REST API v1

Request

Structure of the request HTTP_METHOD /RESOURCE/[IDENTIFIERS]

HTTP_METHOD

HTTP methods which are supported.

```
GET    - data obtaining
PUT    - updating
POST   - creating
DELETE - deleting
```

IDENTIFIERS

Identifiers are devided by "," without “space”.

Answer

Structure's answer:

```
{
    "status" : "OK" || "ERROR",
    "results" : [],
    "error" : "" // The field contains an error message string.
}
```

Authentification and authorization

Basic HTTP authentication supported. All requests to the API must contain correct header *Authorization* when the authentification is on.

If there is no header or not correct data - server's answer will contain **HTTP status** 401 Unauthorized and message:

```
< {"status":"ERROR","results":"","error":"401 Unauthorized request"}
```

For successful authentification is sufficient to add in every request header to the API:

```
Authorization: Basic <base64encode("login":"password")>
```

Supported resources

STB

STB's managment - (it is recommended to use ACCOUNTS instead of STB)

Identifier: personal account, MAC adress (one personal account can contains more than one STB).

Supported methods: GET, PUT, DELETE, POST

Available fields for updating: status, additional_services_on, ls

Available fields for adding: mac, login, password, status, additional_services_on, ls

Fields discription

```
mac                    - MAC address
ls                     - personal account
login                  - login account number
status                 - admin status (1 - on, 0 - off)
online                 - STB status (1 - online, 0 - offline)
additional_services_on - Additive services connection status (for example
Videoclub, Karaoke, etc. (1 - on, 0 - off)
```

Example 1. Data obtain about other STBs.

```
> GET [API_URL]/stb
<
{"status":"OK","results":[{"mac":"00:1A:79:00:15:B3","status":0,"additional_
services_on":"0","ls":1553,"login":"1553","online":"1"},
{"mac":"00:1A:79:00:39:5E","status":0,"additional_services_on":"1","ls":3,"l
ogin":"3","online":"0"}]
```

Example 2. Data obtain about one STB using identifier.

```
> GET [API_URL]/stb/1553
<
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","status":1,"additional_
services_on":"1","ls":1553,"login":"1553","online":"1"}]}
```

Example 3. Data obtain about several STBs using identifier.

```
> GET [API_URL]/stb/1553,3
<
```

```
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","status":1,"additional_
services_on":"1","ls":1553,"login":"1553","online":"1"},
{"mac":"00:1A:79:00:21:40","status":0,"additional_services_on":"0","ls":3,"l
ogin":"3","online":"0"}]}
```

Example 4. Updating of STB's data using identifier.

```
> PUT [API_URL]/stb/1553
> status=1&additional_services_on=0
<
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","status":1,"additional_
services_on":"0","ls":1553}]}
```

Example 5. STB deleting using identifier.

```
> DELETE [API_URL]/stb/1553
< {"status":"OK","results":true}
```

Example 6. STB adding using login & password (for authentification).

```
> POST [API_URL]/stb/
> login=test&password=1234
<
{"status":"OK","results":{"mac":"","status":1,"additional_services_on":"1","
ls":"0","login":"test"}}
```

Example 7. STB adding without additive services.

```
> POST [API_URL]/stb/
> mac=00:1A:79:00:39:5E&additional_services_on=0
<
{"status":"OK","results":{"mac":"00:1A:79:00:39:5E","status":1,"additional_s
ervices_on":"0","ls":"0","login":""}}
```

ACCOUNTS

User account control (instead of the STB's resource).

Identifier: personal account, MAC adress (one personal account can contain only one STB)

Supported methods: GET, PUT, DELETE, POST.

Available fields for updating: password, full_name, account_number, tariff_plan, status, stb_mac.
Available fields for adding: login, password, full_name, account_number, tariff_plan, status, stb_mac.

Required fields: login.

Field discription:

```
login - Authorization login (Unique, required)
password - Authorization password
full_name - User's name (The first and the last name or organization name)
account_number - Account number
tariff_plan - Tariff plane identifier
status - Admin status (1 - on, 0 - off)
stb_mac - MAC address of the STB
stb_sn - Serial number of the device
stb_type - Model of the device
online - STB status (1 - online, 2 - offline)
ip - IP address of STB
version - row with firmware versions, portals, etc
subscribed - The list of identifiers of the optional packages in which
subscribed
comment - comment row, available for admin only
end_date - disabling date (while enable_internal_billing = true) in format
"Y-m-d H:i:s"
account_balance - balance row
last_active - time and date of last activity
```

