач с ID=1233

```
-> GET /epg/1233/link
<- {"status":"OK",
"results":"http://192.168.1.165:8080/tv_archive/1233.stream/playlist.m3u8?397fa9a3e436
0fd539e1d83493235542"}
```

3.14 Ресурс RADIO-CHANNELS

Ресурс предоставляет доступ к списку каналов радио.

Формат:
*(список)***: /radio-channels**

(персонализированный список): **/users/<user_id>/radio-channels**

Название поля	**Описание**
id (int)	id канала
name (string)	название канала
number (int)	номер канала
url (string)	url для проигрывания

Таблица 3.19. Описание полей ресурса radio-channels

3.15 Ресурс KARAOKE

Ресурс позволяет получить доступ к списку KARAOKE роликов.

Формат:
*(список)***: /karaoke?q=<search>**

- *q* - строка, по которой осуществляется поиск в списке (в названии, исполнителе, жанре)

(персонализированный список): **/users/<user_id>/karaoke**

Название поля	**Описание**
id (int)	id канала
name (string)	название ролика

performer (string)	исполнитель
genre (string)	жанр

Таблица 3.20. Описание полей ресурса karaoke

3.16 Ресурс KARAOKE LINK

Ресурс позволяет получить прямую ссылку на проигрывание караоке.

Формат: /karaoke/<karaoke_id>/link

- *ch_id* - ID канала

Идентификатор	*id karaoke ролика*

Таблица 3.21. Описание ресурса karaoke link

Пример 3.37. Получение ссылки на караоке с ID=3

```
-> GET /users/1/karaoke/3/link
<- {"status":"OK", "results":"
http://192.168.1.165:1935/karaoke/ch3161.stream/playlist.m3u8?e57f8e65591ada28f9bc88ba5e2acbc0"}
```

3.17 Ресурс PVR

Данный ресурс совместно с ресурсом TV-CHANNELS RECORD позволяет управлять записями ТВ каналов. Сами записи при этом сохраняются на сервере.

Формат: /users/<user_id>/pvr
Формат: /users/<user_id>/pvr/<record_id>

- *user_id* - ID пользователя
- *record_id* - ID записи, созданной с помошью ресурса TV-CHANNELS RECORD

Название поля	**Описание**
id (int)	id записи
name (string)	название записи
start_time (int, timestamp)	время начала
end_time (string, timestamp)	время окончания
ch_id (int)	ID канала
ch_name (string)	название канала
status (int)	состояние записи (0 - запланирована, 1 - в процессе записи, 2 - окончена)
downloadable (int, 0-1)	флаг доступности для скачивания

Таблица 3.21. Описание полей ресурса pvr

Пример 3.38. Получение списка записей пользователя

```
-> GET /users/1/pvr
<- {"status":"OK", "results":[{"id":9, "name":"", "start_time": 1371463910,
"end_time":1371465910, "ch_id": 4, "ch_name":"1+1", "status":2}]}
```

Пример 3.39. Остановка записи

```
-> POST /users/1/pvr/9/stop
<- {"status":"OK", "results":true]}
```

Пример 3.40. Удаление записи

```
-> DELETE /users/1/pvr/9
<- {"status":"OK", "results":true]}
```

3.18 Ресурс PVR LINK

Ресурс позволяет получить ссылку для проигрывания записи ТВ канала.

Формат: /users/<user_id>/pvr/<record_id>/link

- *user_id* - ID пользователя
- *record_id* - ID записи, созданной с помошью ресурса TV-CHANNELS RECORD

Пример 3.41. Получение ссылки для проигрывания записи

```
-> GET /users/1/pvr/9/link
<- {"status":"OK", "results":"http://192.168.1.100/records/1/9.mpg"}
```

3.19 Ресурс PVR-SUMMARY

Ресурс позволяет получить информацию о наличии свободного времени для записи.

