Customer Identity Access Management Development Guide

Contents

Development Guide

Overview

Last updated：2023-09-04 10:39:49

This guide outlines the methods and corresponding APIs for integrating your application system with CIAM. By interfacing with CIAM, your application can swiftly implement user login, logout, registration, and other functions, while also integrating the flexible and robust configuration and management capabilities of CIAM.

Type

Based on common business scenarios, CIAM categorizes applications into the following types:

Web Application

Applications running on backend web servers, typically developed using languages or frameworks such as Java, .NET, PHP, Node.js, Express, and accessed by users via a browser. As backend programs generally run on protected servers, web applications can effectively store sensitive information such as application passwords, and safeguard dynamic information like tokens from being leaked.

Single-Page Application (SPA)

Frontend applications that run directly in the browser, typically developed using HTML, CSS, and JavaScript technologies in conjunction with frameworks such as React, Vue, and Angular. Single-page applications can directly initiate requests to business-facing APIs without the need for backend program forwarding. However, as both the program and data reside in the user-agent (such as a browser), single-page applications are not suitable for storing or processing protected sensitive information.

Mobile App

Applications installed and run on user devices such as smartphones, tablets, PCs, and smart devices, typically developed using specialized application development languages like Objective-C, Swift, Kotlin, etc. These types of applications are not suitable for storing sensitive information like application passwords, but they can generally protect dynamic information like tokens from being leaked.

M2M Application (Machine to Machine)

Applications running on the backend (such as backend services, command-line programs, daemons), typically implement business functions by calling other APIs, without the need for user involvement. These types of applications can effectively prevent the leakage of sensitive information.

Mini Program Application

WeChat Mini Program Application, operating within the WeChat App on mobile or desktop devices.

Integration Overview

Web applications, SPAs, and mobile apps can be quickly integrated with CIAM as standard OIDC (OAuth) clients, as detailed in Logging in Using the Authentication Portal. The OIDC and OAuth protocols are widely used on the internet, and their corresponding development resources are abundant. For all types of applications, development libraries can generally be found to expedite integration.

- WeChat Mini Program applications need to use the CIAM Mini Program SDK to implement login. For detailed operations, please refer to Integrating WeChat Mini Program (SDK Mode).
- M2M applications typically use the client credentials mode to obtain an Access Token, and then carry this Access Token to access the corresponding APIs.

Note
The APIs in this guide are accessed via the HTTPS protocol, with the prefix of the access address being your user directory domain, which can be viewed on the Domain Settings page. In this guide, `https://sample.portal.tencentciam.com` is used as the prefix for the access address.

Access via Authentication Portal
Login via Authentication Portal
PKCE Authorization Code Mode

Last updated: 2023-09-04 14:05:47

API Description

Redirect the user (browser) to this API address to initiate login. CIAM will redirect the user to the authentication portal for login verification. Upon successful login, CIAM will redirect the user to the address specified by the `redirect_uri` parameter.
If the user is already logged in when initiating login, there is no need to log in again. CIAM will directly redirect the user to the `redirect_uri`.

Note

- In accordance with the security best practices of the OAuth protocol, this interface uses the PKCE authorization code mode.
- The application system Redirect URI used in this section's request example is `https://example.com/callback`.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
GET
```

Request path

```
/oauth2/authorize
```

Sample Request

```
GET /oauth2/authorize?scope=openid&client_id=TENANT_CLIENT_ID&redirect_uri=https%
Host: sample.portal.tencentciam.com
```

Request Parameters

Category	Optional	Description
scope	false	Enter the fixed value `openid`.
client_id	false	The application's `client_id`. Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "Client Id".
redirect_uri	false	The redirect address after successful authorization. It needs to be consistent with the address configured in the console.
response_type	false	Enter the fixed value `code`.
state	true	An application randomly generates a string, which the server will return to the application in the form of an HTTP response parameter. To prevent CSRF attacks, it is recommended to carry this parameter.
code_challenge_method	false	The algorithm for calculating the PKCE code_challenge. Currently, only the fixed value `S256` is supported.
code_challenge	false	PKCE code_challenge, for calculation method please refer to RFC 7636.
auth_source_id	true	Log in using a specific authentication source. If this parameter is empty, the default login page will be displayed.

Example of a successful response

- If the user is not logged in, the default login page of the authentication portal is displayed.

```
HTTP/1.1 302 Found
Location: https://sample.portal.tencentciam.com/portal/login?p_state=MOCK_LOGIN_P
```

- The user is logged in and redirected to the application callback address with the authorization code and state parameter.

```
HTTP/1.1 302 Found
Location: https://example.com/callback?code=DVtNBg5XGqeu2IytLi6WOWwfh7pRc5jc
```

Note

After the application callback address obtains the code parameter, it needs to call the PKCE mode Get Token interface to obtain the Access Token and ID Token, completing the login.

Exception response sample

- The client_id parameter is missing or incorrect.

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

- The redirect_uri parameter does not match the registration information.

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

- The response_type parameter is missing or incorrect.

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

- Unsupported code_challenge_method.

