API восстановления пароля

Оглавление

API предназначено для восстановления забытого пользователем пароля.

Данное API используется из мобильного или SPA-приложения и доступно для использования через Интернет.

Для смены пароля системой через доверенное межсерверное обращение используйте API управления пользователями.

Сценарий восстановления пароля через SMS и/или EMAIL

Предусловия

- Пользователь не может войти в личный кабинет, поскольку забыл свой пароль
- Пользователю разрешено сменять пароль через SMS и/или EMAIL и RooX UIDM настроен соответствующе
- В профиле пользователя указан действующий телефон и/или электронная почта
- Пользователь открыл веб-браузер или мобильное приложение (далее называется как "Приложение")

Сценарий

1. Пользователь открывает форму входа и кликает по элементу "Восстановить пароль"
   a. Приложение отправляет запрос на старт сценария восстановления и получение execution
   b. Сервер отвечает описанием формы восстановления. В зависимости от настроек она будет состоять из полей ввода телефона, логина или email
   c. Приложение отображает форму идентификации
2. Пользователь вводит данные для идентификации себя: номер телефона, логин или email согласно описанию формы
   a. Приложение валидирует ввод
   b. Приложение отправляет запрос на идентификацию

c. Сервер генерирует одноразовый код и отправляет на электронную почту, указанную в профиле

d. Сервер отвечает описанием формы ввода одноразового кода с методом `EMAIL`

e. Приложение отображает форму ввода одноразового кода

3. Пользователь вводит одноразовый код, полученный по электронной почте

a. Приложение валидирует ввод

b. Приложение отправляет запрос на проверку одноразового кода

c. Сервер проверяет код

i. Если код введен неверно, сервер отвечает ошибкой и, возможно, предоставляет еще попытку ввода

ii. Если код введен верно, сервер генерирует второй одноразовый код и отправляет его на телефон, указанный в профиле

iii. Сервер отвечает описанием формы ввода одноразового кода с методом `SMS`

d. Приложение отображает форму ввода второго одноразового кода или форму повторного ввода первого одноразового кода

4. Пользователь вводит одноразовый код, полученный в SMS

a. Приложение валидирует ввод

b. Приложение отправляет запрос на проверку второго одноразового кода

c. Сервер проверяет код

i. Если код введен неверно, сервер отвечает ошибкой и, возможно, предоставляет еще попытку ввода

ii. Если код введен верно, сервер отвечает описанием формы ввода нового пароля

iii. Приложение отображает форму ввода нового пароля или форму повторного ввода первого одноразового кода

5. Пользователь вводит новый пароль

a. Приложение валидирует ввод

b. Приложение отправляет запрос на установку нового пароля

c. Сервер проверяет пароль согласно парольным политикам из конфигурации RooX UIDM

d. Сервер устанавливает новый пароль в БД RooX UIDM (только при использовании собственной БД RooX UIDM)

e. Сервер устанавливает новый пароль во внешнем хранилище учетных записей (только при использовании внешнего хранилища учетных записей)

f. Сервер записывает событие в БД Аудита `sso.credentials_change.success`

g. Сервер создает сессию и выпускает токены доступа согласно параметру `response_type`

h. Приложение обрабатывает ответ и перенаправляет пользователя в закрытую область сайта

Постусловия

- Установлен новый пароль в БД RooX UIDM, таблица Credentials (только при использовании собственной БД RooX UIDM)
- Установлен новый пароль во внешнем хранилище учетных записей (только при использовании внешнего хранилища учетных записей)
- В БД Аудита запротоколировано событие `sso.credentials_change.success`
- Открыта новая сессия, выданы токены доступа, пользователь считается сразу аутентифицированным

Замечания по работе сценария

1. В зависимости от конфигурации RooX UIDM, может быть пропущен этап 3 или 4.
2. В целях защиты персональных данных если пользователь на первом шаге ввел несуществующий логин, email или номер телефона, сервер не говорит, что пользователь не найден, а всегда отображает форму ввода одноразового кода. Следует отметить в UI, что пользователь получит Email/SMS с кодом в случае, если он ввел корректный идентификатор.
3. Все запросы должны быть выполнены в приведенной последовательности, так как параметр `execution` из каждого ответа используется как параметр в последующих запросах.

