Narayana Billing API Documentation.

Current version — 2.2

Introduction

Any applications (including client and server ones) and any equipment working in Narayana interact with the billing system (further — billing) by means of API.

For API access differentiation we use the system of confidential keys — identifiers of the client and/or server hardware and/or software interacting in any way with the billing.

The key can be **client** (demands obligatory user authorization) and **server with a certain access level** (doesn't demand authorization). The client key can be attached to a certain user (static authorization key) and ignore login and password parameters transferred in the query as well as demand authorization (*dynamic authorization key*)

For any key the following parameters can be set:

- API key (client) name
- Key authorization method
- Allowed IP addresses
- Access level
- Default user
- Allowed methods

Key name

If the API client is the affiliated telephone exchange, the client name is used for user attachment to automatic telephone exchange (SIP realm).

Otherwise the client name is used for internal accounting and doesn't have impact on anything.

Authorization method

Authorization method defines what parameter the user search and authorization will be run on. It can be **username** parameter for authorization by user name, **sip_extension** for authorization by the SIP login, **id** for authorization by user's caller ID in the database.

Please note that it's desirable to use unique parameter of authorization method (for example, user name). Use of potentially non-unique parameter (for example, CallerID) can lead to unpredictable consequences.

Allowed IP addresses

The list of IP addresses that API queries can be made from. Subnet mask usage is acceptable. Example: **178.1.1.*, 87.1.1.2**

Access level

It is used for the key type definition and for assignment of a certain access level.

Null access level — the client key. For this key user authorization is obliged (dynamic with password or static), otherwise execution of non-anonymous methods will be forbidden.

Not-null access level — the server key. For this key authorization with password isn't necessary, and the user parameter that query is run by is transferred as the login parameter. However you need to remember that if the owner of the server key isn't the owner of the user found on the login query parameter, the system will deny this query (except when the key has exclusive rights.)

Default user

User parameter (for example, **username** or **sip_extension**, depending on the authorizationmethod) attached to the key (static authorization). If this parameter is set, the billing will ignore query parameters **login** and **password** and the user found by this parameter will be authorized by default.

To make the query on behalf of another user belonging to the key owner it's necessary to transfer the username parameter.

Also sometimes this parameter is used for recording the user variable necessary for work with this key.

For recording the user variable it's necessary to add into **config.api*custom*identities** variable of the configuration file a new authorization method (for example, **payment*method*)** *and to specify in the key parameters* ***payment*method** as and the value specified as User will be available as **request.payment_method**.

Allowed methods

The list of methods available for this key. If this parameter isn't set, all methods are allowed.

Example: **getInfo, getValue**

Query and answer

1. API query initialization

   Query URL: **https://rdx.narayana.im/billing/api/${VERISON}/${APIKEY}/${METHOD}**

   Query method: **POST** or **GET**

2. Query main parameters

   - **login** — user name
   - **password** — password md5-hash (for dynamic authorization)
   - **username** — user name that requests to API are made from (if it's not specified, the current authorized user name is used)

3. API answer

   - Answer in JSON format.

     Answer on the qeury is presented as the JSON array with three variables:

     {'result'} — method result (boolean)

     {'response'} — answer or error code (string)

     {'description'} — error reason description (string)

     Example:

     ```
     GET https://rdx.narayana.im/billing/api/2.2/key/getInfo?login=admin


     {"result":false,"response":"E_AUTH_ERROR"}
     ```

   - Text answer

     Used for **sip_extension**, **id** and other specified in the configuration file authorization types (**config.api_plain_answer** option).

   Contains only the text answer or the error description (**response**)

4. Errors

   The following errors can be sent by the API server due to an incorrect qeury:

   - **HTTP 400 Invalid Request** — the query is incorrect

- **HTTP 401 Unauthorized** — the key is incorrect
- **HTTP 403 Forbidden** — access from this IP denied

NB! If you make the query with an incorrect and/or inactive key, access from your IP address will be locked for an indefinite term.

Error codes

Query initialization errors:

- **E_INCORRECT_API_VERSION** – The version of the server and the client don't match
- **E_UNKNOWN_METHOD** – Unknown method
- **E_UNDEFINED_INSTRUCTION** – Method can't be executed
- **E_INSUFFICIENT_ACCESS** – Insufficient rights for this method execution
- **E_MISSING_ARGUMENT** – Obliged argument missing
- **E_INVALID_ARGUMENT** – Invalid argument type
- **E_LUA_EXCEPTION** – Critical method execution error
- **E_AUTH_FAILED** – Incorrect user parameter and/or password
- **E_HAS_NO_ONE_TOLD_YOU_SHE_IS_NOT_BREATHING** – Entering the endless loop. You shouldn't ever see this error.

Query execution errors:

- **E_USER_LOCKED** – Users is locked
- **E_SQL_ERROR** – SQL-query error
- **E_NOT_EXIST** – Requested item doesn't exist in this context
- **E_DOES_NOT_BELONG_TO_YOU** – Requested item doesn't belong to you
- **E_DATA_ABORT** — Incorrect input
- **E_ASTERISK_QUERY_ERROR** – Asterisk query execution error
- **E_SIP_SERVER_NOT_EXIST** – SIP-server doesn't exist
- **E_WRONG_EXTENSION** – Invalid SIP login
- **E_TRANSACTIONS_LIMITED** – Financial transactions are limited
- **E_TRANSACTIONS_LOCKED** – Financial transactions are locked
- **E_ACCOUNT_HAS_USERS** – Unable to delete non-empty account
- **E_CANT_CREAsterisk_ACCOUNT** – Unable to create account
- **E_CARRIER_ERROR** – SMS sending error: channel returns error
- **E_DID_ORDER_FAIL** – DID-number order error
- **E_INITIAsterisk_CALL_FAIL** – Call initiation error
- **E_INSUFFICIENT_MONEY** – Insufficient money
- **E_INVALID_SMS_FROM** – Incorrect "From" parameter
- **E_INVALID_SMS_TO** – Incorrect "To" parameter
- **E_CODE_HAS_NO_BALANCE** – No money on this prepayed code
- **E_INVITE_LIMIT_REACHED** – Invitation limit is reached
- **E_MAXIMUM_PAYMENT** – Maximum payment sum exceeded
- **E_MINIMAL_PAYMENT** – Payment sum is less than minimal
- **E_MINIMAL_START_PAYMENT** – Start payment sum is less than minimal
- **E_TARGET_WHITELISTED** – IP adress is whitelisted
- **E_TRANSACTION_FAILED** – Finance transaction error
- **E_TTS_ERROR** – Text-to-Speech method error
- **E_UNROUTABLE** – Direction is not serviced
- **E_CANT_CREAsterisk_EXTENSION** — Unable to create SIP account
- **E_CANT_DELETE_EXTENSION** — Unable to delete SIP account
- **E_ALREADY_ON_THIS_TARIFF** — You are already serviced on this tariff
- **E_CONTACT_RESELLER** – This operation is unavailable for reseller users. Please, contact your service provider.

"User" object

When working with API billing systeman the "User" object is generally used.

In this section all object attributes and values it can take are described.

It's possible to request any attribute value by methods **getInfo()** (values of all attributes), **getValue()** (value of one attribute)

You may record values of attributes with functions corresponding to the attribute type:

- **updateUser()** — for user data
- **updateAccount()** — for account data
- **updateExtension()** — for SIP account data

Type description:

- *string* — string type;
- *int* — integer type;
- *float* — float type;

User

- **id**(int) *[read only]* — internal user identifier
- **username**(char[32]) *[read only]* — user name
- **password**(char[32]) — encrypted user password
- **manager**(char[32]) *[read only]* — user owner(manager)
- **accesslevel**(int) — user access level
  - **10** — new user (limits - minimal first deposit)
  - **11** — user
  - **22** — corporate user (has access to users and tariffs management)
  - **25** — partner (has rights of a corporate user plus access to the main routes)
  - **33** — system administrator
  - **44** — master system administrator
- **active(int)** — user status
  - **-1** — locked
  - **0** — deactivated (for example, low balance)
  - **1** — active
  - **2** — external calls block
  - **3** — all calls block
  - **99** — block protection
- **account**(int) — user account number
- **currency**(char[3]) — user currency codee
  - Default — **EUR**
  - Available currencies can be received by **getCurrencies()** method
- **language**(char[20]) — user interface language
- **timezone**(float) — user UTC timezone
- **allowed_rates**(char[20]) — list of user's allowed tariffs
- **allowed_call_len**(int) — user maximal allowed call length
- **notify_email**(char[256]) — email for notifications
- **notify_balance**(int) — notify about minimal balance by email
  - **0** — don't send
  - **1** — send
- **notify_balance_limit**(float) — balance limit for notifications (in system currency)
- **notify_did**(int) — number of days before number rent expiration notification
  - **-1** — don't notify
- **notify_ticket**(int) — send notifications about ticket response receiving
  - **0** — don't send
  - **1** — send
- **invites**(int) — accessable invitations number
- **comment**(char[1024]) — user comment *(for access level >= 33)*

SIP & DISA account

- **disa_pin**(int) — PIN-code for DISA access
- **disa_trusted_cli**(char[32]) — trusted phone number for simplified DISA access
- **sip_extension**(char[32]) — SIP account (5 or 7 characters)
- **sip_password**(char[32]) — SIP password decrypted
- **sip_server**(char[128]]) — SIP-server adress
- **sip_address**(char[128]]) — SIP-server adress (DNS)
- **sip_callerid**(char[32]) — current caller ID
- **sip_language**(char[20]) — internal phone command language
- **sip_rates_id**(int) — tariff identifier for this SIP login
- **sip_rates_options**(char[128]) — tariff identifier options for this SIP login
- **sip_srtp**(int) — SRTP enctypting
  - **0** — off
  - **1** — on forced
- **sip_transport**(char[20]) — available transport for SIP-server connection
  - **udp,tcp,tls** — allow UDP / TCP / TLS
  - **udp,tcp** — allow UDP / TCP
  - **tls** — TLS only
  - **udp** — UDP only
  - **tcp** — TCP only
- **sip_max_call_len**(int) — maximal call length for this SIP login
- **sip_reach**(int) — SIP-login call availability
  - **0** — unavailable
  - **2** — available
- **sip_redirect**(char[64]) — redirect incoming calls on this SIP-login
- **sip_force_callback**(int) — forced call rejection and trusted number Callback
- **sip_realm**(char[16]) *[read only]* — SIP-server name
- **sip_pitch_shift**(int) — voice distortion tone (-1 — off)
- **sip_host**(char[32]) — host authorization IP-adress (billing v2.3-rt)
- **sip_options**(char[128]) — Asterisk peer additional options(billing v2.3-rt)

Account

- **balance**(float) — the user current balance in system currency (EUR)
  - Please use **transferMoney()** method for transferring money to the user
- **overdraft**(float) — user current credit limit in system currency (EUR)
- **refill_allowed**(int) — allow financial transactions
  - **0** — financial transactions forbidden
  - **1** — anonymous financial transactions allowed
  - **2** — all financial transactions allowed
- **postpaid**(int) — postpaid payment system
  - **0** — off (prepaid system)
  - **1** — on (postpaid system; monthly invoice)
- **billing_name**(char[100]) — full name (for the invoice)
- **billing_address**(char[200]) — address (for the invoice)
- **billing_email**(char[100]) — e-mail for billing messages
- **billing_bank_account**(char[200]) — bank account data for financial transactions

Virtual parameters

These user parameters serving for information processing convenience are available only to reading and aren't recorded in the database.