```
HTTP/1.1 302 Found
Location: https://example.com/callback?error=invalid_request&error_description=OAu
```

General Authorization Code Mode

Last updated: 2023-09-04 14:06:36

API Description

Redirect the user (browser) to this API address to initiate login. CIAM will redirect the user to the authentication portal for login verification. Upon successful login, CIAM will redirect the user to the address specified by the `redirect_uri` parameter.
If the user is already logged in when initiating login, there is no need to log in again. CIAM will directly redirect the user to the `redirect_uri`.

Note

- For security reasons, the OAuth protocol does not recommend using this mode. Please prioritize using the PKCE Authorization Code Mode.
- The application system Redirect URI used in this section's request example is `https://example.com/callback`.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
GET
```

Request path

```
/oauth2/authorize
```

Sample Request

```
GET /oauth2/authorize?scope=openid&client_id=TENANT_CLIENT_ID&redirect_uri=https%
Host: sample.portal.tencentciam.com
```

Request Parameters

Category	Optional	Description

scope	false	Enter the fixed value 'openid'.
client_id	false	The application's client_id. Refer to Application Management Page > Select the specified application > Click on Application Configuration > Corresponding "Client Id".
redirect_uri	false	The redirect address after successful authorization. It needs to be consistent with the address configured in the console.
response_type	false	Enter the fixed value 'code'.
state	true	An application randomly generates a string, which the server will return to the application in the form of an HTTP response parameter. To prevent CSRF attacks, it is recommended to carry this parameter.
auth_source_id	true	Log in using a specific authentication source. If this parameter is empty, the default login page will be displayed.

Example of a successful response

- If the user is not logged in, the default login page of the authentication portal is displayed.

```
HTTP/1.1 302 Found
Location: https://sample.portal.tencentciam.com/portal/login?p_state=MOCK_LOGIN_P
```

- The user is logged in and redirected to the application callback address with the authorization code and state parameter.

```
HTTP/1.1 302 Found
Location: https://example.com/callback?code=3ZBaZRXqXZxdkQ1-qwj-YLsWOhYJvQGz
```

> **Note**
> After the application callback address receives the `code` parameter, it needs to call the Standard Authorization Code Mode to Get Token API to obtain the Access Token and ID Token, thereby completing the login process.

Exception response sample

- The client_id parameter is missing or incorrect.

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

- The redirect_uri parameter does not match the registration information.

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

- The response_type parameter is missing or incorrect.

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

Logging out of the authentication portal

Last updated：2023-09-04 10:40:09

API Description

This API redirects the user (browser) to its interface address, logs out, and clears the authentication portal login status. After logging out, CIAM will redirect the user to the application's logout callback address `logout_redirect_uri` .

Note

- This API clears the login status of the authentication portal. The login status of the user in the application system needs to be cleared by the application itself.
- In the request example of this section, the application system's Logout Redirect URI used is `https://example.com/logout` .

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
GET
```

Request path

```
/logout
```

Sample Request

```
GET /logout?client_id=TENANT_CLIENT_ID&logout_redirect_uri=https%3A%2F%2Fexampl
Host: sample.portal.tencentciam.com
```

Request Parameters

Category	Optional	Description

client_id	false	The application's `client_id` . Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "Client Id".
logout_redirect_uri	true	The redirect address after logging out needs to match the `Logout Redirect URI` in the application parameter configuration. If the application has only configured one `Logout Redirect URI` , this parameter can be omitted, and the system will use this configuration as the default redirect address after logging out.
state	true	An arbitrarily generated string by the application, which will be returned as is during redirection.

Example of a successful response

```
HTTP/1.1 302 Found
Location: https://example.com/logout
```

Exception response sample

The client_id parameter is missing.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
 "error" : "invalid_request",
 "error_description" : "Client ID parameter not found"
}
```

Register via the Authentication Portal

Last updated：2023-09-04 10:41:08

API Description

Redirect the user (browser) to this API address to initiate registration. If the application has enabled "Auto-login after registration", the user will be automatically redirected to the application callback address upon successful registration, following the same logic as Using the Authentication Portal for Login. If "Auto-login after registration" is not enabled, the user will be redirected to the CIAM Authentication Portal login page after successful registration.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
GET
```

Request path

```
/oauth2/authorize
```

Sample Request

```
GET /oauth2/authorize?scope=openid&client_id=TENANT_CLIENT_ID&redirect_uri=https%
Host: sample.portal.tencentciam.com
```

Request Parameters

Category	Optional	Description
scope	false	Enter the fixed value `openid`.
client_id	false	The application's `client_id`. Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "Client Id".

redirect_uri	false	The redirect address after successful authorization. It must be consistent with the address configured in the Tenant Management Platform.
response_type	false	Enter the fixed value `code`.
state	true	An application randomly generates a string, which the server will return to the application in the form of an HTTP response parameter. To prevent CSRF attacks, it is recommended to carry this parameter.
code_challenge_method	false	The algorithm for calculating the PKCE code_challenge. Currently, only the fixed value `S256` is supported.
code_challenge	false	PKCE code_challenge。
prompt	true	When this parameter exists and its value is `create`, it represents the initiation of a registration request. In all other cases, it will be interpreted as initiating a login request.

Example of a successful response

Redirect to the Authentication Portal's user registration page.

