Description of the Programming Interface (API) of destream

1. Purpose of the programming interface (API)

The DeStream programming interface (API) allows you to interact with the platform to perform the following actions.

1. Obtaining information about payments (tips) to the user of the DeStream platform.
2. Initiation of payment (transfer of tips) to the user of the DeStream platform.
3. Obtaining information about the user of the DeStream platform.
4. User registration on the DeStream platform.
5. Receiving notifications about events:
    - Changing the status of payment (sending tips) to the user of the DeStream platform;
    - Information in the user profile of the DeStream platform has changed.

2. Requirements for the calling system

To connect an external system to the DeStream platform, follow these steps.

1. Inform the platform administration about the connecting system, and if the application is approved, receive the parameters **ClientId** and **ClientSecret**, which will be used to identify the connecting system and authenticate requests to the program interface.
2. Provide **AuthRedirectURL** - the URL to which the user will be redirected in case of successful authentication (OAuth 2) when adding the relationship between the connecting system and the user's account on the DeStream platform.
3. Provide **PaymentCallbackUrl** - URL of the method for receiving notifications about the status of payments (if it is required to receive callback notifications).
4. Provide **UserDataChangedCallbackUrl** - URL of the method for receiving notifications about information changes in the user profile of the DeStream platform (if it is required to receive callback notifications).

3. General information about the programming interface (API)

Interaction protocol: **HTTPS**.

Data presentation format: **JSON**.

In case of successful request, an HTTP response with a **200** code and the result of the operation in the response body is returned.

In case of an error, an HTTP response is returned with a code value greater than or equal to **400** and the following data structure in the response body describing the reason for the error.

```
{
      code: number;                 //error code
      error_message: string;        // textual description of the error
}
```

Error codes description:

Error code	Description
1	Could not authenticate the calling system. Possibly the authentication data contains a mistake or the access to the calling system has been blocked
2	The user access token (access_token or refresh_token) is incorrect or invalid. It is necessary to update the access_token using the **refresh_token** or authenticate the user again to receive new tokens.
3	Unable to register the user. The provided email is already registered with a different user.
4	The calling system is configured incorrectly. Example: The provided **AuthRedirectURL** is different to the one provided in the authentication request or the token request.
5	The provided X-Api-Signature content is incorrect.
6	The allowed time interval between the date and time in the header of X-Api-RequestDate and the time of receiving this request by the server for processing has been exceeded.
1000	General error code in case it is not described in the error code list. Details are sent in the **error_message** field.

4. Access restriction

4.1. Restricting access to the programming interface (API)

To access the DeStream API the calling system is required to identify and authenticate itself.

Identification is performed using the **ClientId** value previously issued to the calling system.

Authentication is performed using the **ClientSecret** value previously issued to the calling system.

The **ClientId** and **ClientSecret** sparameters are passed in the GET parameters of the same name, or are used to form the values of additional HTTP headers **X-Api-ClientId** and **X-Api-Signature**.

4.2. Restricting access to user data

Some API methods that provide user data require the DeStream platform user to grant permission to access their data. Granting access is confirmed by issuing an access token to the calling system, which must then be used when accessing the appropriate API methods.

The token is issued as a result of authentication according to the OAuth 2 standard.

The DeStream platform must first be provided with the value of the **AuthRedirectUrl** parameter - the URL on the page of the calling system to redirect the user after passing the authentication.

The scenario for performing authentication is as follows.

1. From the page of the calling system where the DeStream platform user is connected to the calling system, redirect the user to the DeStream platform URL.

   GET
   *https://destream.net/oauth2/authorize?response_type=code&client_id=<**ClientId**>&redirect_uri=<**AuthRedirectUrl**>&scope=**profile+tips**&state=<**random sequence**>*

   Where

   response_type = code - indicates the beginning of the data exchange process to gain access;

   <ClientId> - the identifier of the calling system;

   <AuthRedirectUrl> - the URL to which the user will be redirected after passing authentication, must match the previously provided AuthRedirectUrl;

   profile + tips - designation of blocks of user data to which access is requested: profile information and information about payments (tips), the "+" sign is used to separate access identifiers;

   <random sequence> - any unique (random) alphanumeric sequence that will be transmitted to the caller when redirecting the user after the completion of the authentication procedure in unchanged form. Must be checked on the side of the calling system, it is necessary to prevent CSRF attacks.

