RingCentral for Developers BETA

RingCentral OAuth 2.0 Authentication & Authorization*

This pdf includes beta **3-legged OAuth 2.0 authorization code flow*

Contents

Overview 3

RingCentral OAuth 2.0 3

Tokens 3

Authorization Flows 4

Getting an Access Token 5

Authorization code flow [beta] 5

1. Request authorization code 5
2. User login and consent 6
3. Handling authorization code server response 8
4. Exchange code for token 9
5. Handling token server response 10

Resource Owner Password Credentials Flow 13

1. Request Access Token 14
2. Handling token server response 15

Client Authentication 17

Using access token to call RingCentral APIs 18

Refreshing access tokens 19

Revoking access tokens 20

Overview

RingCentral OAuth 2.0

Your application and its users must be authorized by RingCentral in order to eliminate any possibility of abuse. The RingCentral API uses *OAuth 2.0* protocol for authentication and authorization, which is widely supported by the majority of cloud API providers. For more details see OAuth 2.0 protocol specification.

In general, the steps your app needs to take to use RingCentral APIs (including authorization) look like this:

1. Get your app credentials from your Developer Portal account
2. Get an access token using one of the *authorization flows*
3. Use access token to call RingCentral APIs
4. Refresh your access token when necessary.

Note

In order to prevent eavesdropping and tampering, the RingCentral API requires Transport Layer Security. This means that API resources are accessible only through HTTPS connections.

Tokens

OAuth 2.0 protocol defines a number of tokens used to provide a context in each request for authorization or authentication. It is important to understand distinctions between token types:

- **Access token** is a special token issued by authorization server and used by the application to make requests to all endpoints which require authentication.
- **Refresh token** can be provided along with access token once your application successfully passes the authorization. It can be used only once to refresh short-lived access token. The refresh token itself cannot be used to access protected resources.

To prevent possible abuse by means of intercepting tokens and using them illegally, access and refresh token lifetimes are limited. For instance, by default access token is expired in one hour. Refresh token lifetime is typically limited to one week. Actual lifetimes of access and refresh tokens are returned in `expires_in` and `refresh_token_expires_in` attributes of a token endpoint response.

The API requests which include expired access tokens are rejected with `HTTP 401 Unauthorized` response. So an application is forced to obtain a new access token using a refresh token or by passing the authorization flow once again.

Both access and refresh tokens may also be revoked by the user at any time. In this case the application is required to pass the authorization flow again.

If the user who authorized the OAuth session changes his/her credentials (using RingCentral Service Web, Mobile Web or Admin Interface sites), all issued tokens are invalidated immediately, and all established sessions are terminated.

Authorization Flows

There are two main **authorization flows** you can use to get an authorized access to RingCentral API:

1. **Authorization code flow (beta)** is a 3-legged authorization flow and is a preferred flow for your app if it's a *web* application and involves logging in for *multiple users*;
2. **Resource Owner Password Credentials (ROPC)** flow is more suitable for *server apps* which will be used by a *single user*.

Both flows end up with your app obtaining an access token which you will need to call RingCentral APIs.

Flows for refreshing and revoking an access token are the same regardless of the authorization flow that was used for obtaining an access token.

Getting an Access Token

Authorization code flow beta

Authorization code flow protects users' information and lets them control what they share with your app. You are required to use this flow if your app is a web app and will be used by more than one user.

3-legged authorization flow used by RingCentral involves obtaining an *authorization code* from API server, which is exchanged for an access token later. The general flow looks like this:

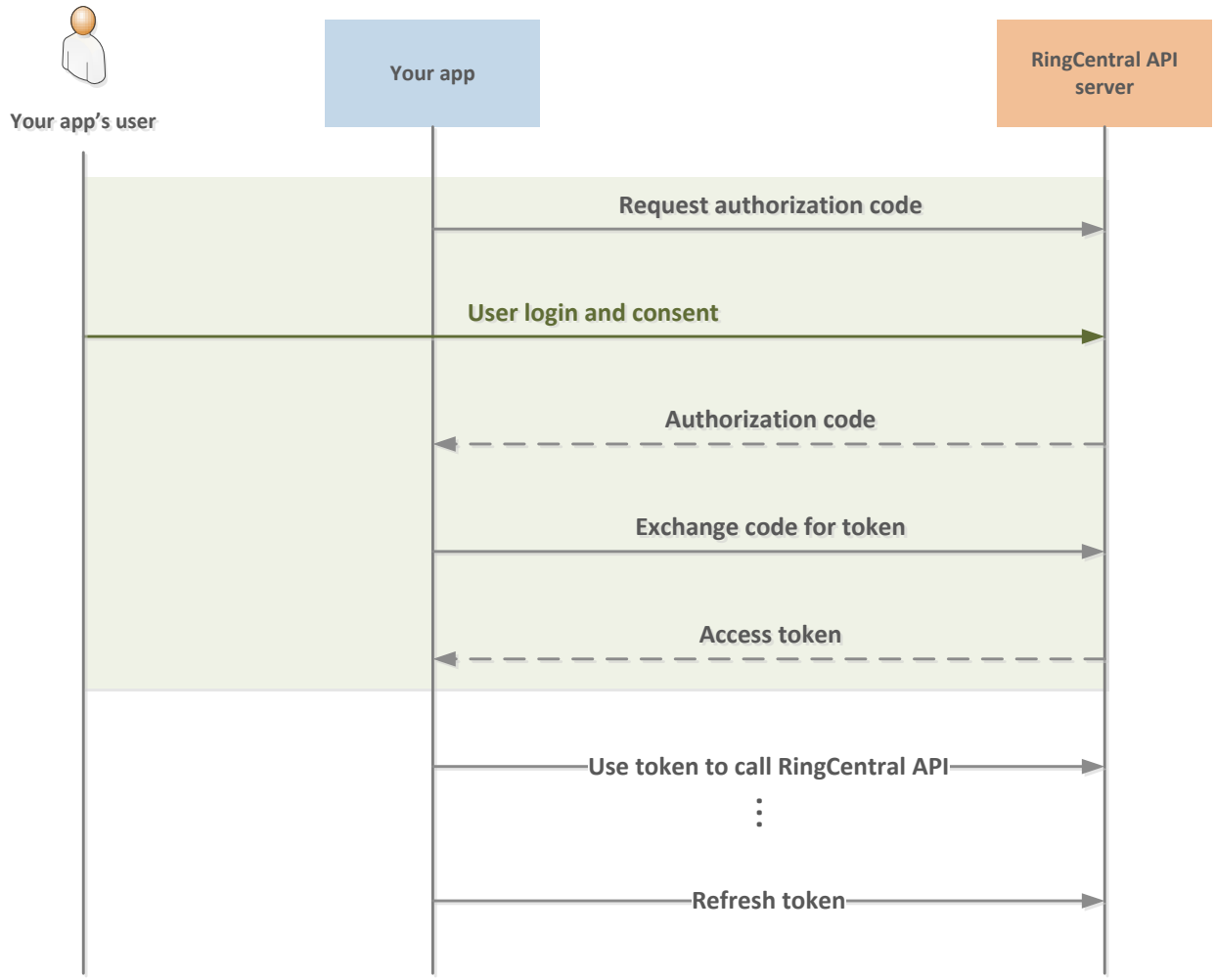

Fig. 1 Authorization Code Flow

The step-by-step details of this flow are explained below.

1. Request authorization code

When your application needs to access a user's data, redirect the user to RingCentral API server. Generate a URL to request access from endpoint **`restapi/oauth/authorize`**. This request must be in the `application/x-www-form-urlencoded` format by passing the following parameters in the HTTP request body:

Parameter	Type	Description
response_type	String	Required. Must be set to **code**.
client_id	String	Required. Enter your application key (Production or Sandbox) here.
redirect_uri	URI	This is a callback URI which determines where the response is sent. The value of this parameter must exactly match one of the URIs you have provided for your app upon registration.
prompt	String	This parameter determines your app user's experience when they go through 3-legged authorization flow. All external applications are required to set this parameter to **login consent**, which means that user will be prompted with a login page and a consent page (see Step 2 "User login and consent"). For private apps using Single Sign-On option, prompt should be set to **login consent sso** - in this case user will have a Single Sign-On option on his login page.
state	String	Recommended. An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter should be used for preventing cross-site request forgery.

2. User login and consent

On this step your app's user is redirected by the browser to a RingCentral authorization page, where user can view the list of permissions your app is asking for.