```
HTTP/1.1 302 Found
Location: https://sample.portal.tencentciam.com/portal/signup?p_state=MOCK_LOGIN_PC
```

Exception response sample

- The client_id parameter is missing or incorrect.

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

- The redirect_uri parameter does not match the registration information.

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

- The response_type parameter is missing or incorrect.

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

- **Unsupported code_challenge_method.**

```
HTTP/1.1 302 Found
Location: https://example.com/callback?error=invalid_request&error_description=OAu
```

Access via Authentication API
User Sign-up

Last updated：2023-09-04 10:41:17

API Description

Register a new user. This API is suitable for scenarios where **the application system develops its own registration function.** If your application uses the CIAM authentication portal, please refer to Registering via the Authentication Portal.

Before invoking this API, ensure that the application's registration process has been configured and enabled. The API parameters must comply with the business rules configured in the registration process. For instance, if the registration process is configured with a phone number as an authentication attribute and the user nickname as a required general attribute, then the parameters must include both the phone number and user nickname; **if the registration process is not configured with a phone number as an authentication attribute, then the parameters cannot include a phone number.**

When the registration information includes a phone number or email address, you must first invoke the Send OTP Verification Code API to send a verification code to the user.

The password is an optional parameter. You can decide whether to require users to set a password based on specific business circumstances.

- If users only log in via SMS OTP, email OTP, or social authentication methods, they do not need to set a password.
- If you need to support user login through username-password authentication, a password should be set.

If a password needs to be set, ensure that the **account-password authentication source is associated** in the application's login process. The API will validate the incoming password based on the password policy of this authentication source. If the password does not meet the policy requirements, the registration cannot be successfully completed.

Note
- This API does not support setting user groups. Users who successfully register will default to the user group configured in the registration process.
- This API does not handle automatic login and identity verification logic. That is, the automatic login and identity verification configurations in the registration process

do not affect this API.

Supported Application Types

Web Application.

Request method

```
POST
```

Request path

```
/signup
```

Request Content-Type

```
application/json
```

Sample Request

- Register using a username and set a password.

```
POST /signup HTTP/1.1
Content-Type: application/json
Authorization: Basic VEVOQU5UX0NMSUVOVF9JRDpURU5BTlRfQ0xJRU5UX1NFQ1JFVA=
Host: sample.portal.tencentciam.com

{
"username" : "MOCK_USERNAME",
"password" : "MOCK_PASSWORD"
}
```

- Register using an email address and nickname, and set a password.

```
POST /signup HTTP/1.1
Content-Type: application/json
Authorization: Basic VEVOQU5UX0NMSUVOVF9JRDpURU5BTlRfQ0xJRU5UX1NFQ1JFVA=
Host: sample.portal.tencentciam.com

{
"email" : "MOCK_USERNAME@example.com",
"email_otp_token" : "MOCK_EMAIL_OTP_TOKEN",
"email_otp" : "MOCK_EMAIL_OTP",
"password" : "MOCK_PASSWORD",
"nickname" : "MOCK_NICKNAME"
}
```

- Register using a phone number, without setting a password.

```
POST /signup HTTP/1.1
Content-Type: application/json
Authorization: Basic VEVOQU5UX0NMSUVOVF9JRDpURU5BTlRfQ0xJRU5UX1NFQ1JFVA=
Host: sample.portal.tencentciam.com

{
"phone_number" : "13612345678",
"phone_number_otp_token" : "MOCK_PHONE_NUMBER_OTP_TOKEN",
"phone_number_otp" : "MOCK_PHONE_NUMBER_OTP"
}
```

Request Header

Name	Description
Authorization	The HTTP Basic authentication request header is in the format of `Basic <credentials>` , where `Basic` is a fixed string, and `<credentials>` is calculated as `base64(url_encode(client_id) + ":" + url_encode(client_secret))` . There is a space between `Basic` and `<credentials>` .

Request Body JSON Parameters

JSON Path	Data Type	Description
username	String	Username: It can contain English letters, numbers, and underscores, must start with a letter, and can be up to 32 characters long.
password	String	User password. If set, it must comply with the password policy of the account password authentication source associated with the application.
phone_number	String	The user's mobile number, limited to the 11-digit mobile numbers of the three major domestic operators. When passing this parameter, you must also pass the

		`phone_number_otp_token` and `phone_number_otp` parameters simultaneously.
phone_number_otp_token	String	The `otp_token` returned by the server after successfully sending the SMS verification code.
phone_number_otp	String	The OTP verification code received on the user's mobile phone.
email	String	The user's email address. When passing this parameter, you must also pass the `email_otp_token` and `email_otp` parameters simultaneously.
email_otp_token	String	The `otp_token` returned by the server after successfully sending the email verification code.
email_otp	String	The OTP verification code received by the user's email.
name	String	User Name.
nickname	String	User Nickname.
zoneinfo	String	User's time zone, such as `Asia/Shanghai` or `Europe/Paris` .
locale	String	User locale information, such as `zh-CN` or `en-US` .

Note
The value of other parameters is the user attribute identifier. The attribute identifier can be viewed on the attribute details interface of the Custom Attribute Page .

Example of Successful Registration Response

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

{
  "sub" : "MOCK_USER_ID"
}
```