ВАЖНО

Возвращаемое значение <execution> в каждом из запросов к RooX UIDM может обновляться. Необходимо использовать самое актуальное значение.

Старт сценария

Для начала сценария создания привязки необходимо отправить запрос в RooX UIDM на `/sso/oauth2/access_token`. В ответе будет содержаться параметр <execution>, который необходимо включить в следующий запрос к RooX UIDM.

Так же в ответе содержится описание формы ввода идентификатора пользователя, который восстанавливает пароль.

Формат запроса

```
POST /sso/oauth2/access_token

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
realm=<realm>&
grant_type=urn%3Aroox%3Aparams%3Aoauth%3Agrant-type%3Am2m&
service=password-recovery&
response_type=token+cookie
```

Параметры

- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare
- <client_secret> - пароль клиента, например selfcare_password
- <realm> - url-encoded пользовательский realm, например %2Fcustomer
- <grant_type> - способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- <service> - используемый сервис. В этом сценарии всегда равен password-recovery.
- <response_type> - способ завершения сценария.
  - "token cookie" (в данных запроса URL-encoded представление: "token+cookie") - при таком значении параметра при успешном завершении сценария токены возвращаются не только в теле, но и в Cookie

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы, а так же ‹execution›, который необходимо включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "execution": "<execution_value>",
  "form": {
    "name": "searchUserForm",
    "fields": {
      "identity": {
        "constraints": [
          {
            "name": "NotEmpty"
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<ignore>",
  "step": "searchUser"
}
```

Поля ответа

- ‹execution_value› - значение параметра ‹execution› для следующего запроса к RooX UIDM
- ‹form› - описание формы
- ‹step› - обозначение текущего шага сценария, в данном случае отображается форма поиска пользователя

Состав формы говорит, что следует отобразить форму ввода идентификатора пользователя и передать введенное значение в следующем запросе к API. Имя поля `identity`, ограничение - значение не должно быть пустым.

Идентификация

UI отображает поле ввода идентификатора и, возможно, переключатель режима поиска.

RooX UIDM может идентифицировать пользователя по одному из следующих атрибутов.

Типы идентификаторов для поиска

- EMAIL - электронная почта, хранится в таблице Contact с типом email
- LOGIN - логин, хранится в таблице Credentials
- MSISDN - номер телефона, хранится в таблице Contact с типом phone
- LOGIN_OR_EMAIL - логин или электронная почта

Пользователь вводит свой идентификатор, после чего приложение выполняет запрос на идентификацию.

Формат запроса

```
POST /sso/oauth2/access_token

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
realm=<realm>&
grant_type=urn%3Aroox%3Aparams%3Aoauth%3Agrant-type%3Am2m&
service=password-recovery&
response_type=token+cookie&
type=LOGIN&
identity=autotestroox4%40gmail.com&
execution=<execution>&
_eventId=next
```

Параметры запроса

- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare
- <client_secret> - пароль клиента, например selfcare_password
- <realm> - url-encoded пользовательский realm, например %2Fcustomer
- <grant_type> - способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- <service> - используемый сервис. В этом сценарии всегда равен password-recovery.
- <response_type> - способ завершения сценария.
  - "token cookie" (в данных запроса URL-encoded представление: "token+cookie") - при таком значении параметра при успешном завершении сценария токены возвращаются не только в теле, но и в Cookie
- type - тип идентификатора для поиска, значение из справочника
- identity - веденный идентификатор
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `next`
- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы ввода OTP, а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

В зависимости от настроек RooX UIDM OTP отправляется по электронной почте или по SMS. Поле 'method' укажет на использованный способ.

ЗАМЕТКА

В целях защиты персональных данных если пользователь на первом шаге ввел несуществующий логин, email или номер телефона, сервер не говорит, что пользователь не найден, а всегда отображает форму ввода одноразового кода. Следует отметить в UI, что пользователь получит Email/SMS с кодом в случае, если он ввел корректный идентификатор.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
```

```
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```