Example 1. User's data requesting.

```
> GET [API_URL]/accounts/00:1A:79:00:39:5E
<
{"status":"OK","results":[{"login":"3210","full_name":"Test","account_number
":"123","tariff_plan":"FULL","stb_sn":"123345","stb_mac":"FF:FF:FF:FF:FF:FF"
,"stb_type":"MAG250","status":1,"subscribed":[]}]
```

Example 2. User's account creating.

```
> POST [API_URL]/accounts/
>
login=3210&password=1234&full_name=Test&account_number=123&tariff_plan=FULL&
status=1
< {"status":"OK","results":true}
```

Example 3. User's account updating.

```
> PUT [API_URL]/accounts/00:1A:79:00:39:5E
> tariff_plan=STANDART
< {"status":"OK","results":true}
```

Example 4. User's account deleting.

```
> DELETE [API_URL]/accounts/00:1A:79:00:39:5E
< {"status":"OK","results":true}
```

STB_MSG

Sending messages to the STB.

Identifier: personal account, MAC address (one personal account can contains more than one STB).

Supported methods: POST.

Available fields for updating: msg (Text message must be url encoded).

With successful adding results returned true.

Example 1. STB's data updating using identifiery.

```
> POST [API_URL]/stb_msg/1553
>
msg=%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B0%20%D1%81%D0%B2%D1%8F%D0
%B7%D0%B8
< {"status":"OK","results":true]}
```

SEND_EVENT

Sending events to STB.

Identificator: Account number, MAC (one Account number can contain more than one STB)

Supported methods: POST

Allowed fields for update: event (event name: *send_msg*, *reboot*, *reload_portal*, *update_channels*, *play_channel*, *update_image*, *cut_off*), msg (message should contain url encoded), ttl (TTL in seconds), need_reboot (reload flag, just for events *send_msg*), channel (channel number, just for events *play_channel*)

After successful adding in results returns true.

Example 1. Sending reboot events to all devices

```
> POST [API_URL]/send_event/
> event=reboot
< {"status":"OK","results":true]}
```

Example 2. Sendung play channel №10 events to certain STB

```
> POST [API_URL]/send_event/00:1A:79:00:00:00
> event=play_channel&channel=10
< {"status":"OK","results":true]}
```

Example 3. Sending turn off event to stb with acount number 12345, during this STB should be turned off (status=0) from resources *STB*, *ACCOUNTS* and *USERS*.

```
> POST [API_URL]/send_event/12345
> event=cut_off
< {"status":"OK","results":true]}
```

STB_MODULES

Controlling access to the modules (Main menu categories).

Identifier: personal account, MAC address (one personal account can contains more than one STB).

Supported methods: PUT, GET.

Available fields for updating: disabled (sections don't displayed, it is neccessary to reboot), restricted (limited access).

Example 1. Access deny to the vclub karaoke, radio module off.

```
> POST [API_URL]/stb_modules/00:1A:79:00:39:5E
> restricted[]=vclub&restricted[]=karaoke&disabled[]=radio
<
{"status":"OK","results":{"disabled":["radio"],"restricted":["vclub","karaok
e"]}}
```

Example 2. Turn off all restrictions.

```
> POST [API_URL]/stb_modules/00:1A:79:00:39:5E
> restricted[]=&disabled[]=
< {"status":"OK","results":{"disabled":[""],"restricted":[""]}}
```

ITV

Channel list obtaining.

Identifier: Channel ID.

Supported methods GET.

Fields discription

```
id      - Channel ID
name    - Channel name
number  - Channel number
base_ch - Base channel identifier
```

Example 1. All channel's data obtaining.

```
> GET [API_URL]/itv
< {"status":"OK","results":[{"id":"37","name":"\u041c-TV
\u0423\u043a\u0440\u0430\u0438\u043d\u0430","number":"64"}, ...]
```

ITV_SUBSCRIPTION

Channel subscription's managment.

Identifier: personal account, MAC address (one personal account can contains more than one STB).

Supported methods: GET, PUT.

Available fields for updating: sub_ch.