Response Parameters

Parameter	Data Type	Description
sub	String	Unique user identifier.

Example of a Failed Registration Response

- The application registration process is not enabled.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "Sign up flow of the application is not enabled."
}
```

- The input parameters are missing authentication attributes or required general attributes configured in the registration process.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Missing required sign-up attribute(s)."
}
```

- The input parameters include authentication attributes or general attributes that are not configured in the registration process.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Unconfigured sign-up attribute(s) found."
}
```

- The input includes an unknown attribute.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Unknown attribute(s) found."
}
```

- Invalid username format.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_username"
}
```

- This username already exists.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "duplicate_username"
}
```

- The format of the phone number is invalid.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "malformed_phone_number"
}
```

- The phone number already exists.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
```

```
"error" : "duplicate_phone_number"
}
```

- The `phone_number_otp_token` is incorrect, has expired, or the parameters used during registration are inconsistent with those used when sending the verification code (for example, the phone numbers are different).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_phone_number_otp_token"
}
```

- The `phone_number_otp` is incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_phone_number_otp"
}
```

- The email format is invalid.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "malformed_email"
}
```

- This email address already exists.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "duplicate_email"
}
```

- The `email_otp_token` is incorrect, has expired, or the parameters used during registration are inconsistent with those used when sending the verification code (for example, different email addresses).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp_token"
}
```

- The `email_otp` is incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp"
}
```

- The password is included in the parameters, but the account-password authentication source is not associated in the application login process.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "No password auth source is associated with the application."
}
```

- The password does not meet policy requirements.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_password"
}
```

Account and Password Authentication

Last updated：2023-09-04 10:41:23

API Description

Validate the user's username and password to obtain the Access Token and ID Token, thereby completing the login process. This interface corresponds to the Resource Owner Password Credentials mode of the OAuth 2.0 protocol.

Note
As the user's password will be transmitted between the user terminal and the application, it is imperative to invoke this interface with a highly trusted application and handle the password transmission appropriately (for instance, ensuring the use of the HTTPS protocol). Where conditions permit, it is recommended to prioritize using Authentication Portal Login.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Host: sample.portal.tencentciam.com
```

```
Content-Type: application/x-www-form-urlencoded
grant_type=password&client_id=TENANT_CLIENT_ID&client_secret=TENANT_CLIENT_SEC
```

Request Parameters

Category	Optional	Description
grant_type	false	Enter the fixed value `password` .
client_id	false	The application's `client_id` . Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "Client Id".
client_secret	true	The application's `client_secret` . Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "client_secret". • This parameter must be passed for Web Applications. • This parameter is not transmitted for single-page applications and mobile apps.
auth_source_id	false	Username-password authentication source ID. This can be viewed on the General Authentication Sources page in the console.
username	false	Username. It is necessary to use the authentication source attributes configured with the username-password authentication source, such as username, phone number, or email address.
password	false	User Password.
scope	true	This can be omitted. If passed, fill in the fixed value `openid` .

Example of a successful response

Authentication succeeded

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
 "access_token" : "eyJraWQiOiI1MzQyOGU3ZS1kOTJiLTQ3OTAtOGIwMC0wMmEyZjc4NjUxI
```

```
  "refresh_token" : "7uqTlthTQrzIZx8joT20chQbakZp81_iv39GTyCpsEyYpWoquNhuB3s6qE
  "scope" : "openid",
  "id_token" : "eyJraWQiOiI1MzQyOGU3ZS1kOTJiLTQ3OTAtOGIwMC0wMmEyZjc4NjUxNzMiI
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	OAuth 2.0 Access Token (JWT)。
token_type	String	Token type, currently, the returned fixed value is `Bearer`.
expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Access Token scope。
refresh_token	String	OAuth 2.0 Refresh Token。
id_token	String	OIDC ID Token (JWT)。

Exception response sample

- Incorrect username or password.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Wrong username or password"
}
```

- The user status is abnormal (such as being locked or frozen).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_grant",
"error_description" : "Abnormal user status"
}
```

- An attribute unsupported by the authentication source is used as the username (for example, an email address is passed as the username, but the account−password authentication source has not configured the email address as an authentication source attribute).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Unsupported username identifier"
}
```