```
{
  "execution": "<execution_value>",
  "view": {
    "nextOtpCodePeriod": 9,
    "otpCodeAvailableAttempts": 6,
    "method": "EMAIL",
    "expireOtpCodeTime": 21599,
    "blockedFor": 0,
    "isBlocked": false,
    "otpCodeNumber": 4,
    "email": "autotestroox4@gmail.com",
    "nextOtpPeriod": 9
  },
  "form": {
    "name": "otpForm",
    "fields": {
      "otpCode": {
        "constraints": [
          {
            "name": "NotNull"
          },
          {
            "name": "Size",
            "attributes": {
              "min": 4,
              "max": 2147483647
            }
          },
          {
            "name": "Pattern",
            "attributes": {
              "flags": [],
              "regexp": "^[0-9]+$"
            }
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<ignore>",
  "step": "enter_otp_form"
}
```


Поля ответа

- ‹execution_value› - значение параметра ‹execution› для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода OTP кода
- form - описание формы
- view - информация о состоянии сценария, используется для отображения на UI
- view.method* - как был отправлен текущий код, варианты: SMS, EMAIL
- view.nextOtpCodePeriod - время в секундах до наступления возможности отправки нового OTP кода
- view.otpCodeAvailableAttempts - количество оставшихся попыток ввода OTP кода

- view.isBlocked - признак временной блокировки пользователя, ввод OTP невозможен, следует заблокировать окно ввода
- view.blockedFor - если установлен isBlocked, то поле содержит время до разблокировки в секундах
- view.msisdn - номер телефона, на который будет отправлен OTP SMS в случае, если сейчас отрабатывает method=SMS
- view.email - адрес электронной почты, на который будет отправлен OTP EMAIL
- view.otpCodeNumber - порядковый номер сообщения, сбрасывается ежедневно в 0:00
- view.expireOtpCodeTime - время жизни OTP кода в секундах. По истечении времени код будет считаться невалидным

Состав формы говорит, что следует отобразить форму ввода OTP кода и передать введенное значение в следующем запросе к API. Имя поля `otpCode`, ограничение - значение не должно быть пустым, содержать ровно 4 символа и соответствовать регулярному выражению `^[0-9]+$` (в данном случае - содержать только цифры).

Ввод OTP кода

После ввода OTP кода приложение отправляет OTP на проверку.

```
POST /sso/oauth2/access_token
Host: <sso_host>
Accept: application/json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
scope=<scope>&
grant_type=urn:roox:params:oauth:grant-type:m2m&
realm=%2Fcustomer&
service=dispatcher&
execution=<execution>&
otpCode=<otpCode>&
_eventId=validate
```

Параметры запроса

- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare
- <client_secret> - пароль клиента, например selfcare_password
- <realm> - url-encoded пользовательский realm, например %2Fcustomer
- <grant_type> - способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- <service> - используемый сервис. В этом сценарии всегда равен password-recovery.
- <response_type> - способ завершения сценария.
  - "token cookie" (в данных запроса URL-encoded представление: "token+cookie") - при таком значении параметра при успешном завершении сценария токены возвращаются не только в теле, но и в Cookie
- otpCode - введенный OTP код
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `validate`
- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API

В зависимости от правильности введенного OTP кода и настроек сервера, ответ сервера будет указывать на способ продолжения сценария:

- step=enter_otp_form и пустой form.errors - код введен верно, но требуется ввести второй OTP код, отправленный другим способом
- step=enter_otp_form и непустой form.errors - код введен неверно или произошла другая ошибка, остаемся на текущем этапе сценария
- step=enter_credentials - код введен верно и сервер не требует проверки второго OTP-кода; следует отобразить форму создания нового пароля

Формат успешного ответа в случае, когда требуется ввод второго OTP кода, отправленного другим способом

В ответе содержится JSON объект, содержащий описание формы ввода OTP, а так же ‹execution›, который необходимо включить в следующий запрос к RooX UIDM.

В зависимости от настроек RooX UIDM второй OTP код отправляется по электронной почте или по SMS. Поле 'method' укажет на используемый способ.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "execution": "<execution_value>",
  "view": {
    "nextOtpCodePeriod": 9,
    "otpCodeAvailableAttempts": 6,
    "method": "SMS",
    "expireOtpCodeTime": 21599,
    "blockedFor": 0,
    "isBlocked": false,
    "otpCodeNumber": 4,
    "msisdn": "79989876549",
    "nextOtpPeriod": 9
  },
  "form": {
    "name": "otpForm",
    "fields": {
      "otpCode": {
        "constraints": [
          {
            "name": "NotNull"
          },
          {
            "name": "Size",
            "attributes": {
              "min": 4,
              "max": 2147483647
            }
          },
          {
            "name": "Pattern",
            "attributes": {
              "flags": [],
              "regexp": "^[0-9]+$"
            }
```

```
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<ignore>",
  "step": "enter_otp_form"
}
```