- **actualbalance** — relevant balance in system currency (EUR) taking into account the overdraft
- **displaycurrency** — chosen currency (sign)
- **displaybalance** — current balance in user currency
- **displayoverdraft** — credit limit in user currency
- **displaynotifybalancelimit** — balance limit for notification in user currency
- **unread_tickets** — number of unread tickets in internal ticket system

---

API functions

Obligatory parameters are highlighted in bold.
If optional parameter has default value, it is specified in [square brackets]

Types description:

- *string* — string type;
- *ustring* — string type with strict values; allowed values are specified in {braces};
- *num* — numerical type;
- *bool* — boolean type (can take "true" or "false" values);

If the type isn't specified — parameter can have any type.

In (brackets) after the method name the necessary access level for this method execution is specified.
Methods with the null access level are anonymous and can be executed without authorization.

In API query examples the API version **2.2** and the key "fffffffff" are used and also authorization data are excluded.
Please make sure you use relevant data before query execution!

*NB!*except for the section **"Work with the user"** results in "Query examples" will be specified without

    {result: true, response: "something"}

but will contain only **«response»** value.

If in "Returns" "*true*/*false*" is specified, then the function has no answer but only result of execution **"result"** (*true* or *false*), but at the same time **"response"** may contain the error code in case of failed execution.

User

getInfo(1) — get user data

Return: JSON array with user data Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getInfo?login=admin
```

```
{"parameter": "value (see "User object" section)",
"sip_extensions": [ {"SIP_account_parameter": "value ((see "User object" section)"}, ... ],
"did_numbers:" [ {"DID_number_parameter": "value (see "Direct numbers" functions)"}, ... ],
"sim_cards:" [ {"SIM_card_attribute": "value (see "SIM-card" Functions)"}, ...],
"broadcasts": [ {"broadcasting_news_parameter": "value (see "News and broadcasting messages" functions)"}, ... ] }
```

getValue(1) — get user parameter value

Return: parameter value Arguments:

- **property**(string) — requested parameter

Query example:

```
GET /billing/api/2.2/fffffffff/getValue?property=balance
```

```
1.00
```

updateUser(1) — update user data

Return: *true* / *false*
Arguments:

- web_password(string) — new user password (md5-hash)
- accesslevel(num) — access level
- active(num) — status
- account(num) — account number
- currency(string) — currency code
- language(string) — interface language
- timezone(num) — UTC timezone
- allowed_rates(string) — list of allowed tariffs
- allowed_call_len(num) — maximal allowed call length (secs)
- notify_email(string) — e-mail for notifications
- notify_balance(num) — notify about minimal balance by email
- notify_did(num) — i% number of days before number rent expiration notification
- notify_ticket(num) — send notifications about ticket response
- notify_balance_limit(num) — minimal balance for notification
- invites(num) — accessable invitations number
- comment(string) — user comment

Query example:

```
GET /billing/api/2.2/fffffffffff/updateUser?username=test&active=0
```

```
true
```

Routing

getRoutes(22) — get list of accessable routes

Return: JSON array with the list of routes and their data
Arguments:

- update(bool) — update requested routes balance *[false]*
- type(ustring{all,sip,sms,default}) — requested routes type *[all]*
- all(bool) — request routes belonging to corporate users *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getRoutes?update=true
```

```
[{"id": route ID,
"prefix":"route SIP-prefix",
"cli": Caller ID type (1 – RU only, 2 – international, 99 – without limits),
"route":"route name",
"owner":"route owner",
"enabled": status (0 or 1),
"type":"  type (default, sms, sip)",
"lastupdate":"last balance update",
"clientlastupdate":"client last balance update",
"balance":"balance"}]
```

getRouteInfo(22) — get route data by its ID

Return: Route data(JSON)
Arguments:

- **id**(num) — route ID

Query example:

```
GET /billing/api/2.2/fffffffffff/getRouteInfo?id=1
```

```
{"id": route ID,
"prefix":"route SIP-prefix",
"cli": Caller ID type (1 – RU only, 2 – international, 99 – without limits),
"route":"route name",
"owner":"route owner",
"enabled": status (0 or 1),
"type":"  type (default, sms, sip)",
"lastupdate":"last balance update",
"clientlastupdate":"client last balance update",
"balance":"balance"}
```

addRoute(22) — add new route to routing

Return: *true* / *false*
Arguments:

- **route**(string) — route name
- **type**(ustring{sip,sms}) — route type
- **sip_server**(string) — SIP-server for route adding
- **prefix**(string) — SIP-prefix for route adding (outlineN/) or SMS-handler name
- host(string) — host address (domain or IP), if necessary *[]*
- user(string) — user name for authorization *[]*
- fromuser(bool) — use user name as **fromuser** in peer settings *[false]*
- pass(string) — password for authorization *[]*
- callerid(num) — caller ID (1 — RU only, 2 — international, 99 — without limits) *[2]*
- balance_handler(string) — balance handler name *[]*
- balance_username(string) — user name for balance handler *[]*
- balance_password(string) — password for balance handler *[]*

Query example:

```
GET /billing/api/2.2/fffffffffff/addRoute?route=test&type=sms&sip_server=sip.serv.er&prefix=ourhandler
```

```
true
```

updateRoute(22) — update route parameters

Return: *true* / *false*
Arguments:

- **id**(num) — route ID
- route(string) — route name
- prefix(string) — SIP-prefix (outlineN/) or SMS handler name
- host(string) — host address (domain or IP)
- user(string) — user name for authorization
- pass(string) — password for authorization
- callerid(num) — caller ID (1 — RU only, 2 — international, 99 — without limits)
- balance_handler(string) — balance handler name
- balance_username(string) — user name for balance handler
- balance_password(string) — password name for balance handler

Query example:

```
GET /billing/api/2.2/fffffffffff/updateRoute?id=1&route=changedName
```

```
true
```

updateRouteState(22) — update route status

Return: *true* / *false*
Arguments:

- **id**(num) — route ID
- **state**(bool) — route status (true or false)

Query example:

```
GET /billing/api/2.2/fffffffffff/updateRouteState?id=1&state=false
```

```
true
```

deleteRoute(22) — delete the route from routing

Return: *true* / *false*
Arguments:

- **id**(num) — route ID

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteRoute?id=1
```

```
true
```

Directions and tarification

resolveDirection(0) — resolve phone number direction

Return: Selected direction data
Arguments:

- **number**(string) — phone number
- select(ustring{id,code,direction}) — output (direction ID , code or name) *[direction]*

Query example:

```
GET /billing/api/2.2/fffffffffff/resolveDirection?number=79271871234
```

```
Russia Mobile - Megafon
```

getDirections(0) — get the list of all directions

Return: JSON array with list of all system directions Arguments:

- distinct(bool) — show only *code* value for direction *[true]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getDirections
```

```
[{"code": direction code,
"id": direction ID,
"direction": "direction name",
"min_len": minimal number length,
"max_len": maximal number length}, ...]
```

getCurrentRates(0) — get current tarification

Return: JSON array with current user tariffs
Arguments:

- currency(string) — selected currency *[EUR]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getCurrentRates
```

```
{"direction ID":
    {"direction":"direction name",
        "routing":[
            {"routename":"route name",
            "route": route ID,
            "displaycost": Cost in user currency,
            "priority": Priority,
            "cost": Cost in system currency
            }
        ]
    }
    ...
}
```

getRatesList(1) — get the list of tariffs

Return: JSON array with list of accessable tariffs Arguments:

- all(bool) — Show tariffs owned by corporate users *[false]*
- is_public(bool) — Show public tariffs only *[false]*
- is_option(bool) — Show tariff options (*true* — tariff options only, *false* — tariffs only, unset — show all)

Query example:

```
GET /billing/api/2.2/fffffffffff/getRatesList
```

```
[{"bill_type":"tarification type",
"id":tariff ID,
"is_public":public tariff?,
"table_name":"DB table",
"owner":"tariff owner",
"display_switch_price":connection price displayed,
"name":"Tariff name",
"switch_price":connection price in system currency,
"did_multiplier":price multiplier for DID-numbers,
"is_option":Tariff option?,
"multiplier":price multiplier,
"description":"Tariff description"}, ...]
```

switchRates(1) — Change tariff, tariff option connection and disconnection

Return: *true* / *false*
Arguments:

- **id**(num) — tariff (option) ID
- state(bool) — status (only for tariff options)
- sip_extension(string) — SIP account for change

Query example:

```
GET /billing/api/2.2/fffffffffff/switchRates?id=1&sip_extension=1000001
```

```
true
```

getRates(1) — get tariff directions by tariff (option) ID

Return: JSON list of all tariff directions
Arguments:

- **id**(num) — tariff (option) ID

Query example:

```
GET /billing/api/2.2/fffffffffff/getRates?id=1
```

```
{"direction ID":
    {"direction":"direction name",
        "routing":[
            {"routename":"route name",
            "route": route ID,
            "displaycost": cost in user currency,
            "priority": priority,
            "cost": cost in system currency
            }
        ]
    }
    ...
}
```

getRatesInfo(22) — get tariff parameters

Return: tariff parameters (JSON)
Arguments:

- **id**(num) — tariff (option) ID

Query example:

```
GET /billing/api/2.2/fffffffffff/getRatesInfo?id=1
```

```
{"bill_type":"Tarification type",
"id":tariff ID,
"is_public":public tariff?,
"table_name":"DB table",
"owner":"Tariff owner",
"display_switch_price":connection price displayed,
"name":"Tariff name",
"switch_price":Connection price in system currency,
"did_multiplier":price multiplier for DID-numbers,
"is_option":Tariff option?,
"multiplier":price multiplier,
"description":"Tariff description"}
```

addRates(22) — add new tariff

Return: *true* / *false*
Arguments:

- **table**(string) — DB table name
- **name**(string) — tariff (option) name
- bill_type(string) — tarification type *[by*minute]_
- is_option(bool) — tariff option? *[false]*
- is_public(bool) — public tariff? *[false]*
- switch_price(num) — connection price *[0.00]*
- multiplier(num) — price multiplier *[1.00]*
- did_multiplier(num) — direct number multiplier *[1.00]*
- description(string) — tariff description *[]*

Query example:

```
GET /billing/api/2.2/fffffffffff/addRates?table=test&name=test
```

```
true
```

updateRates(22) — update tariff parameters

Return: *true* / *false*
Arguments:

- **id**(num) — tariff (option) ID
- name(string) — tariff (option) name
- bill_type(string) — tarification type
- is_option(bool) — tariff option?
- is_public(bool) — public tariff?
- switch_price(num) — connection price
- multiplier(num) — price multiplier
- did_multiplier(num) — direct number multiplier
- description(string) — tariff description

Query example:

```
GET /billing/api/2.2/fffffffffff/updateRates?id=1&name=etst
```

```
true
```

deleteRates(22) — delete tariff

Return: *true* / *false*
Arguments:

- **id**(num) — tariff (option) ID

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteRates?id=1
```

```
true
```

addDirection(22) — add direction in tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — tariff (option) ID
- **direction**(num) — direction ID
- **cost**(num) — tarification unit cost
- **route**(num) — route ID
- priority(num) — priority *[1]*

Query example:

```
GET /billing/api/2.2/fffffffffff/addDirection?rates=1&direction=600&route=10&cost=0.15
```

```
true
```

updateDirection(22) — update direction parameters in tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — changed tariff (option) ID
- **direction**(num) — changed direction ID
- prior(num) — changed direction priority *[1]*
- cost(num) — tarification unit cost
- route(num) — route ID
- priority(num) — priority

Query example:

```
GET /billing/api/2.2/fffffffffff/updateDirection?rates=1&direction=600&route=11&cost=0.18
```

```
true
```

updateDirections(22) — update directions parameters in tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — changed tariff (option) ID
- **directions**(string) — JSON array with list of changed directions (see format in *updateDirection()* function)

Query example:

```
POST /billing/api/2.2/fffffffffff/updateDirections?rates=1
```

```
true
```

deleteDirection(22) — delete direction from tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — changed tariff (option) ID
- **direction**(num) — changed direction ID
- prior(num) — deleted direction priority *[1]*

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteDirection?rates=1&direction=600
```

```
true
```

updateMarkup(22) — automatic price increment in tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — changed tariff (option) ID
- **markup**(num) — multiplier

Query example:

```
GET /billing/api/2.2/fffffffffff/updateMarkup?rates=1&markup=1.2
```

```
true
```

copyDirections(22) — copy current user tariff in selected tariff

Return: *true* / *false*
Arguments:

- **rates**(num) — ID of tariff where current tariff will be copied to

Query example:

```
GET /billing/api/2.2/fffffffffff/copyDirections?rates=2
```

```
true
```

getBillTypes(22) — get list of allowed tarification types

Return: JSON array - list of accessable tarification types
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/getBillTypes
```

```
[{"type":"Type name",
"free_threshold": non-tariffed threshold,
"step":tarification step}, ...]
```

getBillType(33) — get tarification type data by its name

Return: tarification type data (JSON)
Arguments:

- **bill_type**(string) — tarification type name

Query example:

```
GET /billing/api/2.2/fffffffffff/getBillType?bill_type=by_second
```

```
{"type":"Type name",
"free_threshold": non-tariffed threshold,
"step":tarification step}
```

addBillType(33) — add tarification type

Return: *true* / *false*
Arguments:

- **bill_type**(string) — tarification type name
- **free_threshold**(num) — non-tariffed threshold
- **step**(num) — tarification step (secs)

Query example:

```
GET /billing/api/2.2/fffffffffff/addBillType?bill_type=custom_type&free_threshold=3&step=10
```

```
true
```

updateBillType(33) — update tarification type parameters

Return: *true* / *false*
Arguments:

- **bill_type**(string) — changed tarification type name
- free_threshold(num) — non-tariffed threshold
- step(num) — tarification step (secs)

Query example:

```
GET /billing/api/2.2/fffffffffff/updateBillType?bill_type=custom&free_threshold=0
```

```
true
```

deleteBillType(33) — delete tarification type

Return: *true* / *false*
Arguments:

- **bill_type**(string) — deleted tarification type name

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteBillType?bill_type=custom
```

```
true
```

Call proccessing

resolveCallerid(0) — get current user CallerID **! DEPRECAsteriskD: Use getValue?property=sip_callerid !**

Return: user CallerID
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/resolveCallerid
```

```
79010000001
```

createCallback(1) — Callback initialization

Return: *true* / *false*
Arguments:

- callerid(string) — CallerID (if not specified - use profile CallerID)
- **destination1**(string) — first destination
- **destination2**(string) — second destination
- epitch(num) — voice tone modifier (from 1 to 6)

Query example:

```
GET /billing/api/2.2/fffffffff/createCallback?destination1=79010000001&destination2=79020000002
```

```
true
```

callPrepare(50) — call initiation

Return: Allowed/Declined MaxCallTime/ErrorCode Destination CallerID DirectionName RoutePrefix RouteName PitchRx PitchTx
Arguments:

- callerid(string) — CallerID (if not specified - use profile CallerID)
- **destination**(string) — destination
- prior(num) — priority *[1]*
- from_originate(string) — if the call was initiated not directly, note **true** here or DID-number that call came to
- source(string) — call initiation Asterisk IP-address
- originated_by(string) — call initiator (for example: user, system, slave etc.)

Query example:

```
GET /billing/api/2.2/fffffffff/callPrepare?login=10000&destination=79010000001
```

```
Allowed 99999 79010000001 79020000002 RussiaMobile-Tele2 outline1/ route1-cli -1 -1
```

callFinish(50) — call finishing

Return: Saved DirectionName [CallerID/Destination] via [route] time cost CURRENCY SIPSTATUS
Arguments:

- callerid(string) — CallerID (if not specified - use profile CallerID)
- **destination**(string) — destination
- prior(num) — priority *[1]*
- **length**(num) — call length, secs
- status(string) — call SIP-status
- is_did(string) — if the call was initiated on DID-number, the parameter takes this DID-number value

Query example:

```
GET /billing/api/2.2/fffffffff/callFinish?login=10000&destination=79020000002&length=0&status=0CANCEL
```

```
Saved RussiaMobile-Tele2 [79010000001/79020000002] via [route1-cli] 0 0 EUR 0CANCEL
```

getTextBalance(50) — get user balance in text form

Return: balance for Asterisk voice generation Arguments:

- balance(num) — balance for text transformation (if not specified, use user balance)
- balance(string) — currency (if not specified, use user currency)

Query example:

```
GET /billing/api/2.2/fffffffff/getTextBalance?login=10000
```

```
balance&null&euro&null&cents&tadam3s
```

authorizeCall(50) — user authorization in DISA system

Return: SIP extension of authorized user or error code (*Declined XXX*)
Arguments:

- **callerid**(string) — CallerID, from which DISA handler got the call
- **auth**(string) — authorization string (usually SIP extension + PIN-code)

Query example:

```
GET /billing/api/2.2/fffffffff/authorizeCall?callerid=79010000001&auth=100006666
```

```
10000
```

authorizeRealm(50) — affiliated Asterisk query validation

Return: *true* / *false*
Arguments:

- **source**(string) — affiliated Asterisk IP-address
- **auth**(string) — MD5-hash of the string "${API-key}${Asterisk IP-address}${API-key name}"

Query example:

```
GET /billing/api/2.2/fffffffff/authorizeRealm?params
```

```
true
```

createTTS(50) — text-to-voice transformation

Return: audio stream in **wav** format Arguments:

- **text**(string) — text

Query example:

```
GET /billing/api/2.2/fffffffff/createTTS?text=hello+world
```

```
<binary data>
```

Direct numbers

getDisaDIDList(0) — get the list of DISA numbers

Return: JSON array - list of accessable DISA numbers Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getDisaDIDList
```

```
[{"did":"access number", "comment":"DISA\/Region"}, ... ]
```

updateDIDState(1) — update direct number status

Return: *true* / *false*
Arguments:

- **did**(string) — changed DID-number
- parameter(ustring{auto*renew, enabled}) — changed parameter _[auto*renew]_
- **state**(bool) — status

Query example:

```
GET /billing/api/2.2/fffffffff/updateDIDState?did=78005555550&parameter=enabled&state=false
```

```
true
```

getDIDPool(1) — get direct numbers pool (list of countries, locations or DID-numbers)

Return: direct numbers pool Arguments:

- country(string) — country
- area(string) — region
- search_pattern(string) — search qeury

Query examples:

```
GET /billing/api/2.2/fffffffff/getDIDPool
```

```
[{"country": "Country"}, ...]
```

```
GET /billing/api/2.2/fffffffff/getDIDPool?country=Country
```

```
[{"area": "Region"}, ...]
```

```
GET /billing/api/2.2/fffffffff/getDIDPool?country=Country&area=Region&search_pattern=666
```

```
[{"displaycost": rent price in user currency,
"displayinstallcost": connection price in user currency,
"cost": rent price in system currency,
"installcost": connection price in system currency,
"country":"Country",
"area":"Region",
"voice_support":"Voice support",
"sms_support":"SMS support",
"did":"DID-number",
"pool_id": Pool ID,
"arg1": "First parameter for connection",
"arg2": "Second parameter for connection",
"arg3": "Third parameter for connection"}, ...]
```

orderDID(1) — direct number order

Return: *true* / *false*
Arguments:

- **pool_id**(num) — pool ID
- **did**(string) — direct number
- arg1(string) — first parameter
- arg2(string) — second parameter
- arg3(string) — third parameter

Query example:

```
GET /billing/api/2.2/fffffffff/orderDID?pool_id=1&did=78005555550&arg1=something&arg2=somewhere
```