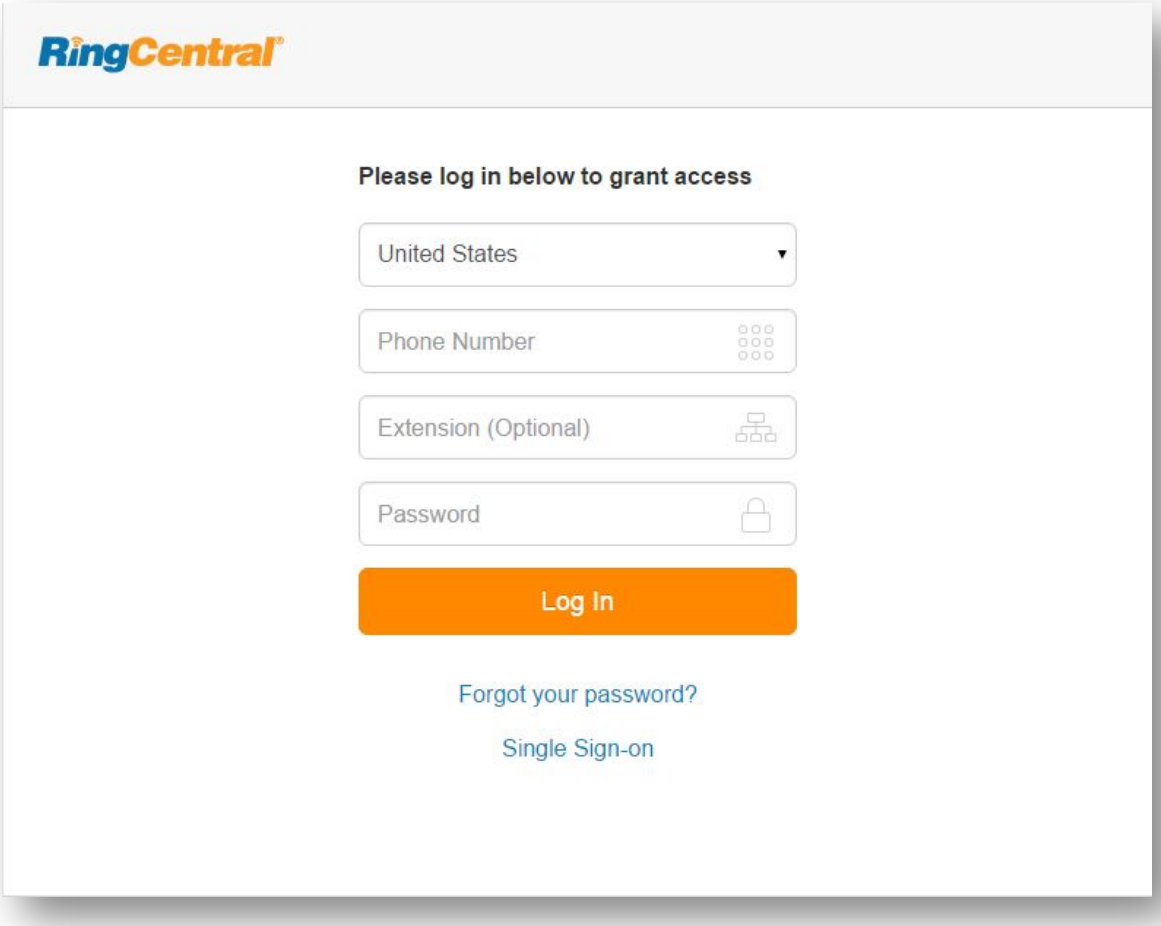

Fig. 2 User login page

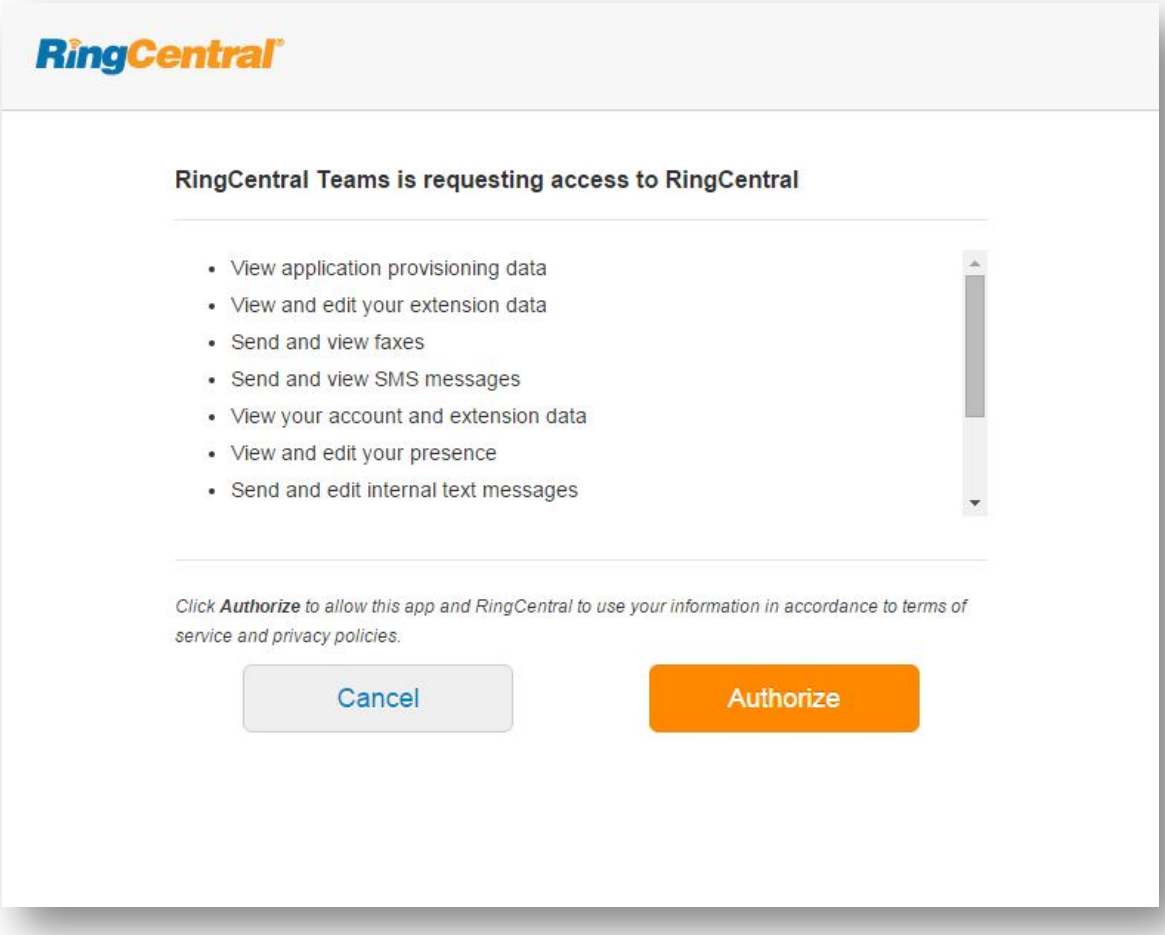

Fig. 3 User consent page

After confirming the permissions, user enters his/her RingCentral credentials, and the browser redirects back to the redirect URI you've provided in request.

3. Handling authorization code server response

The authorization server responds to your application's access request by using the URL specified in the request.

If the user approves the access request, then the response contains an authorization code. If the user does not approve the request, the response contains an error message.

An authorization code response contains:

Parameter	Type	Description
code	string	The authorization code returned for your application.

expires_in	integer	The remaining lifetime of the authorization code.
state	string	This parameter will be present in response if it was present in the client authorization request. The value will be copied from the one received from the client.

If authentication has been passed successfully, you will get a response similar to the following:

```
HTTP/1.1 302 Found

Location:
com.ringcentral.rcmobile:/oauth2Callback?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz&expires_in=60
```

4. Exchange code for token

After the web server receives the authorization code, it can exchange the authorization code for an access token using token endpoint **`/restapi/oauth/token`** (usage plan group is *Auth*).

Token requests must include client authentication (see Client Authentication section).