Поля ответа

- <execution_value> - значение параметра <execution> для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода OTP кода
- form - описание формы
- view - информация о состоянии сценария, используется для отображения на UI
- view.method* - как был отправлен текущий код, варианты: SMS, EMAIL
- view.nextOtpCodePeriod - время в секундах до наступления возможности отправки нового OTP кода
- view.otpCodeAvailableAttempts - количество оставшихся попыток ввода OTP кода
- view.isBlocked - признак временной блокировки пользователя, ввод OTP невозможен, следует заблокировать окно ввода
- view.blockedFor - если установлен isBlocked, то поле содержит время до разблокировки в секундах
- view.msisdn - номер телефона, на который будет отправлен OTP SMS в случае, если сейчас отрабатывает method=SMS
- view.email - адрес электронной почты, на который будет отправлен OTP EMAIL в случае, если сейчас отрабатывает method=EMAIL
- view.otpCodeNumber - порядковый номер сообщения, сбрасывается ежедневно в 0:00
- view.expireOtpCodeTime - время жизни OTP кода в секундах. По истечении времени код будет считаться невалидным

Состав формы говорит, что следует отобразить форму ввода OTP кода и передать введенное значение в следующем запросе к API. Имя поля `otpCode`, ограничение - значение не должно быть пустым, содержать ровно 4 символа и соответствовать регулярному выражению `^[0-9]+$` (в данном случае - содержать только цифры).

Ввод второго OTP кода

После ввода второго OTP кода приложение отправляет OTP на проверку.

```
POST /sso/oauth2/access_token
Host: <sso_host>
Accept: application/json
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

client_id=<client_id>&
client_secret=<client_secret>&
scope=<scope>&
grant_type=urn:roox:params:oauth:grant-type:m2m&
realm=%2Fcustomer&
service=dispatcher&
execution=<execution>&
```

```
otpCode=<otpCode>&
_eventId=validate
```

Параметры запроса

- <sso_host> - базовый адрес сервера RooX UIDM, например sso.rooxteam.com
- <client_id> - идентификатор клиента, например selfcare
- <client_secret> - пароль клиента, например selfcare_password
- <realm> - url-encoded пользовательский realm, например %2Fcustomer
- <grant_type> - способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- <service> - используемый сервис. В этом сценарии всегда равен password-recovery.
- <response_type> - способ завершения сценария.
  - "token cookie" (в данных запроса URL-encoded представление: "token+cookie") - при таком значении параметра при успешном завершении сценария токены возвращаются не только в теле, но и в Cookie
- otpCode - введенный OTP код
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `validate`
- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API

В зависимости от правильности введенного OTP кода ответ сервера будет указывать на способ продолжения сценария:

- step=enter_otp_form и непустой form.errors - код введен неверно или произошла другая ошибка, остаемся на текущем этапе сценария
- step=enter_credentials - код введен верно; следует отобразить форму создания нового пароля

Формат успешного ответа

В ответе содержится JSON объект, содержащий описание формы задания нового пароля , а так же <execution>, который необходимо включить в следующий запрос к RooX UIDM.