```
true
```

getDIDList(1) — get the list of direct numbers

Return: JSON array - list of direct numbers Arguments:

- sort_by(ustring{expiration, cost, did) — sort by (rent expiration, cost, number) *[expiration]*
- own(bool) — show owned direct numbers only *[true]*

Query example:

```
GET /billing/api/2.2/fffffffff/getDIDList
```

```
[{"displaycost": rent price in user currency,
"pool": Pool ID,
"renew_notify": expiration notification sent,
"expiration":"Expiration date in format YYYY-MM-DD HH:MM:SS",
"enabled": DID enabled?,
"did": Direct number,
"auto_renew": automatic prolongation on?,
"comment":"Comment",
"owner":"Owner",
"cost":Rent price in system currency}, ...]
```

addDID(22) — Add new number

Return: *true* / *false*
Arguments:

- **did**(string) — direct number
- **owner**(string) — direct number owner
- comment(string) — comment *[]*
- cost(num) — rent cost *[0.00]*
- expiration(string) — expiration date in YYYY-MM-DD HH:MM:SS format

Query example:

```
GET /billing/api/2.2/fffffffff/addDID?did=78005555550&owner=user&cost=100
```

```
true
```

updateDID(22) — update direct number parameters

Return: *true* / *false*
Arguments:

- **did**(string) — changed direct number
- owner(string) — new owner
- comment(string) — comment
- cost(num) — rent cost
- expiration(string) — expiration date in YYYY-MM-DD HH:MM:SS format

Query example:

```
GET /billing/api/2.2/fffffffff/updateDID?did=78005555550&cost=101
```

```
true
```

deleteDID(22) — delete direct number

Return: *true* / *false*
Arguments:

- **did**(string) — deleted direct number

Query example:

```
GET /billing/api/2.2/fffffffff/deleteDID?did=78005555550
```

```
true
```

getDIDInfo(22) — get direct number data

Return: direct number data (JSON)
Arguments:

- **did**(string) — direct number

Query example:

```
GET /billing/api/2.2/fffffffff/getDIDInfo?did=78005555550
```

```
{"displaycost": rent price in user currency,
"expiration":"Expiration date in format YYYY-MM-DD HH:MM:SS",
"owner":"Owner",
"comment":"Comment",
"did": Direct number,
"auto_renew": automatic prolongation on?,
"enabled": DID enabled?,
renew_notify": expiration notification sent,
"cost":Rent price in system currency}
```

renewDID(22) — direct number prolongation

Return: *true* / *false*
Arguments:

- **did**(string) — prolongated DID-number

Query example:

```
GET /billing/api/2.2/fffffffff/renewDID?did=78005555550
```

```
true
```

addDIDPool(33) — add direct numbers pool

Return: *true* / *false*
Arguments:

- **country**(string) — Country
- **area**(string) — Region
- cost(num) — rent cost for month *[0.00]*
- installcost(num) — connection cost *[0.00]*
- provider(string) — Provider *[local]*
- user(string) — User (goes to handler) *[]*
- pass(string) — password (goes to handler) *[]*
- param1(string) — Parameter 1 (goes to handler) *[]*
- param2(string) — Parameter 2 (goes to handler) *[]*
- param3(string) — Parameter 3 (goes to handler) *[]*
- qty(num) — quantity of numbers shown*[100]*

Query example:

```
GET /billing/api/2.2/fffffffff/addDIDPool?country=Russia&area=Moscow&cost=50
```

```
true
```

updateDIDPool(33) — update direct numbers pool parameters

Return: *true* / *false*
Arguments:

- **id**(num) — changed direct numbers pool ID
- country(string) — country
- area(string) — region
- cost(num) — rent cost for month
- installcost(num) — connection cost
- provider(string) — Provider
- user(string) — User (goes to handler)
- pass(string) — password (goes to handler)
- param1(string) — Parameter 1 (goes to handler)
- param2(string) — Parameter 2 (goes to handler)
- param3(string) — Parameter 3 (goes to handler)
- qty(num) — quantity of numbers shown

Query example:

```
GET /billing/api/2.2/fffffffff/updateDIDPool?id=1&cost=100
```

```
true
```

getDIDPools(33) — get list of direct numbers pools

Return: list of direct numbers pools(JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getDIDPools
```

```
[{"id": pool ID,
"country":"Country",
"area":"Region"
"cost": cost per month,
"installcost": installation price,
"displayinstallcost": installation price in user currency,
"displaycost": cost per month in user currency,
"quantity": numbers quantity per page,
"provider":"Provider"
"username":"User name for provider",
"password":"Password for provider",
"param1":"Parameter №1 for provider",
"param2":"Parameter №2 for provider",
"param3":"Parameter №3 for provider"}, ...]
```

getDIDPoolInfo(33) — get direct numbers pool data

Return: direct numbers pool data (JSON)
Arguments:

- **id**(num) — direct numbers pool ID

Query example:

```
GET /billing/api/2.2/fffffffff/getDIDPoolInfo?id=1
```

```
{"id": pool ID,
"country":"Country",
"area":"Region"
"cost": cost per month,
"installcost": installation price,
"displayinstallcost": installation price in user currency,
"displaycost": cost per month in user currency,
"quantity": numbers quantity per page,
"provider":"Provider"
"username":"User name for provider",
"password":"Password for provider",
"param1":"Parameter №1 for provider",
"param2":"Parameter №2 for provider",
"param3":"Parameter №3 for provider"}
```

deleteDIDPool(33) — delete direct numbers pool

Return: *true* / *false*
Arguments:

- **id**(num) — deleted direct numbers pool ID

Query example:

```
GET /billing/api/2.2/fffffffff/deleteDIDPool?id=1
```

```
true
```

getDIDCount(50) — direct number existing check

Return: Quantity of DID-numbers found
Arguments:

- did(string) — initial DID-number

Query example:

```
GET /billing/api/2.2/fffffffffff/getDIDCount?did=78005555550
```

```
1
```

DIDTables Filters

getDIDActions(1) — get the list of allowed actions

Return: list of allowed actions for DIDTables (JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/getDIDActions
```

```
{"voice":
    {"Action":"Default destination value"},
    {"Action":"Default destination value"},
    ...
},
"sms":{
    {"Action (SMS)":"Default destination value"},
    {"Action (SMS)":"Default destination value"},
    ...
}
```

getFilters(1) — Get the list of allowed filters

Return: list of allowed sources/patterns for DIDTables (JSON)
Arguments:

- resolver(string) — get list of allowed patterns by the handler
- sources(bool) — get sources only *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getFilters
```

```
{"Source1":{ "Pattern1", "Pattern2" },...}
```

addDIDRule(1) — add new DIDTables rule

Return: *true* / *false*
Arguments:

- **did**(string) — direct number
- source(string) — filter
- pattern(string) — pattern
- policy(ustring{ACCEPT,REJECT}) — Policy (accept/reject)
- sms_action(string) — SMS processing action
- sms_destination(string) — Destination (SMS)
- action(string) — incoming call processing action
- server(string) — server that the rule is running on
- destination(string) — Destination
- callerid(string) — caller ID number (TRANSIT string will be replaced with the caller's number)
- timeout(num) — Timeout (secs)
- condition(string) — Condition (SUCCESS - successful, UNSUCCESS - failed, BUSY - busy, NOANSWER - no answer, UNAVAIL - unavailable)
- input(string) — Input (if "condition" is selected as "Input matching")
- position(ustring{start,end}) — Insert the rule at the start *[start]* or end *[end]*

Query example:

```
GET /billing/api/2.2/fffffffffff/addDIDRule?did=78005555550&source=number&pattern=*&policy=ACCEPT&destination=10000
```

```
true
```

updateDIDRule(1) — update DIDTables rule

Return: *true* / *false*
Arguments:

- **did**(string) — Direct number
- **rule**(num) — changed rule number
- pos(num) — new position
- source(string) — filtr
- pattern(string) — pattern
- policy(ustring{ACCEPT,REJECT}) — Policy (accept/reject)
- sms_action(string) — SMS processing action
- sms_destination(string) — Destination (SMS)
- action(string) — incoming call processing action
- server(string) — server that the rule is running on
- destination(string) — Destination
- callerid(string) — caller ID number (TRANSIT string will be replaced with the caller's number)
- timeout(num) — Timeout (secs)
- condition(string) — Condition (SUCCESS - successful, UNSUCCESS - failed, BUSY - busy, NOANSWER - no answer, UNAVAIL - unavailable)
- input(string) — Input (if "condition" is selected as "Input matching")

Query example:

```
GET /billing/api/2.2/fffffffffff/updateDIDRule?did=78005555550&rule=1&pos=2
```

```
true
```

deleteDIDRule(1) — delete DIDTables rule

Return: *true* / *false*
Arguments:

- **did**(string) — direct number
- **rule**(num) — deleted rule number
- pattern(string) — pattern (to delete by pattern — note *rule* as *-1*)

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteDIDRule?did=78005555550&rule=-1&pattern=74955555550
```

```
true
```

getDIDRules(1) — get the list of direct number rules Return: list of direct number rules (JSON)
Arguments:

- **did**(string) — direct number
- rule(num) — rule number (if not specified — returns all rules)

Query example:

```
GET /billing/api/2.2/fffffffffff/getDIDRules?did=78005555550
```

```
[{"timeout": Timeout (secs),
"server":"Server the rule is running on",
"did": Direct number,
"policy":"Policy (accept/reject)",
"condition":"Condition","sms_action":"internal","source":"number","action":"playfile","pattern":"*","destination":"beep","rule":1,"callerid":"TRANSIT"},{"timeout":30,"server":"ptr-aa.narayana.im","did":78124099441,"policy":"ACC
```

resolveDIDRule(50) — get processing instruction for DIDTables rule

Return: OK ACCEPT/REJECT Owner(extension) Action(action) Server (sip*server) Destination (destination) Caller ID (callerid) Timeout(timeout) SMS Action (sms*action) SMS Destination (sms_destination) — for rule execution
Return: SKIP ruleNumber — for transfer to next executable rule Return: FIN — if there are no more rules Arguments:

- **did**(string) — direct number
- rule(num) — rule ID *[1]*
- **callerid**(string) — CallerID
- status(string) — SIP-status
- input(string) — Input (SIP-status must be **99INPUT**)

Query example:

```
GET /billing/api/2.2/fffffffff/resolveDIDRule?params
```

```
OK ACCEPT 10000 call rdx.narayana.im 10000 TRANSIT 30 internal 10000
```

Personal plan sets (aliases)

getAliases(1) — get the list of aliases

Return: list of aliases (JSON)
Arguments:

- all(bool) — show all aliases (otherwise - only owned aliases)

Query example:

```
GET /billing/api/2.2/fffffffff/getAliases
```

```
[{"global": Global alias (0/1),
"id": alias ID,
"owner":"alias owner",
"alias": alias name,
"destination": Destination}, ...]
```

getAliase(1) — get alias data

Return: alias data (JSON)
Arguments:

- **id**(num) — alias ID

Query example:

```
GET /billing/api/2.2/fffffffff/getAlias?id=11
```

```
{"global": Global alias (0/1),
"id": alias ID,
"owner":"alias owner",
"alias": alias name,
"destination": Destination}
```

addAlias(1) — add new alias

Return: *true* / *false*
Arguments:

- **alias**(string) — alias
- **destination**(string) — destination
- sip_extension(string) — SIP login that alias is set to
- global(bool) — is alias global? true/false

Query example:

```
GET /billing/api/2.2/fffffffff/addAlias?alias=111&destination=79010000001
```

```
true
```

updateAlias(1) — update alias parameters

Return: *true* / *false*
Arguments:

- **id**(num) — alias ID
- destination(string) — destination
- sip_extension(string) — SIP login that alias is set to

Query example:

```
GET /billing/api/2.2/fffffffff/updateAlias?alias=111&destination=79010000002
```

```
true
```

deleteAlias(1) — delete alias

Return: *true* / *false*
Arguments:

- **id**(num) — alias ID

Query example:

```
GET /billing/api/2.2/fffffffff/deleteAlias?id=11
```

```
true
```

Data security

getBans(22) — get the list of bans *iptables*

Return: list of bans (JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getBans
```

```
[{"reason":Ban reason,
"source":"Ban source (host) or whitelist",
"id": Ban ID,
"target":"Ban target (host)",
"date":"Ban date",
"clientdate":"Ban date in client timezone",
"permanent": is the ban permanent?}, ...]
```

flushBans(22) — reset bans on user SIP-server

Return: *true* / *false*
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/flushBans
```

```
true
```

banIP(33) — ban IP-address

Return: *true* / *false*
Arguments:

- **source**(string) — ban source (host) or whitelist
- **target**(string) — ban target (host)
- reason(string) — ban reason *[]*

Query example:

```
GET /billing/api/2.2/fffffffffff/banIP?source=whitelist&target=127.0.0.1&reason=ya+je+localhost
```

```
true
```

unbanIP(33) — unbanIP-address

Return: *true* / *false*
Arguments:

- **id**(num) — ban ID

Query example:

```
GET /billing/api/2.2/fffffffffff/unbanIP?id=1
```

```
true
```

updateBanPermanent(33) — ipdate ban status

Return: *true* / *false*
Arguments:

- **id**(num) — ban ID
- **state**(bool) — ban status (*true* — permanent, *false* — temporary)

Query example:

```
GET /billing/api/2.2/fffffffffff/updateBanPermanent?id=1&state=true
```

```
true
```

flushAllBans(33) — reset all bans on all Narayana servers

Return: *true* / *false*
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/flushAllBans
```

```
true
```

closeChannels(50) — close all user active channels

Return: *true* / *false*
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/closeChannels?login=10000
```

```
true
```

Prepaid codes (invites). New users registration.

registerRequest(0) — new user registration request / Invite-code generation

Return: Generated invite-code Arguments:

- **signature**(string) — unique client signature (generated by Web-server)

Query example:

```
GET /billing/api/2.2/fffffffffff/registerRequest?signature=test
```

```
0000000000000000
```

registerByCode(0) — new user registration by invite

Return: *true* / *false*
Arguments:

- **code**(string) — invite-code (prepaid card code)
- register(bool) — registration (otherwise — registration simulation) *[false]*
- user(string) — user name
- web_password(string) — user password MD5-hash for entering Web-interface
- language(string) — Web-interface language

Query example:

```
GET /billing/api/2.2/fffffffffff/registerByCode?code=0000000000000000&register=true&user=test&web_password=098f6bcd4621d373cade4e832627b4f6&language=nenglish
```

```
true
```

activatePrepaidCode(1) — invite-code (prepaid card code) activation on user account

Return: *true* / *false*
Arguments:

- **code**(string) — invite-code
- **amount**(num) — amount of money to transfer (if specified as *0* — transfer all accessable balance)

Query example:

```
GET /billing/api/2.2/fffffffffff/activatePrepaidCode?code=0000000000000000&amount=0
```

```
true
```

getPrepaidCodes(1) — get the list of generated invite-codes (prepaid card codes)

Return: list of generated invite-codes(JSON)
Arguments:

- active(bool) — show inactive codes only *[false]*
- all(bool) — show all codes (otherwise - only codes, created by you) *[false]*
- count(bool) — show number of codes only *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getPrepaidCodes
```

```
[{"created":"creation date",
"clientcreated": "creation date in user timezone",
"balance": Balance,
"displaybalance": Balance in user currency,
"code":"Code",
"accesslevel": Access level,
"owner":"Code creator",
"activated_to":"Activated by",
"activated_date":"Activation date"
"client_activated_date":"Activation date in user timezone"}, ...]
```

createPrepaidCode(1) — generate new invite-code (prepaid card code)

Return: generated invite-code Arguments:

- accesslevel(num) — access level *[10]*
- balance(num) — balance *[0.00]*
- target(string) — User (or SIM-card), who can activate this code (without restrictions if not specified) *[]*

Query example:

```
GET /billing/api/2.2/fffffffff/createPrepaidCode?balance=10.00
```

```
true
```

deletePrepaidCode(1) — delete inactive invite-code (prepaid card code)

Return: *true* / *false*
Arguments:

- **code**(string) — invite-code

Query example:

```
GET /billing/api/2.2/fffffffff/deletePrepaidCode?code=0000000000000000
```

```
true
```

SMS

receiveSMS(0) — get new SMS from external source

Return: *true* / *false*
Arguments:

- **msg_handler**(string) — handler ID (usually attached to API-key, so not obligatory)

Query example:

```
GET /billing/api/2.2/fffffffff/receiveSMS?from=18005555550&to=78005555550&text=Hello+world
```

```
true
```

receiveDeliveryReport(0) — get delivery report from external source

Return: *true* / *false*
Arguments:
* **msg_handler**(string) — handler ID (usually attached to API-key, so not obligatory)
Query example:

```
GET /billing/api/2.2/fffffffff/receiveDeliveryReport?id=AAAAAAAAAAAAAAAAA&status=DELIVERED
```

```
true
```

msgCallback(0) — callback initiation by means of SMS from external source

Return: *true* / *false*
Arguments:

- **msisdn**(string) — phone number that initiation request came from
- **text**(string) — request text in format "SIPEXTEN PIN [CALLERID] DESTINATION1 [DESTINATION2]" (in square braces are non-obligatory parameters)
- **to**(string) — phone number that initiation request came to

Query example:

```
GET /billing/api/2.2/fffffffff/msgCallback?msisdn=78005555550&to=18005555550&text=40000+1111+74999999999+79111111111
```

```
true
```

readSMS(1) — mark all SMS as read

Return: *true* / *false*
Arguments:

- **status**(num) — mark SMS with this status only
- **did**(string) — mark sms for this direct number only

Query example:

```
GET /billing/api/2.2/fffffffff/readSMS?status=0&did=18005555550
```

```
true
```

sendSMS(1) — Send SMS

Return: unique SMS ID Arguments:

- callerid(string) — Caller ID (if not specified, it's taken from user profile)
- **to**(string) — destination number
- **text**(string) — SMS text (urlencoded)

Query example:

```
GET /billing/api/2.2/fffffffff/sendSMS?to=18005555550&text=Hello+World!
```

```
uniqueid
```

sendHLR(1) — Send HLR-request

Return: OK,IMSI,MCC,MNC,LAC HLR-request result
Arguments: —

- **to**(string) — number for HLR-request

Query example:

```
GET /billing/api/2.2/fffffffff/sendHLR?to=18005555550
```

```
OK,999881111111111,999,888,000000
```

sendVoiceMessage(1) — send voice message

Return: unique message ID
Arguments:

- callerid(string) — Caller ID (if not specified, it's taken from user profile)
- **to**(string) — destination phone number
- **text**(string) — message text (urlencoded) or URL to audio file

Query example:

```
GET /billing/api/2.2/fffffffff/sendVoiceMessage?callerid=18005555550&to=780055555550&text=hello,+world
```

```
uniqueid
```

sendSIPMessage(1) — send SIP-message to service client

Return: *true* / *false*
Arguments:

- callerid(string) — caller ID (if not specified, it's taken from user profile)
- **to**(string) — destination phone number
- **text**(string) — message text (urlencoded)

Query example:

```
GET /billing/api/2.2/fffffffff/sendVoiceMessage?to=40000&text=hello+world
```

```
true
```

deliveryReport(1) — get report of unique ID message delivery

Return: SMS Sent (Sent) / SMS Deliv (Delivered) / SMS Undeliv (Not delivered)
Arguments:

- **string**(num) — record ID in call log

Query example:

```
GET /billing/api/2.2/ffffffffff/deliveryReport?id=uniqueid
```

```
SMS Deliv
```

setDeliveryReport(95) — set SMS delivery report

Return: *true* / *false*
Arguments:

- **id**(num) — SMS ID
- **status**(num) — Status

Query example:

```
GET /billing/api/2.2/ffffffffff/setDeliveryReport?id=1&status=0
```

```
true
```

getSMS(95) — get the list of all SMS**! DEPRECAsteriskD: Use readSMS() !**

Return: list of SMS (JSON)
Arguments:

- **status**(num) — status of SMS Query example:

```
GET /billing/api/2.2/ffffffffff/getSMS?status=0
```

```
[{"id": SMS ID (for setDeliveryReport),
"date": "message receiving daate (server time)",
"src": "source",
"dest": "destination",
"msg": "message",
"status": status (0 – unread, 1 – read),
"type": "type (in – incoming, out – outcoming)"}, ...]}
```

User control

getUsers(22) — show the list of users

Return: list of users(JSON)
Arguments:

- all(bool) — show corporate users only *[false]*

Query example:

```
GET /billing/api/2.2/ffffffffff/getUsers
```

```
[{"parameter": "value"}, ...]
See the list of parameters in  _getInfo() function
```

registerUser(22) — register new user

Return: *true* / *false*
Arguments:

- **user**(string) — user nmae
- **web_password**(string) — user password md5-hash
- **accesslevel**(num) — access level
- **sip_server**(string) — SIP-server
- **sip_extension**(string) — SIP-login
- **sip_password**(string) — SIP-password
- balance(num) — initial balance in registering user currency *[0.00]*
- language(string) — user interface language *[config.default*language]_
- currency(string) — user currency *[EUR]*
- sip_srtp(bool) — use SRTP enctyption *[false]*
- account(string) — account number (or auto to create automatically) *[auto]*

! NB: even if the currency of registered user is specified, as initial balance the currency of registering user will be used.