Request Body

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

Parameter	Type	Description
grant_type	string	Must be set to code for authorization code flow
code	string	Provide your authorization code received in the previous step
redirect_uri	URI	This is a callback URI which determines where the response is sent. The value of this parameter must exactly match one of the URIs you have provided for your app upon registration.
access_token_ttl	integer	Optional. Access token lifetime in seconds; the possible values are from 600 sec (10 min) to 3600 sec (1 hour). The default value is 3600 sec. If the value specified exceeds the default one, the default value is set. If the value specified is less than 600 seconds, the minimum

		value (600 sec) is set
refresh_token_ttl	integer	Optional. Refresh token lifetime in seconds. The default value depends on the client application, but as usual it equals to 7 days. If the value specified exceeds the default one, the default value is applied.
scope	string	Optional. List of API permissions to be used with access token (see Application Permissions). Can be omitted when requesting all permissions defined during the application registration phase
endpoint_id	string	Optional. Unique identifier of a client application generated by the client and valid during the application lifetime

5. Handling token server response

Response Body

Content Type: `application/json, application/xml`

Parameter	Type	Description
access_token	string	Access token to pass to subsequent API requests
expires_in	integer	Issued access token TTL (time to live), in seconds
refresh_token	string	Refresh token to get a new access token, when the issued one expires
refresh_token_expires_in	integer	Issued refresh token TTL (time to live), in seconds
scope	string	List of permissions allowed with this access token, white-space separated
token_type	string	Type of token. Use this parameter in

		Authorization header of requests
owner_id	string	Extension identifier
endpoint_id	string	Optional. Unique identifier of a client application generated by the client and valid during the application lifetime

Example 1

Request example

```
POST /restapi/oauth/token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic
cmVsLWFsbC1wZXJtaXNzaWXFjMmpRZmlQcnlkSUkweE92QQ==

grant_type=password&username=18559100010&extension=101&password
=121212
```

Response example

```
HTTP/1.1 200 OK
Content-Type: application/json

{
"access_token" :
"U1BCMDFUMDRKV1MwMXxzLFSvXdw5PHMsVLEn_MrtcyxUsw",
"token_type" : "bearer",
"expires_in" : 7199,
"refresh_token" :
"U1BCMDFUMDRKV1MwMXxzLFL4ec6A0XMsUv9wLriecyxS_w",
"refresh_token_expires_in" : 604799,
"scope" : "AccountInfo CallLog ExtensionInfo Messages SMS",
"owner_id" : "256440016"
}
```

Example 2

If the application brand ID does not match the account brand ID, then the `OAU-101` error is returned with the following message: "Parameter [brandId] is invalid" and HTTP status code `403 Forbidden`.

Request example

```
POST /restapi/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic WW91hguitkcEtleTpZb3VyQXBwU2VjcmV0
Accept: application/json
               access_token_ttl=7200&grant_type=client_credenti
als&brand_id=654321
```

Response example

```
{
"error":  "invalid_client",
"error_description":  "Access to account denied"
"errors" : [ {
"errorCode" :  "OAU-101",
"message" :  "Parameter [brandId] is invalid",
"parameters": [ {
     "parameterName" :  "brandId",
     "parameterValue" :  "123456" //account brand ID
     } ]
} ]
}
```

Resource Owner Password Credentials Flow

Resource Owner Password Credentials, or ROPC, is the simplest OAuth 2.0 authorization flow. It is suitable mostly for server apps which will be used by a single user. Typically the user enters credentials in the form which is provided by the application itself (instead of being redirected to the RingCentral website to enter credentials through Web Browser).

Please note that this flow is considered to be less secure and requires an additional level of trust between you and the application.

This authorization flow uses Resource Owner Password Credentials OAuth grant type.

Two steps are required for this flow:

1. The application by itself obtains user credentials from the user.
2. The application supplies user credentials and application credentials in a request to a token endpoint. Once the credentials are successfully verified, the application receives the access token and the refresh token in HTTP response.

Resource Owner Password Credentials flow used by RingCentral results in obtaining an *access token* from API server. The general ROPC flow looks like this:

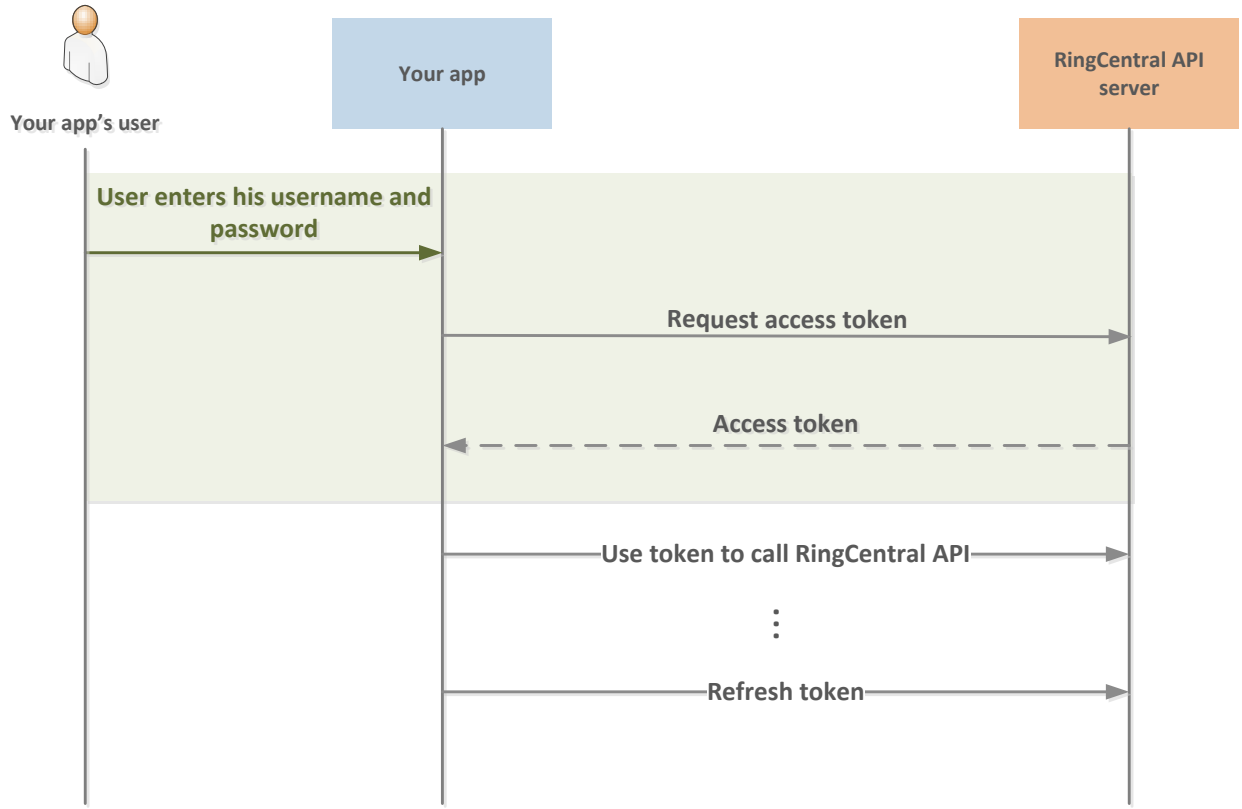

Fig. 4 Resource Owner Password Credentials Flow

Below find the step-by-step instructions on how to perform two-legged authorization using the RingCentral API.

1. Request Access Token

You have to implement a way of obtaining user credentials from the users of your application.

Once your application has obtained credentials from the user, it can send a specific request to token endpoint **/restapi/oauth/token** (usage plan group is *Auth*).

Token requests must include client authentication (see Client Authentication section).

Request Body

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

Parameter	Type	Description
grant_type	string	Must be set to **password** for Resource Owner Credentials flow
access_token_ttl	integer	Optional. Access token lifetime in seconds; the possible values are from 600 sec (10 min) to 3600 sec (1 hour). The default value is 3600 sec. If the value specified exceeds the default one, the default value is set. If the value specified is less than 600 seconds, the minimum value (600 sec) is set
refresh_token_ttl	integer	Optional. Refresh token lifetime in seconds. The default value depends on the client application, but as usual it equals to 7 days. If the value specified exceeds the default one, the default value is applied.
username	string	Phone number linked to account or extension in account in E.164 format with or without leading "+" sign
extension	string	Optional. Extension short number. If company number is specified as a username, and extension is not specified, the server will attempt to authenticate client as main company administrator
password	string	Required. User's password.
scope	string	Optional. List of API permissions to be used with access token (see Application Permissions). Can be omitted when requesting all permissions defined during the application registration phase
endpoint_id	string	Optional. Unique identifier of a client application generated by the client and valid during the application

lifetime

2. Handling token server response

Response Body

Content Type: `application/json, application/xml`

Parameter	Type	Description
access_token	string	Access token to pass to subsequent API requests
expires_in	integer	Issued access token TTL (time to live), in seconds
refresh_token	string	Refresh token to get a new access token, when the issued one expires
refresh_token_expires_in	integer	Issued refresh token TTL (time to live), in seconds
scope	string	List of permissions allowed with this access token, white-space separated
token_type	string	Type of token. Use this parameter in Authorization header of requests
owner_id	string	Extension identifier
endpoint_id	string	Optional. Unique identifier of a client application generated by the client and valid during the application lifetime