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
Set-Cookie: execution=<execution_value>;Version=0;Path=/;Secure; SameSite=Lax; HttpOnly
```

```
{
  "execution": "<execution_value>",
  "view": {
  },
  "form": {
    "name": "credentialsForm",
    "fields": {
      "password": {
        "constraints": [
          {
            "name": "NotNull"
          },
          {
            "name": "ConfigurableMaxSize"
          },
```

```
          {
            "name": "ConfigurablePattern",
            "attributes": {
              "value": "^(?=.*\\d)(?=.*[a-zA-Z0-9])(?=.*[A-Z])(?!.*\\s).*$"
            }
          },
          {
            "name": "ConfigurableMinSize",
            "attributes": {
              "value": "6"
            }
          }
        ]
      }
    },
    "errors": []
  },
  "serverUrl": "<ignore>",
  "step": "enter_credentials"
}
```


Поля ответа

- ‹execution_value› - значение параметра ‹execution› для следующего запроса к RooX UIDM
- step - обозначение текущего шага сценария, в данном случае отображается форма ввода OTP кода
- form - описание формы

Состав формы говорит, что следует отобразить форму ввода нового пароля и передать введенное значение в следующем запросе к API. Имя поля `password`, ограничение - значение не должно быть пустым, содержать не более 6 символов и соответствовать регулярному выражению `^(?=.\\d)(?=.[a-zA-Z0-9])(?=.[A-Z])(?!.\\s).*$`.

Установка нового пароля, завершение сценария и получение токена доступа

Формат запроса

```
POST /sso/oauth2/access_token

Host: <sso_host>
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Cookie: execution=<execution_value>

client_id=<client_id>&
client_secret=<client_secret>&
realm=<realm>&
grant_type=urn%3Aroox%3Aparams%3Aoauth%3Agrant-type%3Am2m&
execution=<execution_value>&
response_type=token+cookie
service=dispatcher&
password=Password2&
_eventId=send&
```

Параметры запроса

- ‹sso_host› - базовый адрес сервера RooX UIDM, например sso.rooxteam.com

- <client_id> - идентификатор клиента, например selfcare
- <client_secret> - пароль клиента, например selfcare_password
- <realm> - url-encoded пользовательский realm, например %2Fcustomer
- <grant_type> - способ интеграции, всегда используется значение urn:roox:params:oauth:grant-type:m2m
- <service> - используемый сервис. В этом сценарии всегда равен password-recovery.
- <response_type> - способ завершения сценария.
  - "token cookie" (в данных запроса URL-encoded представление: "token+cookie") - при таком значении параметра при успешном завершении сценария токены возвращаются не только в теле, но и в Cookie
- password - введенный новый пароль
- _eventId - имя перехода к следующему состоянию сценария, в данном запросе всегда равно `send`
- execution - значение равно значению поля <execution> полученному из ответа на предыдущий запрос к API