Fields discription

```
ls                    - personal account
mac                   - MAC address
sub_ch                - Channel ID list
additional_services_on - Additive services connection status
```

Example 1. Data obtaining about subscription of all STBs.

```
> GET [API_URL]/itv_subscription
<
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","sub_ch":["27"],"ls":15
53,"additional_services_on":"0"}, {"mac":"00:1A:79:00:15:B3","sub_ch":["27",
"29"],"ls":3,"additional_services_on":"0"}]}
```

Example 2. Data obtaining about subscription of concrete personal account.

```
> GET [API_URL]/itv_subscription/1553
<
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","sub_ch":["27"],"ls":15
53,"additional_services_on":"0"},{"sub_ch":["58"],"mac":"00:1A:79:00:15:B3",
"ls":"1553","additional_services_on":"0"}]}
```

Example 3. Data obtaining about subscription of concrete STB.

```
> GET [API_URL]/itv_subscription/00:1A:79:00:39:5E
<
{"status":"OK","results":[{"mac":"00:1A:79:00:39:5E","sub_ch":["27"],"ls":15
53,"additional_services_on":"0"}]}
```

Example 4. Data obtaining about all STBs on the personal account.

```
> PUT [API_URL]/itv_subscription/1553
> sub_ch[]=27&sub_ch[]=29&additional_services_on=1
<
{"status":"OK","results":[{"sub_ch":["27","29"],"mac":"00:1A:79:00:15:B3","l
s":"1553","additional_services_on":"1"},{"sub_ch":["27","29"],"mac":"00:1A:7
9:00:39:5E","ls":"1553","additional_services_on":"1"}]}

===== TARIFFS =====
```

Information about tariff plans and service packets

Identificator: ID of tariff plan

Supported methods: GET

Field description

```
id                    - ID of tariff plan
external_id           - ID for external systems
name                  - name
user_default          - default option of tariff plan
packages              - service packet's description
```

Description of service packet's field

```
id                    - ID of service packet
external_id           - ID for external system
name                  - name
type                  - type (video, tv, radio, module)
description           - description
all_services          - flag (it shows that packet contains all services)
service_type          - packet type (periodic or single)
rent_duration         - rent duration (для service_type = single)
```

```
optional                - flag (it shows that packet is optional)
```

Example 1. Getting the data about tariff plans

```
> GET [API_URL]/tariffs
<
{"status":"OK","results":[{"id":"10","external_id":"","name":"\u0422\u0430\u
0440\u0438\u0444 \u041f\u0430\u043a\u0435\u0442
+","user_default":"0","packages":[{"id":"10","external_id":"all_video"...
```

SERVICES_PACKAGE

Information about service packets

Identificator: ID of service packet

Supported methods: GET

Fields description

```
id                      - ID of service packets
external_id             - ID for external services
name                    - name
type                    - type (video, tv, radio, module)
description             - description
all_services            - flag (it shows that packet contains all services)
service_type            - packet type (periodic or single)
rent_duration           - rent duration (for service_type = single)
services                - service list or 'all' (it shows all services of
such type)
```

Example 1. Getting the data about tariff plans

```
> GET [API_URL]/services_package
<
{"status":"OK","results":[{"id":"10","external_id":"all_video","name":"\u041
2\u0441\u0435
\u0432\u0438\u0434\u0435\u043e","type":"video","description":"\u041f\u043e\u
043b\u043d\u044b\u0439 \u0434\u043e\u0441\u0442\u0443\u043f \u043a
\u0432\u0438\u0434\u0435\u043e\u043a\u043b\u0443\u0431\u0443","all_services"
:"1","service_type":"periodic","rent_duration":"0","services":"all"},...
```

ACCOUNT_SUBSCRIPTION

Account management on optional packets

Identificator: account, MAC address (one account can contain more then one STB)

Supported methods: GET, POST, PUT, DELETE

Available fields for updating: subscribed, subscribed_id, unsubscribed (for PUT), unsubscribed_id (for PUT)

Fields description

```
mac                    - MAC address
subscribed             - list of optional packets (external_id)
subscribed_id          - list of optional packets (internal_id)
```

Example 1. Data getting about subscribed optional packets

```
> GET [API_URL]/account_subscription/1553
<
{"status":"OK","results":[{"mac":"00:1A:79:00:15:B3","subscribed":["tv_6","t
v_3"]}]}
```

Example 2. Subscription on optional packet

```
> PUT [API_URL]/account_subscription/1553
> subscribed[]=tv_2
< {"status":"OK","results":true}
```

Example 3. Unsubscription on optional packets

```
> PUT [API_URL]/account_subscription/1553
> unsubscribed[]=tv_2
< {"status":"OK","results":true}
```

Пример 4. Information updating about all subscribed packets

```
> POST [API_URL]/account_subscription/1553
> subscribed[]=tv_2&subscribed[]=tv_3
< {"status":"OK","results":true}
```

Пример 5. Unsubscription on all optional packets

```
> DELETE [API_URL]/account_subscription/1553
< {"status":"OK","results":true}
```