- The authentication source is neither the preferred nor the associated authentication source for the application.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Auth source and application not associated"
}
```

OTP Authentication by SMS and Email

Last updated：2023-09-04 14:08:14

API Description

Verify the OTP code sent via SMS or email to obtain the Access Token and ID Token, thereby completing the login process. Before invoking this API, you need to send the verification code to the user via the Send OTP Code API.

Note
You can support automatic user registration by passing the `auto_signup=true` parameter.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/json
```

Sample Request

SMS OTP Login

```
POST /oauth2/token HTTP/1.1
Content-Type: application/json
Host: sample.portal.tencentciam.com
```

```
{
  "grant_type" : "http://tencentciam.com/oauth2/grant-type/otp/sms",
  "client_id" : "TENANT_CLIENT_ID",
  "client_secret" : "TENANT_CLIENT_SECRET",
  "auth_source_id" : "MOCK_SMS_OTP_AUTH_SOURCE_ID",
  "phone_number" : "13612345678",
  "otp_token" : "MOCK_OTP_TOKEN",
  "otp" : "123456"
}
```

Email OTP Login

```
POST /oauth2/token HTTP/1.1
Content-Type: application/json
Host: sample.portal.tencentciam.com

{
  "grant_type" : "http://tencentciam.com/oauth2/grant-type/otp/email",
  "client_id" : "TENANT_CLIENT_ID",
  "client_secret" : "TENANT_CLIENT_SECRET",
  "auth_source_id" : "MOCK_EMAIL_OTP_AUTH_SOURCE_ID",
  "email" : "MOCK_USERNAME@example.com",
  "otp_token" : "MOCK_EMAIL_OTP_TOKEN",
  "otp" : "123456"
}
```

Request Body JSON Parameters

JSON Path	Data Type	Description
grant_type	String	• SMS OTP Login Input: `http://tencentciam.com/oauth2/grant-type/otp/sms` • Email OTP Login Input: `http://tencentciam.com/oauth2/grant-type/otp/email`
client_id	String	The `client_id` of the application. It must be consistent with the one used when sending the verification code.
client_secret	String	The `client_secret` of the application. This parameter must be passed for Web applications. It is not required for Single Page Applications and Mobile Apps.

auth_source_id	String	The ID of the SMS OTP or email OTP authentication source. It needs to be consistent with the one used when sending the verification code.
phone_number	String	The user's mobile number. It needs to be consistent with the one used when sending the verification code. Pass this parameter when logging in with SMS OTP.
email	String	The user's email address. It needs to be consistent with the one used when sending the verification code. Pass this parameter when logging in via email OTP.
otp_token	String	The `otp_token` returned by the server after successfully sending the verification code.
otp	String	The OTP code received on the user's mobile phone or email.
auto_signup	Boolean	If you need to support automatic user registration, pass this parameter as true. Otherwise, it is not necessary to pass it.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiJmZTQ4YTJjYS1INGU3LTQyMGEtOThjOS01OGM5NmI2NzUw
  "refresh_token" : "B-72VIkQa3jQNuo9Xbbl-muoh4w7nYu-7Q3Wb-qmPgyftN1CgXPov2aW
  "scope" : "openid",
  "id_token" : "eyJraWQiOiJmZTQ4YTJjYS1INGU3LTQyMGEtOThjOS01OGM5NmI2NzUwZjIiL(
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Parameter	Data Type	Description
access_token	String	OAuth 2.0 Access Token (JWT)。
token_type	String	Token type, currently, the returned fixed value is `Bearer`.

expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Access Token scope.
refresh_token	String	OAuth 2.0 Refresh Token.
id_token	String	OIDC ID Token (JWT).

Exception response sample

- The otp_token is erroneous or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Unknown or expired otp_token"
}
```

- The OTP is incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Unknown or expired OTP"
}
```

- The parameters used are inconsistent with those used when sending the verification code (for example, different mobile numbers).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Mismatched OTP token and OTP sending parameters"
}
```

- The user corresponding to the phone number or email cannot be found (in cases where automatic user registration is not permitted).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "User not found"
}
```

- The status of the user associated with the mobile number or email is abnormal (such as being locked or frozen).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Abnormal user status"
}
```

- The authentication source is neither the preferred nor the associated authentication source for the application.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Auth source and application not associated"
}
```

WeChat Authorization Login

Last updated: 2023-09-04 10:41:35

API Description

After a website, WeChat Official Account, or mobile app uses WeChat login to obtain the user's temporary authorization ticket (code), this API is called to validate the code, obtain the CIAM's Access Token and ID Token, and complete the login process.
CIAM will associate the local user with WeChat's unionid or openid (with a preference for unionid). If the user does not exist, a new user will be automatically created. Upon successful login, CIAM will map the user information returned from WeChat to the local user data based on the attribute mapping configuration of the authentication source.