   As a result, the user will be prompted to pass the authentication on the DeStream platform, and will also be asked for permission to provide data to the calling system.

   If the user logged in and granted permissions to the external system, they will be redirected to the page of the calling system specified in the *redirect_url* parameter. The request will pass the **code** parameter with the value of the authorization code, which should subsequently be used to obtain an access token. The **state** parameter will also be passed with a **unique token** value, which must be checked for immutability to protect against CSRF attacks.

In case of an error, a response will either be returned in accordance with the description in section “3. General information about the programming interface (API) ", or redirected to the page of the calling system specified in the *redirect_url* parameter, describing the **error** parameter in the request to display to the user.

2. Get an access token by making a request

POST

*https://api.destream.net/api/v2/oauth2/token?grant_type=authorization_code&client_id=<**ClientId**>&client_secret=<**ClientSecret**>&redirect_uri=<**AuthRedirectUrl**>&code=<**code**>*
Where

grant_type = authorization_code - indicates that the authorization code will be used for access;

<ClientId> - the identifier of the calling system;

<ClientSecret> - the "secret" of the calling system;

<AuthRedirectUrl> - URL for redirecting the user after passing authentication, must match the previously provided **AuthRedirectUrl**, used for integrity control;

<code> is the authorization code that was received in the *code* parameter when returning to the address of the calling system after the successful execution of the *authorize* method (the first step).

The *Content-Type* HTTP header must be passed with the value *application/x-www-form-urlencoded*.

Content-Type: *application/x-www-form-urlencoded*

If the method is successfully executed, a JSON object of the following type will be received in the response.

```
{
      access_token: string;
      refresh_token: string;
      expires_in: number;
      scope: string;
      token_type: string;
}
```

Where

access_token - contains an access token to methods that return user data. The token has a limited duration. Following its expiration, the token must be renewed. Access to data using the expired token will be terminated. The corresponding error will be returned when calling access methods to user data;

refresh_token - contains a token for refreshing the access token after its expiration date;

expires_in - contains the time in seconds after which the access token will expire;

scope - contains the designation of user data blocks to which access is granted through the issued access token;

token_type - contains the type of access token.

If an error occurs, the response will be returned according to description in section “3. General information about the programming interface (API) ". In particular, if the authorization code has expired or is not valid, a response with a 401 Unauthorized HTTP code will be returned. In this case, you need to re-execute the oauth2 / authorize user access request method (step 1).

3. In case of expiration of the access token, it is necessary to update it by calling the next method.

   POST

   *https://api.destream.net/api/v2/oauth2/token?grant_type=refresh_token&client_id=<**ClientId**>&client_secret=<**ClientSecret**>&scope=<**scope**>&refresh_token=<**refresh_token**>*

   Where

   grant_type=refresh_token - indicates that a refresh token will be used for access access token (refresh token);

   <ClientId> is the ID of the calling system;

   <ClientSecret> - "secret" of the calling system;

   <scope> is the value of the scope field received in the JSON object in response to the token method call (step 2) along with the access token whose value needs to be updated;

   <refresh_token> is the value of the refresh_token field received in the JSON object in the response to the method call token (step 2) along with the access token whose value needs to be updated.

   The Content-Type HTTP header must be passed with the value application/x-www-form-urlencoded.

Content-Type: application/x-www-form-urlencoded

If the method is successfully executed, the response will receive a JSON object of the same form as in executing the method in step 2 with the updated value of the access token. In case of an error, the result is completely identical to the description of the method call in step 2.

In case of an error, a response will be returned in accordance with the description in section “3.
General information about the program interface (API). In particular, if the refresh token has expired or not is valid, a 401 Unauthorized HTTP response will be returned. In this case, it is necessary re-execute the access request method on the oauth2/authorize user.

5. Programming interface (API) methods

5.1. Members

5.1.1. Creating a user on the platform

The user creation method allows you to automatically create a user on the DeStream platform with his email. A user can be created only if the passed email is not already in use by an existing platform user.