Query example:

```
GET /billing/api/2.2/ffffffffff/registerUser?user=newuser&web_password=5f4dcc3b5aa765d61d8327deb882cf99&accesslevel=11&sip_server=sip.server.com&sip_extension=40000&sip_password=verysecretpassword&balance=10.00
```

```
true
```

updateUser(1) — update user parameters

Return: *true* / *false*
Arguments:

- user(string) — user login
- web_password(string) — new authorization password (md5-hash)
- accesslevel(num) — access level
- active(num) — status (1 — active, 0 — inactive, -1 — locked)
- account(num) — account number
- currency(string) — user currency
- language(string) — user interface language
- timezone(num) — UTC timezone
- rates(num) — tariff ID
- allowed_rates(string) — allowed tariffs
- allowed_call_len(num) — maximal call length
- notify_email(string) — E-Mail for notifications
- notify_balance(num) — get balance minimum reaching notification
- notify_did(num) — number of days prior to DID-number for notification disconnection
- notify_ticket(num) — get notifications on ticket responses
- notify_balance_limit(num) — minimal balance threshold for notifications
- invites(num) — number of invitations
- comment(string) — comment

Query example:

```
GET /billing/api/2.2/ffffffffff/updateUser?user=newuser&active=-1
```

```
true
```

deleteUser(22) — delete user

Return: *true* / *false*
Arguments:

- **user**(string) — user login

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteUser?user=newuser
```

```
true
```

SIP accounts

getExtensions(1) — get the list of user SIP-accounts

Return: list of user SIP-accounts (JSON)
Arguments:

- own(bool) — get owned SIP-accounts only *[true]*
- all(bool) — get all SIP-accounts *[false]*
- owner(string) — get SIP-accounts of specified user

Query example:

```
GET /billing/api/2.2/fffffffffff/getExtensions?own=true
```

```
[{"sip_transport":"SIP transport (udp,tcp,tls/udp/tcp/tls/udp,tcp)",
"disa_pin": PIN-code,
"sip_callerid": "Caller ID",
"owner":"SIP-account owner",
"sip_password":"SIP-password",
"sip_host":"Host for IP authorization",
"sip_srtp": SRTP enctyption (0/1),
"sip_language":"SIP-account language (internal command language)",
"sip_reach": user call availability (2 — available, 0 — unavailable),
"sip_redirect": Redirect incoming calls here,
"sip_extension": SIP-login,
"sip_realm":"SIP-server short name",
"disa_trusted_cli": "Trusted CallerID",
"rates_name":"Tariff name",
"sip_max_call_len": maximal call length,
"sip_address":"SIP-server address" (new in v2.3),
"sip_rates_id": tariff ID,
"sip_rates_options": tariff options (separated by commas),
"sip_server":"SIP-server IP-address",
"lastupdated":"last status update" (for realtime),
"sip_force_callback": Forced Callback (0/1),
"sip_pitch_shift": voice distortion (-1 — off)}, ...]
```

getExtensionInfo(1) — get SIP-account data

Return: SIP-account data (JSON) or the requested parameter value
Arguments:

- **sip_extension**(string) — SIP-account
- property(string) — requested parameter for getting the value from

Query example:

```
GET /billing/api/2.2/fffffffffff/getExtensionInfo?sip_extension=40000
```

```
{"sip_transport":"SIP transport (udp,tcp,tls/udp/tcp/tls/udp,tcp)",
"disa_pin": PIN-code,
"sip_callerid": "Caller ID",
"owner":"SIP-account owner",
"sip_password":"SIP-password",
"sip_host":"Host for IP authorization",
"sip_srtp": SRTP enctyption (0/1),
"sip_language":"SIP-account language (internal command language)",
"sip_reach": user call availability (2 — available, 0 — unavailable),
"sip_redirect": Redirect incoming calls here,
"sip_extension": SIP-login,
"sip_realm":"SIP-server short name",
"disa_trusted_cli": "Trusted CallerID",
"rates_name":"Tariff name",
"sip_max_call_len": maximal call length,
"sip_address":"SIP-server address" (new in v2.3),
"sip_rates_id": tariff ID,
"sip_rates_options": tariff options (separated by commas),
"sip_server":"SIP-server IP-address",
"lastupdated":"last status update" (for realtime),
"sip_force_callback": Forced Callback (0/1),
"sip_pitch_shift": voice distortion (-1 — off)}
```

getAvailExtension(1) — get the nearest SIP-account available for registration

Return: SIP-account available for registration
Arguments:

- own(bool) — return next available additional (user) SIP-account for registration (XXXXXyy) *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getAvailExtension?own=true
```

```
4000001
```

createExtension(1) — create SIP-account

Return: *true* / *false*
Arguments:

- owner(string) — SIP-account owner
- **sip_extension**(string) — SIP-account
- **sip_password**(string) — SIP password
- sip_server(string) — SIP server address
- sip_callerid(string) — Default Caller ID
- sip_language(string) — Default SIP-account language
- sip_srtp(bool) — Use SRTP enctyption *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/createExtension?sip_extension=4000001&sip_password=password
```

```
true
```

updateExtension(1) — update SIP-account

Return: *true* / *false*
Arguments:

- sip_extension(string) — SIP-account for updating
- extension(string) — new SIP-account number (name)
- sip_password(string) — new SIP password
- sip_server(string) — new SIP server
- sip_srtp(num) — Use SRTP enctyption (0/1)
- sip_transport(ustring) — allowed SIP transport (udp,tcp,tls/udp/tcp/tls/udp,tcp)
- sip_callerid(string) — caller ID
- sip_max_call_len(num) — maximal call length
- sip_reach(num) — user availability for call (2 — available, 0 — unavailable)
- sip_redirect(string) — where to redirect incoming calls for this SIP-login
- sip_force_callback(num) — forced Callback to trusted number (**disa*trusted*cli**) (0/1)
- sip_language(string) — internal phone command language
- sip_rates_id(num) — tariff ID

- disa_pin(num) — PIN-code
- disa_trusted_cli(string) — Trusted number

Query example:

```
GET /billing/api/2.2/fffffffff/updateExtension?sip_extension=40000&sip_srtp=1&sip_reach=2
```

```
true
```

deleteExtension(1) — delete SIP-account

Return: *true* / *false*
Arguments:

- **sip_extension**(string) — SIP-account for deleting

Query example:

```
GET /billing/api/2.2/fffffffff/deleteExtension?sip_extension=40000
```

```
true
```

Event log

getLogs(1) — get logs

Return: logs (JSON)
Arguments:

- limit(num) — get last N records[100]_
- only_completed(bool) — whow successful calls only *[false]*
- resolve_info(bool) — show additional information about the user *[false]*
- user(string) — user to get event log from

Query example:

```
GET /billing/api/2.2/fffffffff/getLogs?resolve_info=true
```

```
[{"direction":"Direction",
"route":"Route",
"cost": Price,
"date":"Record date",
"displaycost": Price in user currency,
"id": record ID,
"username":"User name",
"displaycurrency":"User currency",
"status":"Status",
"clientdate":"Date in user timezone",
"destination":"Destination",
"time":"Call length",
"callerid":"Caller ID",
"sip_extension": "SIP-login",
"info": "Called user additional information",
"debug": "Call debuginformation"}, ...]
```

deleteLogs(1) — delete event log

Return: *true* / *false*
Arguments:

- user(string) — user name for delete the log from (if not specified — delete own log)

Query example:

```
GET /billing/api/2.2/fffffffff/deleteLogs
```

```
true
```

Interface functions

getCourse(0) — get internal currency exchange rate and other data

Return: Exchange rate (JSON)
Arguments:

- **currency**(string) — currency to get data

Query example:

```
GET /billing/api/2.2/fffffffff/getCourse?currency=EUR
```

```
{"eur_course": Euro rate,"currency":"Currency code","displayname":"€","transfer_fee":Transer between accounts cost,"referral_reward":Reward for invitation}
```

getCurrencies(0) — get the list of currencies

Return: list of currencies (JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getCurrencies
```

```
[{"eur_course":1,"currency":"EUR","displayname":"€"},{"eur_course":68.917,"currency":"RUB","displayname":"rub. "}]
```

News and broadcast messages

getNews(0) — get news

Return: list of all news or concrete news (JSON)
Arguments:

- id(string) — news ID to get
- all(bool) — show all news
- broadcast(bool) — show broadcast message news only

Query example:

```
GET /billing/api/2.2/fffffffff/getNews
```

```
[{"is_broadcast": broadcasting? (0/1),
"id":"news ID",
"author":"news author",
"message":"Message text",
"access": minimal access level required,
"language":"language of viewing",
"lastupdated":"last update (server timezone)",
"clientlastupdated":"last update (user timezone)",",
"broadcast":"Broadcasting message",
"date":"date (server timezone)",
"clientdate":"date (client timezone)",
"header":"Message header",
"lastupdated_by":"Last updated by",
"typ":"type (for representation only; warning/error/success/info)"}, ... ]
```

addNews(33) — add news

Return: *true* / *false*
Arguments:

- **id**(string) — news ID

- is*broadcast(bool) — Broadcasting? _[false]*
- user(string) — who can read this news (if not specified - can be read by anyone)
- access(num) — minimal access level (if 0 — even unauthorized users) *[0]*
- language(string) — language of users that can read the news (if not specified — can be read by user of any language)
- typ(ustring{info,warning,error,success}) — news type (used for web-interface representation) *[info]*
- broadcast(string) — broadcast message
- header(string) — news header
- message(string) — news text

Query example:

```
GET /billing/api/2.2/fffffffff/addNews?id=testnews&header=test&message=hello+world
```

```
true
```

updateNews(33) — update news

Return: *true* / *false*
Arguments:

- **id**(string) — news ID
- is*broadcast(bool) — Broadcasting? _[false]*
- user(string) — who can read this news (if not specified - can be read by anyone)
- access(num) — minimal access level (if 0 — even unauthorized users) *[0]*
- language(string) — language of users that can read the news (if not specified — can be read by user of any language)
- typ(ustring{info,warning,error,success}) — news type (used for web-interface representation) *[info]*
- broadcast(string) — broadcast message
- header(string) — news header
- message(string) — news text