Формат: /users/<user_id>/pvr-summary

- *user_id* - ID пользователя

Пример 3.42. Получение информации о овободном времени (в минутах) для записи

```
-> GET /users/1/pvr-summary
<- {"status":"OK", "results":{"total":"600", "free":"330"}}
```

3.19 Ресурс TV-CHANNELS RECORD

Ресурс позволяет запустить запись канала. Дальнейшее управление этой записью осуществляется через ресурс PVR.

Формат: /users/<user_id>/tv-channels/<ch_id>/record

- *user_id* - ID пользователя
- *ch_id* - ID ТВ канала

Идентификатор	*id видео*
Поля для добавления	*end_time*

Таблица 3.22. Описание ресурса video

Описание возвращаемого результата идентично ресурсу PVR.

Пример 3.42. Запуск процесса записи канала (с текущего момента, максимально разрешенной длительности)

```
-> POST /users/1/tv-channels/4/record
<- {"status":"OK", "results":{"id":9, "name":"", "start_time": 1371463910,
"end_time":1371465910, "ch_id": 4, "ch_name":"1+1", "status":1}}
```

Пример 3.43. Запуск процесса записи канала (с указанием времени конца записи)

```
-> POST /users/1/tv-channels/4/record
-> end_time=1371465910
<- {"status":"OK", "results":{"id":9, "name":"", "start_time": 1371463910,
"end_time":1371465910, "ch_id": 4, "ch_name":"1+1", "status":1}}
```

3.20 Ресурс EPG RECORD

Ресурс позволяет создать запись канала по программе передач - запланировать запись ТВ передачи. Дальнейшее управление этой записью осуществляется через ресурс PVR.

Формат: /users/<user_id>/tv-channels/<ch_id>/epg/<program_id>/record

- *ch_id* - ID ТВ канала
- *user_id* - ID пользователя
- *program_id* - ID ТВ передачи

Описание возвращаемого результата идентично ресурсу PVR.

Пример 3.44. Создание запланированной записи тв программы

```
-> POST /users/1/tv-channels/4/epg/213/record
<- {"status":"OK", "results":{"id":9, "name":"", "start_time": 1371463910,
"end_time":1371465910, "ch_id": 4, "ch_name":"1+1", "status":0}}
```

3.21 Ресурс MESSAGE

Ресурс позволяет получить текущее служебное и пользовательское сообщение. После запроса сообщения оно автоматически помечается как отправленное и при следующем запросе уже не возвращается.

Формат: /users/<user_id>/message

- *user_id* - ID пользователя

Название поля	**Описание**
id (int)	id сообщения
type (string)	тип сообщения
send_time (int, timestamp)	время создания сообщения
msg (string)	тело сообщения (опционально)

Таблица 3.23. Описание полей ресурса messages

Тип	**Описание**
send_msg	Текстовое сообщение
update_epg	На сервере обновилось EPG
reboot	Команда на выполнение перезагрузки
cut_off	Отключение аккаунта
cut_on	Включение аккаунта

Таблица 3.24. Список типов сообщений

3.22 Ресурс MESSAGE-HISTORY

Ресурс позволяет получить доступ к списку последних сообщений от сервера middleware.

Формат: /users/<user_id>/message-history

- *user_id* - ID пользователя

Название поля	**Описание**
id (int)	id сообщения
type (string)	тип сообщения
send_time (int, timestamp)	время создания сообщения
msg (string)	тело сообщения (опционально)

Таблица 3.23. Описание полей ресурса messages

Полезные источники

[1] REST API Design Rulebook, 978-1-4493-1050-9 | ISBN 10:1-4493-1050-8
[2] http://en.wikipedia.org/wiki/Representational_State_Transfer
[3] http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
[4] RFC2616
[5] API Best Practices Blog
[6] RESTful best practices
[7] API design and more
[8] OAuth
[9] RFC5849
[10] The OAuth 2.0 Authorization Protocol
[11] HTTP Authentication: MAC Access Authentication