Note

- Before using this API, we recommend that you read the corresponding official WeChat login documentation to understand the detailed login process: WeChat Login for Website Applications, WeChat Official Account Web Authorization, WeChat Login for Mobile Applications.
- If a WeChat Official Account or mobile app needs to obtain user information such as nicknames and profile photos, please ensure that the authorization is initiated with `snsapi_userinfo` as the `scope`.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Content-Type: application/json
Host: sample.portal.tencentciam.com

{
 "grant_type" : "http://tencentciam.com/oauth2/grant-type/social/wechat/code",
 "client_id" : "TENANT_CLIENT_ID",
 "client_secret" : "TENANT_CLIENT_SECRET",
 "auth_source_id" : "MOCK_WECHAT_AUTH_SOURCE_ID",
 "code" : "MOCK_CODE"
}
```

Request Body JSON Parameters

JSON Path	Data Type	Description
grant_type	String	Enter the fixed value `http://tencentciam.com/oauth2/grant-type/social/wechat/code` .
client_id	String	Client ID of the application.
client_secret	String	The client_secret of the application. This parameter must be passed for web applications; it is not required for single-page applications and mobile apps.
code	String	The temporary authorization ticket `code` obtained after getting user authorization on WeChat.
auth_source_id	String	Choose any one of the following authentication source IDs based on your actual needs: • PC WeChat Login Authentication Source ID • WeChat Webpage Login Authentication Source ID • Mobile App WeChat Login Authentication Source ID
getUserInfo	Boolean	If you need to obtain user information such as nicknames and profile photos in addition to the user's WeChat openid and unionid, pass `true` for this parameter. Otherwise, this parameter can be omitted. Please note: When this parameter is set to `true` , if you are using a WeChat Official Account or a mobile app, please

		ensure that the authorization is initiated with `snsapi_userinfo` as the `scope` .

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiI3NjVkMjImYi05ODE1LTRmZGQtYmU5NS00NzJhOTYzZmQ5
  "refresh_token" : "4gCXN_3PuKIXG1MdhqxJWhHGpeC_sjphxwx6AqMsyENU-Hj2rEf94q1x
  "scope" : "openid",
  "id_token" : "eyJraWQiOiI3NjVkMjImYi05ODE1LTRmZGQtYmU5NS00NzJhOTYzZmQ5YmYi
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	OAuth 2.0 Access Token (JWT)。
token_type	String	Token type, currently, the returned fixed value is `Bearer` .
expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Access Token scope。
refresh_token	String	OAuth 2.0 Refresh Token。
id_token	String	OIDC ID Token (JWT)。

Exception response sample

- Invalid code.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_grant",
"error_description" : "WeChat error: 40029"
}
```

- The user status is abnormal (such as being locked or frozen).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Abnormal user status"
}
```

- The authentication source does not exist or is not enabled.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Auth source not found or disabled"
}
```

- The authentication source is not a `WeChat Mini Program In-App Login` authentication source.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Wrong auth source type"
}
```

- The authentication source is not associated with an application.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_auth_source",
"error_description" : "Auth source and application not associated"
}
```

WeChat Mini Program Login (API Mode)

Last updated：2023-09-04 10:41:51

API Description

After the WeChat Mini Program application obtains the temporary login credential `code` from WeChat via `wx.login()`, it calls this API to validate the `code`, obtains the CIAM's Access Token and ID Token, and completes the login process.
CIAM will associate the local user with WeChat's `unionid` or `openid` (with a preference for `unionid`). If the user does not exist, a new user will be automatically created. Upon successful login, CIAM will map the user information returned by WeChat to the local user data based on the attribute mapping configuration of the authentication source.

Note

- You can also access the WeChat Mini Program more easily and quickly through the SDK method. For more information, please refer to Accessing WeChat Mini Program (SDK Mode).
- Before using this API, it is recommended that you read the official WeChat Mini Program login documentation to understand the detailed login process for the mini program.

Supported Application Types

Mini Program Application.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Content-Type: application/json
Host: sample.portal.tencentciam.com