Method
POST
URL
https://api.destream.net/api/v2/users/register
Request headers
X-Api-ClientId contains **ClientId**. X-Api-RequestDate contains the date and time when the request was formed in ISO8601 format. X-Api-Signature contains the checksum of the content of the request body according to the formula SHA-512 (ClientId + X-Api-RequestDate + ClientSecret), meaning that SHA-512 (HASH) is derived from the **ClientId** concatenation, the date and time of the request, and the system's **ClientSecret** value, which creates the payment.
Request body

`{`
`email: string;` // e-mail of the platform user being created
`}`

Execution result

`{`
`data: {`
`user_id: string;` // Unique identifier of the created user
`email_confirmed: boolean;` // Is the email address verified by the user?
`email: string;` // e-mail of the created user
`nickname: string;` // Nickname of the created user
`user_token: {`
`access_token: string;` // Access token for information about the created user
`refresh_token: string;` // Refresh token access token

```
        expires_in: number;          // Time of validity of the access token
        scope: string;               // User data blocks to which the token grants access
        token_type: string;          // Token type
    }
  }
}
```

In case of error

If an error occurs, the response will be returned according to the description in section “3. General information about the programming interface (API) ".

In particular, if the user with the passed email already exists on the platform, a response with the *HTTP code 409 Conflict* will be returned.

The value of the fields from the user_token object is used to obtain information about the created user through the appropriate methods.

5.1.2. Retrieving information from the platform user profile

The method for obtaining information from the user profile returns data about the user and his settings on the platform.

Method
GET
URL
https://api.destream.net/api/v2/users
Request headers
X-Api-ClientId contains **ClientId**. Authorization contains a string like "**token_type** + space + **access_token**" (without quotes and plus signs), where the values of token_type and access_token are taken from the fields of the same name of the access token object returned by the auth2 / token method or by the users / register user creation method.
Request body
Not transmitted (empty).

Execution result

```
{
 data: {
 user_id: string;              // Unique user ID
 email_confirmed: boolean;     // Is the email address verified by the user?
 email: string;                // user's e-mail
 nickname: string;             // Username
 limits: [                     // Array of objects with the values of the max and min tip amount
     {                         // for each set currency, one array element per currency
        currency: string;      // EUR, RUB, USD
        min: number;           // Value of the minimum payment amount (tip)
        max: number;           // Value of the maximum payment amount (tip)
      }
   ]
  }
}
```

In case of error

In case of an error, a response will be returned in accordance with the description in section “3. General information about the programming interface (API) ".

In particular, if the access token has expired or is not valid, a 401 Unauthorized HTTP response will be returned. You need to execute the access token refresh method using the refresh token (refresh_token) by the oauth2 / token method.

5.1.3. Obtaining information about payments received by the user (tips)

The method for obtaining information about payments received by the user returns a list of data blocks about payments to the user, starting with the latest.

Method

GET

URL

https://api.destream.net/api/v2/users/tips?offset=<**number of items to skip**>&limit=<**number of items per response**>&after_date=<**date in ISO8601 format**>

Request parameters

<number of items to skip> —number of payment list items to be skipped from

beginning of the list, sorted by payment date in descending order (latest to earliest);

optional parameter, if not passed, the value is 0, i.e. the list will contain elements,

starting from the latest payment;

<number of items per response> - the number of items in the payment list that you want to return in response to a request; optional parameter, if not passed, the value is 10, that is, the response will contain a list often elements; the maximum possible value is 30;

<date in ISO8601 format> — date and time in ISO8601 format, payments made on or after,

to include in the list; optional parameter, if not passed, all payments are returned, starting with the latest one.

Request headers

X-Api-ClientId contains the **ClientId**.

Authorization contains a string like ”**token_type**+space+**access_token**” (without quotes and pluses),

where the token_type and access_token values are taken from the same-named fields of the access token object, returned by the auth2/token method or the users/register user creation method.

Request Body

Not transmitted (empty).

Execution result

```
{
 data: [{
  user_id:string       // Unique user ID of the payee
  sender:string;       // Nickname of the sender of the payment
  payment_id: string   // Payment ID
  amount: number;      // Amount of payment
  currency:string;     // Payment currency EUR, RUB, USD
  message:string;      // Message from the sender of the payment (tip)
  date:string;         // Date and time of payment in ISO8601 format
  additional_data: string;  // Additional parameters in the format "parameter=value"
                       // which were received when creating the payment
 }],
 total:number;         // Total number of payments to the user (starting from after_date if
                       // passed)
 response_date:string;  // Date and time of the server response in ISO8601 format.
}
```

In case of error

In case of an error, a response will be returned in accordance with the description in section “3. Are common information about the program interface (API)”.