Query example:

```
GET /billing/api/2.2/fffffffff/updateNews?id=testnews&message=hello+earth
```

```
true
```

deleteNews(33) — Delete news

Return: *true* / *false*
Arguments:

- **id**(string) — news ID

Query example:

```
GET /billing/api/2.2/fffffffff/deleteNews?id=testnews
```

```
true
```

Ticket system

getUnreadTickets(1) — get number of unread tickets **! DEPRECAsteriskD: function is no longer used !**

Return: number of unread tickets
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getUnreadTickets
```

```
1
```

createTicket(1) — create ticket or repond to existing one

Return: Number of created / existing ticket
Arguments:

- **destination**(string) — receiver login
- **subject**(string) — Subject
- **message**(string) — Message
- comment(string) — Comment (for administrators only)
- ticket(num) — ticket number for responding

Query example:

```
GET /billing/api/2.2/fffffffff/createTicket?destination=support&subject=Help&message=Phone+is+not+ringing!
```

```
5162
```

getTickets(1) — get the list of tickets

Return: list of tickets (JSON)
Arguments:

- limit(num) — get N last tickets *[800]*
- all(bool) — get all tickets *[false]*

Query example:

```
GET /billing/api/2.2/fffffffff/getTickets
```

```
[{"status": ticket status (1 – unread, 0 – read, -1 – closed),
"unread": unread (true/false),
"subject":"subject",
"sender":"source",
"recipient":"Destination",
"ticket": ticket number,
"msgid": message number,
"date":"date in server timezone",
"clientdate":"date in user timezone"}, ... ]
```

getTicket(1) — get ticket

Return: ticket messages (JSON)
Arguments:

- **ticket**(num) — ticket number

Query example:

```
GET /billing/api/2.2/fffffffff/getTicket?params
```

```
[{"type":"message type (in/out)",
"ticket": ticket number,
"reply":"reply to",
"message":"message",
"status":status (0 – read, 1 – unread, -1 – closed),
"subject":"subject",
"sender":"source",
"username":"source user name (when sender == support)",
"date":"date (server timezone)",
"clientdate":"date (user timezone)",
"recipient":"destination",
"msgid": message ID}, ... ]
```

updateTicket(1) — update ticket status

Return: *true* / *false*
Arguments:

- **ticket**(num) — ticket number
- action(ustring{close,open,mark*unread,assign*to) — action *[open]*

- assign_to(string) — list of users to whom assign the ticket

Query example:

```
GET /billing/api/2.2/fffffffff/updateTicket?ticket=5162&action=mark_unread
```

```
true
```

deleteTicket(33) — delete ticket

Return: *true* / *false*
Arguments:

- **ticket**(num) — ticket number
- username(string) — user (to delete all his tickets)

Query example:

```
GET /billing/api/2.2/fffffffff/deleteTicket?ticket=5162
```

```
true
```

deleteMessage(33) — delete message

Return: *true* / *false*
Arguments:

- **id**(num) — message ID

Query example:

```
GET /billing/api/2.2/fffffffff/deleteMessage?id=10000
```

```
true
```

Finances and statistics

getPaymentMethods(0) — get the list of allowed payment methods

Return: list of allowed payment methods (JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffff/getPaymentMethods
```

```
{"private": {"Verified users payment method":"Method name"},
"public":{"All users payment method":"Method name"},
"default": "Default payment method"}
```

finishInvoice(0) — handler request on payment finishing

Return: *depends on handler*
Arguments: *depends on handler*

Query example:

```
POST /billing/api/2.2/fffffffff/finishInvoice
```

```
ok
```

getStats(1) — Get statistics

Return: statistics (JSON)
Arguments: —

- financials(bool) — get financial data *[false]*
- statistic(bool) — get statistics *[false]*
- date1(string) — date (from)
- date2(string) — date (to)

Query example:

```
GET /billing/api/2.2/fffffffff/getStats
```

```
{"forecast": {
  "forecast": for how many days there's enough money for user,
  "recommmend": recommended payment in system currency(EUR),
  "display_recommend": recommended payment in user currency},
"financials": {
  "gsm_cost": roaming cost in system currency (EUR),
  "display_gsm_cost": roaming cost in user currency,
  "call_cost": call cost in system currency (EUR),
  "display_call_cost": call cost in user currency,
  "sms_cost": SMS cost in system currency(EUR),
  "display_sms_cost": SMS cost in user currency,
  "gprs_cost": mobile Internet cost in system currency (EUR),
  "display_gprs_cost": mobile Internet cost in user currency,
  "other_cost": other cost (incl. transactions) in system currency (EUR)
  "display_other_cost": other cost (incl. transactions) in user currency ,
  "total_cost": total costs in system currency (EUR),
  "display_total_cost": total costs in user currency},
"stat":{
  "total_calls": total calls,
  "success_calls": succesful calls,
  "success_calls_percent": succesful calls percentage,
  "calls_duration": total call length,

  "total_sms": total SMS,
  "success_sms": succesful SMS,
  "success_sms_percent": succesful SMS percentage,

  "total_gsm_calls": total GSM-calls
  "success_gsm_calls": succesful GSM-calls,
  "success_gsm_calls_percent": succesful GSM-calls percentage,
  "gsm_calls_duration": total GSM-calls length,

  "gprs_traffic": traffic received/sent (mob. Internet), Kbytes}
}
```

transferMoney(1) — transfer money to another user account

Return: *true* / *false*
Arguments:

- **destination**(string) — Destination (login or account number)
- **amount**(num) — sum in sender currency
- comment(string) — comment

Query example:

```
GET /billing/api/2.2/fffffffff/transferMoney?destination=100000000&amount=100&comment=Enjoy
```

```
true
```

createInvoice(1) — create payment invoice

Return: payment invoice data (JSON)
Arguments: —

- **payment_method**(string) — payment handler
- **payment_type**(string) — payment method
- **payment_amount**(num) — sum in requesting user currency

Query example:

```
GET /billing/api/2.2/fffffffff/createInvoice?payment_method=pscb&payment_type=qiwi
```

```
{"comission": comission (percentage),
"transaction": transaction (invoice) number,
"msg": "arbitrary message about payment features",
"method": "Query method in URL (GET/POST)",
"url": "URL to invoice payment",
-- Note: the following parameters should be transferred into specified URL by method specified in "method"! --
"parameter1": "value1",
"parameter2": "value2"
```

generateInvoice(1) — generate PDF-invoice by specified data **! DEPRECAsteriskD: Use viewInvoice(invoice) !**

Return: PDF-invoice
Arguments:

- **invoice**(num) — Invoice number
- **date**(string) — Date of billing
- **deadline**(string) — Deadline date
- **services**(string) — Services in format Service@Units tariffed@Unit name@Unit cost,Service@Units tariffed@Unit name@Unit cost,..

Query example:

```
GET /billing/api/2.2/fffffffff/generateInvoice?invoice=1&date=2017-06-24 00:00:00&deadline=2017-06-25 00:00:00&services=Service@1@Unit@100
```

```
%PDF-1.4\n1 0 obj\n<< BINARY PDF FILE
```

monthlyInvoice(1) — generate monthly invoice

Return: PDF-invoice Arguments:

- month(num) — month number for invoice generation (if not specified, the previous one is used)
- email(bool) — Send invoice on e-mail *[false]*

Query example:

```
GET /billing/api/2.2/fffffffff/monthlyInvoice
```

```
%PDF-1.4\n1 0 obj\n<< BINARY PDF FILE
```

getInvoices(1) — get the list of invoices

Return: list of invoices(JSON)
Arguments:

- own(bool) — show owned invoices only *[true]*
- bank(bool) — show bank transaction only *[false]*
- done(bool) — show paid invoices

Query example:

```
GET /billing/api/2.2/fffffffff/getInvoices?params
```

```
[{"id": invoice number,
"owner":"invoice owner",
"comission": comission,
"vat": VAT,
"amount": sum in system currency (EUR),
"displayamount": sum in user currency,
"status": status,
"msg":"arbitrary message about payment features",
"method":"Payment method",
"handler": "handler(system information)",
"billing_data":"User full name and address",
"txdata": "Transaction system information (see _createInvoice()_ function)",
"date":"invoce date (server timezone)",
"clientdate":"invoce date (user timezone)",
"validthru":"invoice valid through (server timezone)",
"clientvalidthru": "invoice valid through (user timezone)}, ... ]
```

getInvoiceInfo(1) — get invoice data

Return: invoice data (JSON)
Arguments:

- **invoice**(num) — invoice number

Query example:

```
GET /billing/api/2.2/fffffffff/getInvoiceInfo?id=1
```

```
{"id": invoice number,
"owner":"invoice owner",
"comission": comission,
"vat": VAT,
"amount": sum in system currency (EUR),
"displayamount": sum in user currency,
"status": status,
"msg":"arbitrary message about payment features",
"method":"Payment method",
"handler": "handler(system information)",
"billing_data":"User full name and address",
"txdata": "Transaction system information (see _createInvoice()_ function)",
"date":"invoce date (server timezone)",
"clientdate":"invoce date (user timezone)",
"validthru":"invoice valid through (server timezone)",
"clientvalidthru": "invoice valid through (user timezone)}
```

viewInvoice(1) — download PDF-invoice

Return: binary PDF-file
Arguments:

- **invoice**(num) — invoice number

Query example:

```
GET /billing/api/2.2/fffffffff/viewInvoice?params
```

```
%PDF-1.4\n1 0 obj\n<< BINARY PDF FILE
```

deleteInvoice(33) — delete invoice

Return: *true* / *false*
Arguments:

- **invoice**(num) — invoice number

Query example:

```
GET /billing/api/2.2/fffffffff/deleteInvoice?invoice=1
```

```
true
```

Accounts

updateAccount(1) — update account parameters

Return: *true* / *false*
Arguments:

- account(num) — account number
- balance(num) — balance in **requesting** user currency
- overdraft(num) — credit limit in **requesting** user currency
- refill_allowed(num) — financial transaction allowed (0 — forbidden, 1 — allowed, 2 — verified)
- billing_name(string) — full name for invoicing
- billing_address(string) — address for invoicing
- billing_email(string) — email for invoicing
- billing_bank_account(string) — bank account number for invoicing
- postpaid(num) — postpaid account (send invoice in the end of month) (0/1)