Example 1

Request example

```
POST /restapi/oauth/token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
```

```
Authorization: Basic
cmVsLWFsbC1wZXJtaXNzaWXFjMmpRZmlQcnlkSUkweE92QQ==

grant_type=password&username=18559100010&extension=101&password
=121212
```

Response example

```
HTTP/1.1 200 OK
Content-Type: application/json

{
"access_token" :
"U1BCMDFUMDRKV1MwMXxzLFSvXdw5PHMsVLEn_MrtcyxUsw",
"token_type" : "bearer",
"expires_in" : 7199,
"refresh_token" :
"U1BCMDFUMDRKV1MwMXxzLFL4ec6A0XMsUv9wLriecyxS_w",
"refresh_token_expires_in" : 604799,
"scope" : "AccountInfo CallLog ExtensionInfo Messages SMS",
"owner_id" : "256440016"
}
```

Example 2

If the application brand ID does not match the account brand ID, then the `OAU-101` error is returned with the following message: "Parameter [brandId] is invalid" and HTTP status code `403 Forbidden`.

Request example

```
POST /restapi/oauth/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic WW91hguitkcEtleTpZb3VyQXBwU2VjcmV0
Accept: application/json

access_token_ttl=7200&grant_type=client_credentials&brand_id=65
4321
```

Response example

```
{
"error":  "invalid_client",
"error_description":  "Access to account denied"
"errors" : [ {
"errorCode" :  "OAU-101",
```

```
"message" :  "Parameter [brandId] is invalid",
"parameters": [ {
"parameterName" :  "brandId",
"parameterValue" :  "123456" //account brand ID
} ]
} ]
}
```

Client Authentication

Each application (client) that intends to obtain an access token must be authenticated. To authenticate the application we use **application key** and **application secret** issued during application registration. They are passed to the token endpoint as username and password using the HTTP Basic authentication scheme.

For example, you have obtained application key `YourAppKey` and application `secretYourAppSecret`. Combine them in a string with a colon `YourAppKey:YourAppSecret` and encode with Base64; thus you will get the following authorization token `WW91ckFwcEtleTpZb3VyQXBwU2VjcmV0`. Put this value into your token request as shown in example below (the example represents ROPC flow):

```
POST /restapi/oauth/token HTTP/1.1
Host: platform.ringcentral.com
Authorization: Basic WW91ckFwcEtleTpZb3VyQXBwU2VjcmV0
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
grant_type=password&username=18887776655&extension=102&password
=987654321
```

Example values are:

- `platform.ringcentral.com` - name of the RingCentral API server
- `WW91ckFwcEtleTpZb3VyQXBwU2VjcmV0` - Base64 encoded HTTP Basic string generated from application credentials (application key and secret)
- `18887776655` - RingCentral customer login (phone number)
- `102` - particular extension number
- `987654321` - password to log in as the extension 102 of the account 18887776655

Client authentication uses the same principles for both ROPC and Authorization Code flows.

Using access token to call RingCentral APIs

Now your application should use the issued access token to perform the required actions. Each request must pass the access token using one of the following ways:

- `access_token` query parameter with the issued access token specified as a value:

  For example, to get a specific address-book entry, you need to perform the following request:

```
GET
/restapi/v1.0/account/1110475004/extension/1110475004/address-
book/contact/29874662828?access_token=2YotnFZFEjr1zCsicMWpAA
Host: platform.ringcentral.com
Accept: application/json
Connection: keep-alive
```

- `Bearer` authentication scheme followed by access token in the `Authorization` header.

  For example, to get a specific address-book entry, you need to perform the following request:

```
GET
/restapi/v1.0/account/1110475004/extension/1110475004/address-
book/contact/29874662828
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
Host: platform.ringcentral.com
Accept: application/json
Connection: keep-alive
```

Refreshing access tokens

Refreshing access tokens is described in corresponding Developer Guide section and is the same regardless of the flow used for obtaining the access token.

Revoking access tokens

Revoking access tokens is described in corresponding Developer Guide section and is the same regardless of the flow used for obtaining the access token.