5.1.4. Subscribe to notifications about payments received by the user

Subscribing to notifications about payments received by the user allows you to receive information about payments received by the user (tips) in real time via web socket. Published only successfully completed payments and PayPal payments are in processing status. For interaction you can use the client library of your programming language to interact with SignalR.

Method
GET
URL
Access point address for receiving alerts https://api.destream.net/ws/v2/donations?client_id=<ClientId>&access_token=<UserAcessToken>
Request parameters
<ClientId> is the ID of the calling system. **<UserAccessToken>** is the access token returned in the access_token field of the user_token object obtained in result of executing the oauth2/token method.
Execution result

Event "**donationReceived**" with data in json format:
`{`
`  user_id:string` // Unique user ID of the payee
`  sender:string;` // Nickname of the sender of the payment
`  payment_id: string` // Payment ID
`  amount: number;` // Amount of payment
`  currency:string;` // Payment currency EUR, RUB, USD
`  message:string;` // Message from the sender of the payment (tip)
`  date:string;` // Date and time of payment in ISO8601 format
`  additional_data: string;` // Additional parameters in the format "parameter=value"
`}`

In case of error
In case of an error, a response will be returned in accordance with the description in section “3. Are common information about the program interface (API).

5.2. Payments

5.2.1. Create payment

The payment creation method generates a URL for redirecting to the payment method selection form and making a payment (sending a tip) to the platform user. The tipping user of the calling system must be redirected to the resulting URL.

The received URL is valid for 24 hours (expiration time may be changed).

Method
POST
URL
https://api.destream.net/api/v2/payments
Request headers
X-Api-ClientId contains the **ClientId**. X-Api-RequestDate contains the date and time the request was generated in ISO8601 format. X-Api-Signature contains a checksum of the content of the request body according to the formula SHA-512(ClientId+XApi-RequestDate+ClientSecret), that is, SHA-512 (HASH) is taken from the concatenation of **ClientId**, date and time request and the **ClientSecret** value of the system that creates the payment.
Request body

```
{
    user_id: string;           // Unique identifier of the payment receiving user
    amount: number;
    currency: string;          // EUR, RUB, USD
    message: string;
    success_url: string;       // URL to be redirected in case of successful payment
    fail_url: string;          // URL to redirect to in case of a problem payment
    additional_data: string;  // any set of additional parameters in the "parameter =
                               //value" format.
                               // maximum length is 100 characters
                               // which will be returned unchanged upon request
                               // information about a payment or in a notification about
                               // a change in its status
}
```

Execution result

```
{
   data: {
```

```
      payment_url: string;   // URL of the payment method selection form and
                             // making a payment
      payment_id: string;    // payment ID
      user_id: string;       // id of the user who receives the payment (tip)
   }
}
```

In case of error

In case of an error, a response will be returned in accordance with the description in section “3. General information about the programming interface (API) ".

5.2.2. Getting information about payments

The method for getting information about payments returns a list of payment data blocks that were previously created by the calling system, starting with the latest.

Method

GET

URL

https://api.destream.net/api/v2/payments?offset=<number of items to skip>&limit=<number of items per response>&after_date=<date in ISO8601 format>&payment_ids=<id1>,<id2>…

Request parameters

<number of items to skip> —number of payment list items to be skipped from

beginning of the list, sorted by payment date in descending order (latest to earliest);

optional parameter, if not passed, the value is 0, i.e. the list will contain elements,

starting from the latest payment;

<number of items per response> - the number of items in the payment list that you want to return in response to a request; optional parameter, if not passed, the value is 10, that is, the response will contain a list of ten elements; the maximum possible value is 30;

<date in ISO8601 format> — date and time in ISO8601 format, payments made on or after, to include in the list; optional parameter, if not passed, all payments are returned, starting with the latest one.