! NB: users with access level over 33 (administrators) cannot change users balance! Use transferMoney() function

Query example:

```
GET /billing/api/2.2/fffffffffff/updateAccount?account=100000000&postpaid=1
```

```
true
```

getAccounts(22) — get the list of users accounts

Return: list and data of users accounts (JSON)
Arguments:

- all(bool) — show accounts of resellers *[false]*

Query example:

```
GET /billing/api/2.2/fffffffffff/getAccounts
```

```
[ {"account": account number,
"users":["list", "of account", "users"],
"balance": balance in system currency (EUR),
"displaybalance": balance in requesting user currency,
"overdraft": credit limit in system currency,
"displayoverdraft": credit limit in user currency,
"displaycurrency":"€ ($/rub./etc.)",
"refill_allowed": financial transactions allowed (0 — forbidden, 1 — allowed, 2 — verified),
"postpaid": postpaid account(invoice in the end of month) }, ... ]
```

getAccountInfo(22) — get account data

Return: account data(JSON)
Arguments:

- **account**(num) — account number

Query example:

```
GET /billing/api/2.2/fffffffffff/getAccountInfo?account=100000000
```

```
{"account": account number,
"users":["list", "of account", "users"],
"balance": balance in system currency (EUR),
"displaybalance": balance in requesting user currency,
"overdraft": credit limit in system currency,
"displayoverdraft": credit limit in user currency,
"displaycurrency":"€ ($/rub./etc.)",
"refill_allowed": financial transactions allowed (0 — forbidden, 1 — allowed, 2 — verified),
"postpaid": postpaid account(invoice in the end of month) }
```

deleteAccount(22) — delete account

Return: *true* / *false*
Arguments:

- **account**(num) — account number

Query example:

```
GET /billing/api/2.2/fffffffffff/deleteAccount?account=100000000
```

```
true
```

API access

getKey(33) — get API-key data

Return: API-key data (JSON)
Arguments:

- **name**(string) — API-key ID

Query example:

```
GET /billing/api/2.2/fffffffffff/getKey?name=TESTKEY1
```

```
{"maintenance":"last maintenance date (for system purposes)",
"auth_by":"Authorization by",
"dnsname":"DNS-name (for SIP-servers)",
"source":"IP-address or mask (*.*.*.*)",
"name":"Key ID",
"secure_only": HTTPS only (0/1),
"enabled": Enabled (0/1),
"accesslevel": Access level,
"key":"API-key"}
```

getKeys(33) — get the list of API-keys

Return: list of API-keys (JSON)
Arguments: —
Query example:

```
GET /billing/api/2.2/fffffffffff/getKeys
```

```
[{"maintenance":"last maintenance date (for system purposes)",
"auth_by":"Authorization by",
"dnsname":"DNS-name (for SIP-servers)",
"source":"IP-address or mask (*.*.*.*)",
"name":"Key ID",
"secure_only": HTTPS only (0/1),
"enabled": Enabled (0/1),
"accesslevel": Access level,
"key":"API-key"}, ...]
```

addKey(33) — add new API-key

Return: *true* / *false*
Arguments:

- **name**(string) — API-client ID
- **auth_by**(string) — Authorization by (id,username,sip_extension,msisdn,iccid...)
- **source**(string) — IP-address (or mask)
- **apikey**(string) — API-key
- **accesslevel**(num) — access level (0 for client keys)
- default_user(string) — default attached user
- allowed_methods(string) — methods allowed for execution
- secure_only(num) — request only by *https://*

Query example:

```
GET /billing/api/2.2/fffffffffff/addKey?params
```

```
answer example
```

updateKey(33) — update API-key data

Return: *true* / *false*
Arguments:

- name(string) — API-client ID
- auth_by(string) — Authorization by (id,username,sip_extension,msisdn,iccid...)
- source(string) — IP-address (or mask)
- apikey(string) — API-key
- accesslevel**(num) — access level (0 for client keys)
- default_user(string) — default attached user
- allowed_methods(string) — methods allowed for execution
- secure_only(num) — request only by *https://*
- enabled(num) — Status (1 — active, 0 — inactive)

Query example:

```
GET /billing/api/2.2/fffffffff/updateKey?name=TESTKEY1&secure_only=1
```

---

revokeKey(33) — revoke API-key

Return: *true* / *false*
Arguments:

- **name**(string) — API-client ID

Query example:

```
GET /billing/api/2.2/fffffffff/revokeKey?name=TESTKEY1
```

```
true
```

GSM

gsmRequest(0) — request to GSM-provider

Return: _depends on provider
Arguments:

- **provider**(string) — provider name (located in **inc/gsm/**)
- action(string) — subprogram for call (otherwise Default)

Query example:

```
POST /billing/api/2.2/fffffffff/gsmRequest
```

```
ok
```

SIM-cards

getSIMList(1) — get the list of SIM-cards

Return: list of SIM-cards (JSON)
Arguments:

- all(bool) — show all accessable SIM-cards (including inactive) *[true]*
- own(bool) — show SIM-cards owned by current user only *[false]*

Query example:

```
GET /billing/api/2.2/fffffffff/getSIMList
```

```
[{"allow_direct_sms": Allow direct SMS bypassing Narayana network,
"msisdn": MSISDN (user number),
"owner":"SIM-card owner",
"lastnetworkdate":"Last network registration date",
"iccid":"SIM-card ICCID",
"puk1":PUK1-code,
"puk2":PUK2-code,
"callback_cid":CallerID for Callback,
"sip_extension":SIP-profile,
"provider":"Provider",
"enabled": SIM-card enabled (1/0),
"lastnetwork":network MCCMNC}, ...]
```

getSIMInfo(1) — get SIM-card data

Return: SIM-card data (JSON)
Arguments:

- **iccid**(string) — SIM-card ICCID

Query example:

```
GET /billing/api/2.2/fffffffff/getSIMInfo?iccid=89372
```

```
{"allow_direct_sms": Allow direct SMS bypassing Narayana network,
"msisdn": MSISDN (user number),
"owner":"SIM-card owner",
"lastnetworkdate":"Last network registration date",
"iccid":"SIM-card ICCID",
"puk1":PUK1-code,
"puk2":PUK2-code,
"callback_cid":CallerID for Callback,
"sip_extension":SIP-profile,
"provider":"Provider",
"enabled": SIM-card enabled (1/0),
"lastnetwork":network MCCMNC}
```

bindSIM(1) — SIM-card binding initiation

Return: *true* / *false*
Arguments:

- **iccid**(string) — SIM-card ICCID
- **puk1**(string) — PUK1-code
- **puk2**(string) — PUK2-code

Query example:

```
GET /billing/api/2.2/fffffffff/bindSIM?iccid=89372&puk1=1111&puk2=2222
```

```
true
```

unbindSIM(1) — SIM-card unbinding initiation

Return: *true* / *false*
Arguments:

- **iccid**(string) — SIM-card ICCID

Query example:

```
GET /billing/api/2.2/fffffffff/unbindSIM?iccid=89372
```

```
true
```

updateSIM(1) — update SIM-card parameters

Return: *true* / *false*
Arguments:

- **iccid**(string) — SIM-card ICCID
- owner(string) — SIM-card owner
- puk1(string) — PUK1-code
- puk2(string) — PUK2-code
- sip_extension(string) — used SIP-profile
- callback_cid(string) — CallerID for Callback
- directsms(num) — allow direct SMS bypassing Narayana network (0 — no, 1 — yes)
- enabled(num) — SIM-card activation (0 — off, 1 — on)

Query example:

```
GET /billing/api/2.2/fffffffff/updateSIM?iccid=89372&sip_extension=12345
```

```
true
```

addSIM(33) — add SIM-card

Return: *true* / *false*
Arguments:

- **provider**(string) — Provider
- **msisdn**(string) — MSISDN (client number)
- **iccid**(string) — ICCID/IMSI (SIM-card unique number)
- puk1(string) — PUK1-code (used for activation)
- puk2(string) — PUK2-code (used for activation)

Query example:

```
GET /billing/api/2.2/fffffffff/addSIM?provider=test&msisdn=111&iccid=89372&puk1=0000&puk2=0000
```

```
true
```

deleteSIM(33) — delete SIM-card

Return: *true* / *false*
Arguments:

- **iccid**(string) — SIM-card ICCID

Query example:

```
GET /billing/api/2.2/fffffffff/deleteSIM?iccid=89372
```

```
true
```

notifySIM(50) — send SMS-notification to SIM-card

Return: *true* / *false*
Arguments:

- **sim**(string) — SIM-card ICCID
- **from**(string) — sender number
- **text**(string) — message text

Query example:

```
GET /billing/api/2.2/fffffffff/notifySIM?sim=89372&from=111&text=hello+world
```

```
true
```

Technical maintenance

asteriskRequest(22) — Asterisk query

Return: Asterisk answer
Arguments:

- **asterisk**(string) — Asterisk IP-address
- **destination**(string) — Query destination (for example, **interface**)
- **process**(string) — First argument (generally, it is method name)
- params(string) — Query paramseters

Пример запроса:

```
GET /billing/api/2.2/fffffffff/asteriskRequest?asterisk=127.0.0.1&destination=sipreload&process=sipreload
```

```
ok
```

updateDialplan(33) — update dialplan version

Return: OK
Arguments:

- **asterisk**(string) — Asterisk IP-address
- **url**(string) — dialplan file URL where **/** is replaced with **~**

Query example:

```
GET /billing/api/2.2/fffffffff/updateDialplan?asterisk=127.0.0.1&url=localhost~dialplan
```

```
OK 0082R20170626_205
```

restoreDialplan(33) — restore dialplan version

Return: OK
Arguments:

- **asterisk**(string) — Asterisk IP-address

Query example:

```
GET /billing/api/2.2/fffffffff/restoreDialplan
```

```
OK 0082R20170626_205
```

updateInterface(33) — update *interface* file

Return: *true* / *false*
Arguments:

- **asterisk**(string) — Asterisk IP-address
- **url**(string) — interface file URL where **/** is replaced with **~**

Query example:

```
GET /billing/api/2.2/fffffffff/updateInterface?asterisk=127.0.0.1&url=localhost~interface
```

```
OK 0.12.4
```

doMaintenance(90) — carry out regular procedures in the database(exchange rates, invoices, DID-numbers etc.)

Return: *true* / *false*
Arguments: Query example:

```
GET /billing/api/2.2/ffffffffff/doMaintenance
```

```
true
```