{
 "grant_type" : "http://tencentciam.com/oauth2/grant-type/social/wechat/jscode",
 "client_id" : "WECHAT_MINI_PROGRAM_CLIENT_ID",
 "auth_source_id" : "WECHAT_MINI_PROGRAM_AUTH_SOURCE_ID",
 "js_code" : "MOCK_JS_CODE"
}
```

Request Body JSON Parameters

JSON Path	Data Type	Description
grant_type	String	Enter the fixed value `http://tencentciam.com/oauth2/grant-type/social/wechat/jscode` .
client_id	String	The client_id of the mini program application.
auth_source_id	String	WeChat Mini Program internal login authentication source ID.
js_code	String	The temporary login credential code returned by the WeChat Mini Program's wx.login() interface.
getUserInfo	Boolean	If you need to obtain user information such as nickname and avatar, pass `true` for this parameter, otherwise, this parameter can be omitted. When this parameter is set to `true` , both `encryptedData` and `iv` parameters must be passed simultaneously.
encryptedData	String	The encrypted user information data, encryptedData, returned by the wx.getUserProfile() interface.
iv	String	The initial vector iv of the encryption algorithm returned by the wx.getUserProfile() interface.
getPhoneNumber	Boolean	If you need to obtain the user's mobile number, pass this parameter as `true` . Otherwise, this parameter can be omitted.

		When this parameter is set to `true` , the `phone_code` parameter must also be passed.
phone_code	String	The dynamic token code returned by the WeChat Mini Program's get mobile number API.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiI3NjVkMjImYi05ODE1LTRmZGQtYmU5NS00NzJhOTYzZmQ5
  "scope" : "openid",
  "id_token" : "eyJraWQiOiI3NjVkMjImYi05ODE1LTRmZGQtYmU5NS00NzJhOTYzZmQ5YmYi
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	OAuth 2.0 Access Token (JWT)。
token_type	String	Token type, currently, the returned fixed value is `Bearer` .
expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Access Token scope。
refresh_token	String	OAuth 2.0 Refresh Token。
id_token	String	OIDC ID Token (JWT)。

Exception response sample

- The js_code is invalid.

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

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

{
"error" : "invalid_grant",
"error_description" : "WeChat error: 40029"
}
```

- The user status is abnormal (such as being locked or frozen).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant",
"error_description" : "Abnormal user status"
}
```

- The authentication source does not exist or is not enabled.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Auth source not found or disabled"
}
```

- The authentication source is not a `WeChat Mini Program In-App Login` authentication source.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_auth_source",
"error_description" : "Wrong auth source type"
}
```

- The authentication source is not associated with an application.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_auth_source",
"error_description" : "Auth source and application not associated"
}
```

- Specified to retrieve user information, but essential parameters are missing.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Missing required parameter: encryptedData"
}
```

- Specified to obtain the user's phone number, but necessary parameters are missing.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Missing required parameter: phone_code"
}
```

Send OTP Verification Code

Last updated：2023-09-04 14:15:51

API Description

Dispatch SMS or email OTP codes to users for the purposes of logging in, registering, or updating user information.

Supported Application Types

Web Applications, M2M Applications.

Request method

```
POST
```

Request path

```
/otp/send
```

Request Content-Type

```
application/json
```

Sample Request

- In the SMS OTP login scenario, dispatch an SMS verification code for login purposes.

```
POST /otp/send HTTP/1.1
Content-Type: application/json
Authorization: Basic VEVOQU5UX0NMSUVOVF9JRDpURU5BTlRfQ0xJRU5UX1NFQ1JFVA=
Host: sample.portal.tencentciam.com

{
"usage" : "login",
"phone_number" : "13612345678",
"auth_source_id" : "MOCK_SMS_OTP_AUTH_SOURCE_ID"
}
```

- In the context of email OTP login, dispatch an email verification code for sign–in purposes.

```
POST /otp/send HTTP/1.1
Content-Type: application/json
Authorization: Basic Q0xJRU5UXzRfSUQ6Q0xJRU5UXzRfU0VDUkVU
Host: sample.portal.tencentciam.com

{
"usage" : "login",
"email" : "MOCK_USERNAME@example.com",
"auth_source_id" : "MOCK_EMAIL_OTP_AUTH_SOURCE_ID"
}
```

- In the user registration scenario, send an SMS verification code for mobile number binding.

```
POST /otp/send HTTP/1.1
Content-Type: application/json
Authorization: Basic Q0xJRU5UXzRfSUQ6Q0xJRU5UXzRfU0VDUkVU
Host: sample.portal.tencentciam.com

{
"usage" : "signup",
"phone_number" : "13612345678"
}
```

- In the user registration scenario, dispatch an email verification code for binding the email address.

```
POST /otp/send HTTP/1.1
Content-Type: application/json
Authorization: Basic Q0xJRU5UXzRfSUQ6Q0xJRU5UXzRfU0VDUkVU
Host: sample.portal.tencentciam.com

{
"usage" : "signup",
"email" : "MOCK_USERNAME@example.com"
}
```

- Dispatch SMS verification codes for binding or updating mobile numbers in user information update scenarios.

```
POST /otp/send HTTP/1.1
```

```
Content-Type: application/json
Authorization: Basic Q0xJRU5UXzRfSUQ6Q0xJRU5UXzRfU0VDUkVU
Host: sample.portal.tencentciam.com

{
"usage" : "update_userinfo",
"phone_number" : "13612345678"
}
```

- In the scenario of resetting the password, dispatch an email verification code.

```
POST /otp/send HTTP/1.1
Content-Type: application/json
Authorization: Basic Q0xJRU5UXzRfSUQ6Q0xJRU5UXzRfU0VDUkVU
Host: sample.portal.tencentciam.com

{
"usage" : "reset_password",
"email" : "MOCK_USERNAME@example.com"
}
```

Request Header

Name	Description
Authorization	HTTP Basic Authentication request header, formatted as `Basic <credentials>` , where `Basic` is a fixed string, and `<credentials>` is calculated as `base64(url_encode(client_id) + ":" + url_encode(client_secret))` . A single space separates `Basic` and `<credentials>` .