<id1>,<id2>… — comma-separated list of payment identifiers payment_id,

returned by the payment creation method, which should be returned in the response; maximum you can specify up to 20 identifiers; optional parameter, if not passed, all payments will be returned.

Request headers

X-Api-ClientId contains the **ClientId**.

X-Api-RequestDate contains the date and time the request was generated in ISO8601 format.

X-Api-Signature contains a checksum of the content of the request body according to the formula SHA-512(ClientId+XApi-RequestDate+ClientSecret), that is, SHA-512 (HASH) is taken from the concatenation of **ClientId**, date and time request and the **ClientSecret** value of the system that creates the payment.

Request Body

Not transmitted (empty).

Execution result

```
{
  data: [{
    user_id:string              // Unique user ID of the payee
    sender:string;              // Nickname of the sender of the payment
    payment_id: string          // Payment ID
    amount: number;             // Amount of payment
    currency:string;            // Payment currency EUR, RUB, USD
    message:string;             // Message from the sender of the payment (tip)
    date:string;                // Date and time of payment in ISO8601 format
    additional_data: string;    // Additional parameters in the format "parameter=value"
                                // which were received when creating the payment
    transaction_id: string      // Payment transaction ID
    transaction_status_code: number;  // Payment status code:
                                // -1 — transaction declined (DECLINED)
                                // 0 - new transaction status (NEW)
                                // 1 - transaction in processing (PROCESSING)
                                // 2 - transaction completed successfully (COMPLETED)
    transaction_status_text: string;  // Payment status text
  }],
  total:number;                 // Total number of payments to the user (starting from after_date if
                                //passed)
  response_date:string;         // Date and time of the server response in ISO8601 format.
}
```

In case of error

In case of an error, a response will be returned in accordance with the description in section “3. Are common information about the program interface (API).

6. Notifications

6.1. General information

Notifications are designed to inform connected systems about changes on the platform Destream. Implemented notifications about changes in user information and new states payments.

Notifications are sent as an HTTP(S) POST request with Content-Type: application/json.

The additional X-Signature header of the sent POST request contains a checksum the content of the request body according to the formula SHA-512(body+**ClientSecret**), that is, the SHA-512 hash is taken from concatenation of the request body and the **ClientSecret** value of the system to which the notification is sent.

The checksum must be verified by a checksum calculation on the side recipient and comparing it with the checksum passed in the header. Equality of specified values will confirm that the content of the request is unchanged during transmission.

A notification is considered delivered if a response with an HTTP 200 OK status was received in response to the request.

6.2. Payment Status Change Notifications

A notification about a change in the payment status is sent to the external system by calling **PaymentCallbackUrl**.

The following data structure is passed in the body of the method call request.

```
{
  data: {
    user_id:string            // Unique user ID of the payee
    sender:string;            // Nickname of the sender of the payment
    payment_id: string        // Payment ID
    amount: number;           // Amount of payment
    currency:string;          // Payment currency EUR, RUB, USD
    message:string;           // Message from the sender of the payment (tip)
    date:string;              // Date and time of payment in ISO8601 format
    additional_data: string;  // Additional parameters in the format "parameter=value"
    transaction_id: string // Payment transaction ID
    transaction_status_code: number;  // Payment status code:
                              // -1 — transaction declined (DECLINED)
                              // 1 - transaction in processing (PROCESSING)
                              // 2 - transaction completed successfully (COMPLETED)
   transaction_status_text: string;  // Payment status text
  }
}
```

6.3. User Information Change Notification

A notification about a change in user information is sent to an external system via calling **UserDataChangedCallbackUrl**. The following data structure is passed in the body of the method call request.

```
{
  data: {
        user_id:string                  // Unique user ID
        email_confirmed: boolean;       // Is the email address verified by the user
        email:string;                   // user's e-mail
```

```
        nickname:string;                // User nickname
        limits: [                       // An array of objects with the values of the
                                        // maximum and minimum payment
                                        // amounts (tips)
        {                               // for each set currency, one array element
                                        // per currency
          currency:string;              // EUR, RUB, USD
          min: number;                  // The value of the minimum payment
                                        // amount (tip)
          max: number;                  // The value of the maximum payment
                                        // amount (tip)
        }
      ]
   }
}
```

Note. In the current implementation, notifications are sent only upon confirmation email address user.