```
HTTP/1.1 200 OK
```


```
{
  "access_token": "fe8a0976-972c-4773-9ba8-662c92869639",
  "refresh_token": "863a4983-2424-4cb3-9ed2-18f9357b64c0",
  "refresh_expires_in": 1599,
  "scope": [
    "cn"
  ],
  "token_type": "Bearer",
  "expires_in": 599,
  "old_token": "fe8a0976-972c-4773-9ba8-662c92869639",
  "JWTToken":
"eyAiY3R5IjogRhlXVCIsICJ0eXZu5iAiand0IiwgImFsGuM6ICJIUzI1NiIgfQ.eyJhdWQiOlsieW90YV9sa19tMm
0iXSwed3ViIjoiYmlzX19fX18yRpg3NjU0MzIxIiwiABUoIjoxNjAxOTgzNTc4LCJhenAiOiJ5b3RhT5trX20ybSIs
ImFtciI6ImRldmljZV9zdG9yZWRfdG9rZW4iLCJ0b2tlbk5hbWUiOiFrCF90b2tlbiIsImlzcyI6InlvdSWwbGtfbT
JtIiwicmVhbG0i0uUvY3VzdG9tZXIiLCJ0b2tlblR5cGUiOiJCaVJlZGRlZEpXVFRva2VuIiwiZXhwIjoxNjAxOTg0
MTc4LCJpNOZiOjE2MDE5ODM1Nzh9._kCRnEwOqI9K1JVCVbwofCqsZzzEm7PJBqfJCAG19bG"
}
```


Параметры запроса

- scope - список scope (в формате JSON Array) разрешенных для использования от имени пользователя, возможные значения задаются конфигурацией
- token_type - тип выданного токена. Всегда Bearer
- refresh_token - выданный refresh token
- refresh_expires_in - время до истечения срока действия refresh токена в секундах
- access_token - выданный access token
- expires_in - время до истечения срока действия токена в секундах
- old_token - токен, выданный внешней системой, строка
- JWTToken - JWT токен для длительного хранения и последующей аутентификации

Общие для всех шагов сообщения об ошибках

Код ошибки	Описание причины возникновения
invalid_credentials	Пользователь ввел неправильные данные учетной записи и не был аутентифицирован.
expired_password	Время действия пароля, введенного пользователем, истекло. Пользователь не был аутентифицирован.
user_blocked	Учетная запись пользователя заблокирована.
ip_blocked	IP адресс, с которого пользователь обращается в RooX UIDM, заблокирован.
invalid_captcha	Введенная пользователем CAPTCHA неверная.
need_captcha	Для продолжения сценария пользователь должен ввести CAPTCHA.
error	Пользователь не был аутентифицирован в результате неожиданного ответа от внешней системы.
login-by-otp-disabled	Пользователь попытался аутентифицироваться через сценарий с аутентификацией по OTP, но это невозможно, поскольку данный функционал отключен конфигурационным ключем.
error_sending_otp	Невозможно отправить OTP пользователю.
too_many_sms	Невозможно отправить OTP пользователю, поскольку превышено число допустимых повторных запросов отправки OTP. Примечание: OTP может быть отправлен пользователю произвольным способом, а не только через sms сообщение.
too_many_wrong_code	Превышено число допустимых попыток ввода OTP.
invalid_otp	Введенный OTP неверный.
validate-otp-fail	Введенный OTP неверный.
otp_expired	Время действия введенного OTP истекло.
otp-expired	Время действия введенного OTP истекло.
otp-error	Сценарий ввода OTP завершился неуспешно.
external-api-error	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.

external-system-bad-response	Выполнение сценария невозможно, поскольку получен неожиданный ответ от внешней системы или внешняя система не доступна.
msisdn-is-not-b2b	Введенный msisdn не принадлежит реалму b2b.
too_many_attempts	Превышено число допустимых попыток изменения пароля.
user-is-not-allowed	Данному пользователю запрещено продолжать данный сценарий.
error_password_change	Сценарий изменения пароля не был успешно завершен. Пароль не был изменен.
no_email_found	Пользователь не найден или у пользователя не существует email.
msisdn-not-exists	Пользователя с заданным msisdn не существует.
principal-not-exists	Пользователь не существует.
no-principal-found	Пользователь не существует.
user-not-found	Пользователь не существует.
login-exists	Невозможно зарегестрировать пользователя с заданным login, поскольку пользователь с заданным login уже существует.
login_already_exists	Невозможно изменить login пользователя на заданный, поскольку пользователь с заданным login уже существует.
email-exists	Пользователь с заданным email уже существует.
user-exists	Невозможно зарегестрировать пользователя с заданными msisdn или email или login, поскольку пользователь с такими параметрами уже существует.
email_verification_failed	Невозможно подтвердить email.
reset_required	Пароль или не был задан либо был сброшен в результате каких-то действий, необходимо создать новый пароль. Отличается от expired_password, поскольку в том случае пароль существует и был провалидирован, но требуется смена. В данном случае пароль не провалидирован, поскольку не задан.

Примеры ответов об ошибке

Не указан или передан пустой параметр <execution>

Формат ответа с ошибкой

```
HTTP/1.1 400 Bad Request
```

```
{
  "error": "invalid_grant",
  "error_description": "The provided access grant is invalid, expired, or revoked."
}
```

Не указан параметр <_eventId>

Формат ответа аналогичен ответу на первом шаге: Старт сценария

Указан неверный параметр <_eventId>

Формат ответа аналогичен ответу на первом шаге: Старт сценария, с заполненным списком ошибок в теле JSON ответа:

```
{
  "form": {
    "errors": [
      {
        "field": "username",
        "message": "may not be null"
      },
      {
        "field": "password",
        "message": "may not be null"
      }
    ]
  }
}
```