Description of the protocol of interaction with the service

Interaction with the service is performed using the REST protocol.

Request format

Requests are made via the HTTP 1.1 protocol using SSL (HTTPS), to the address:

https://3dsec.sberbank.ru/sbersafe/<method_name>

See also:Service connection coordinates.

HTTP requests must contain headers. The composition is shown in the table below.

Header element and its value	Description
`POST /api/bindings/ HTTP/1.1`	The type of interaction with the server, the request used, and the version of the HTTP protocol.
`Host: 3dsec.sberbank.ru/sbersafe`	The server which the request is sent to.
`Content-Type: application/json`	Content type.
`Accept: application/json; version=1.0`	API version.
`Authorization: <access_token>`	Authentication token (see the Authentication section for details).

Each parameter is specified by a pair of «key»:«value» inside the JSON document.

UTF-8 encoding must be used.

Below is an example of the request.

```
POST /api/bindings/ HTTP/1.1
Host: 3dsec.sberbank.ru/sbersafe
Content-Type: application/json
Accept: application/json; version=1.0
Authorization: 410012345678901.0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEF
GHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123


{

«phone»:«9997778866»

}
```

Safety requirements

- All network interactions are made only through HTTPS.
- The application must verify that the server's SSL certificate is valid. If the SSL certificate has not

passed verification, you must immediately terminate the session to prevent the authorization data leakage.
- Do not store the authentication token explicitly, including in the format of cookies files.
- Never use an authentication token in the POST request parameters.

Response format

The service response is a JSON document in UTF-8 encoding (see The application/json Media Type for JavaScript Object Notation (JSON) and the official website of JSON).

Each response has a unique `requestId` in the UUID format.

The content of the document depends on the result of the query execution (see example below).

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: <content-lenght>
Expires: Thu, 01 Dec 2018 16:00:00 GMT
Cache-Control: no-cache
{
  "response":{
    "param1":"value1",
    "param2":"value2"
  },
  "requestId":"requestId uuid",
  "status":"SUCCESS"
}
```

The response may also contain fields that are not described in this document. It should be ignored.

If authorization is rejected, the server responds with the HTTP code 4xx. Possible reasons of rejection:

- the request cannot be parsed;
- the request is missing an HTTP header `Authorization` (for more details see the Authentication section);
- the Authorization header specifies a nonexistent, invalid, or expired token (for more information see Authentication section);
- an operation was requested for which the token does not have rights.

The response has the `WWW-Authenticate` header (according to The OAuth 2.0 Authorization Framework: Bearer Token Usage).

If the request is rejected, the following fields are present in the response:

Field	Description	Example
code	Short error code. Can be used for automatic processing. For all development errors such as invalid input field format, non-existent object identifier, etc., use the same code, for example, `inernalError`.	expiredCard
description	Description of the error in your own English words. Not for automatic processing, but for debugging and to help the developer.	Required field 'pan' is empty
message	Message to display to the user.	Your card is expired

Codes reasons of rejection of the operation:

HTTP response code	Value of the `error_code` field	Description
400	invalidRequest	The HTTP request format does not match the protocol. The request cannot be parsed, or the header `Authorization` is missing or has an incorrect value.
429	tooManyRequests	The server is overloaded, try again later.

Below is an example of a response with an error message.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalidRequest", error_description="Invalid
request", error_message=""
{
  "error": {
    "code":"general.error",
    "description":"Some error occurred in Java",
    "message":"Internal error"
  },
  "requestId":"gbhjnkme-rdcfgv-hbjnkm-7689ui-okp3ew",
  "status":"FAIL"
}
```

Data type

The data type of the protocol	Corresponding JSON type	Description
string	string	UTF-8 encoded text string.
amount	number	Amount. Fixed-point number, two digits after the point.
boolean	boolean	Boolean value: true, false.
int	number	32-bit signed integer.
long	number	64-bit signed integer.
object	object	Nested JSON object.
array	array	Array of JSON objects.
datetime	string	Timestamp according to standard RFC3339 in the following format yyyy-MM-dd'T'HH:mm:ssZZ.

«datetime» format:

- `YYYY` – year, always 4 digits;
- `MM` – month, always 2 digits (for example, 01=January);
- `DD` – day of the month, exactly 2 digits (from 01 to 31);
- `T` – Latin character «T» in upper case;
- `hh` – hours, always 2 digits (24-hour format, from 00 to 23);
- `mm` – minutes, always 2 digits (from 00 to 59);
- `ss` – seconds, always 2 digits (from 00 to 59);
- `ZZ – Timezone descriptor, required parameter. Can have values: * Z – UTC, uppercase «Z»; *` «+hh or hh – offset relative to UTC (indicates that the specified local time is ahead of or behind UTC by the specified number of hours and minutes).

```
2012-01-31T12:00:00Z
or
2018-07-21T19:30:45+04:00 – 19 hours 30 minutes 45 seconds July 21, 2018, Europe/Moscow time
zone (UTC+04:00)
```