Request Body JSON Parameters

JSON Path	Data Type	Description
usage	String	Use cases for OTP codes. • For SMS and email OTP login scenarios, input `login` . • For user registration scenarios, input `signup` . • For updating user information scenarios, input `update_userinfo` . • For resetting user password scenarios, input `reset_password` . • In the absence of input parameters, it is assumed to

		represent a login scenario.
phone_number	String	The user's mobile number, restricted to the 11-digit numbers of the three major domestic carriers. Pass this parameter when sending an SMS OTP code.
email	String	The user's email address. This parameter is passed when sending the OTP verification code via email.
auth_source_id	String	SMS OTP or Email OTP authentication source ID. This can be viewed on the General Authentication Source list page in the console. For SMS and Email OTP login scenarios, pass this parameter and the system will use the verification code length and validity period configured in the authentication source. In other scenarios, if this parameter is not passed, the system will default to a 6-digit numeric verification code with a validity period of 60 seconds.

Example of a successful response

Verification code sent successfully.

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

{
  "otp_token" : "MOCK_OTP_TOKEN"
}
```

Response Parameters

Parameter	Data Type	Description
otp_token	String	OTP token, to be carried for subsequent OTP verification. Valid for 5 minutes.

Exception response sample

- Incorrect phone number format.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "malformed_phone_number"
}
```

- The email format is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "malformed_email"
}
```

- Unable to send SMS due to insufficient quota, typically because the free SMS quota has been exhausted. It is necessary to configure the SMS template in the console.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "insufficient_sms_quota"
}
```

- Unable to send emails due to insufficient email quota, typically because the free email quota has been exhausted. It is necessary to configure the Email Template in the console.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "insufficient_email_quota"
}
```

- The email address either does not exist or is on the blocklist.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_email"
}
```

- Failed to send verification code.

```
HTTP/1.1 503 Service Unavailable
Content-Type: application/json;charset=UTF-8

{
"error" : "temporarily_unavailable",
"error_description" : "Failed to send OTP. Please try again later."
}
```

- Registration scenario, email is in use.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
  "error": "email_is_used"
}
```

- In the registration scenario, the mobile number is in use.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8


{
  "error": "phone_number_is_used"
}
```

- The frequency of sending SMS to a single mobile number has exceeded the limit.
  - If you are using a self-purchased SMS service, you can adjust the SMS delivery rate limit policy in the SMS console.
  - If the free SMS quota is being utilized, the frequency limit is as follows: no more than 50 SMS messages can be sent to the same mobile number within a natural day; for identical content, no more than one SMS message can be sent to the same mobile number within 30 seconds.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "sms_rate_limit_exceeded",
"error_description" : "SMS rate limit exceeded for same phone number"
}
```

Get Token
PKCE Authorization Code Mode

Last updated：2023-09-04 10:42:24

API Description

Upon obtaining the `code` returned by the authentication portal through the PKCE authorization code mode, the application system calls this API to retrieve the Access Token and ID Token, thereby completing the login process.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Host: sample.portal.tencentciam.com
Content-Type: application/x-www-form-urlencoded

client_id=TENANT_CLIENT_ID&grant_type=authorization_code&code=MOCK_CODE&redir
```

Request Parameters

Category	Optional	Description

client_id	false	The application's `client_id`. It must be consistent with the one used during authorization acquisition.
grant_type	false	Enter the fixed value `authorization_code`.
code	false	The authorization code returned during the acquisition of authorization.
redirect_uri	false	The redirect URL after successful authorization. It must be consistent with the address specified when obtaining the authorization.
code_verifier	false	PKCE code_verifier. It must be consistent with the code_verifier used to generate the code_challenge during authorization.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
 "access_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNI
 "refresh_token" : "8FuXWpwMZI9oA8ASvCUrqap61N7RvPON6DjWFk-Saiv4dOR8y2tNf9el
 "scope" : "openid",
 "id_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNDYiLC
 "token_type" : "Bearer",
 "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	OAuth 2.0 Access Token (JWT).
refresh_token	String	OAuth 2.0 Refresh Token.
scope	String	Scope of the Access Token.
id_token	String	OIDC ID Token (JWT).

token_type	String	Token type, currently set to a fixed value: `Bearer` .
expires_in	Number	Validity period of the Access Token, measured in seconds.

Note
CIAM returns an ID Token in JWT format. Please refer to the OIDC official documentation for decrypting and verifying the ID Token. Alternatively, you can use the relevant development libraries to complete the decryption and verification. The public key required for verification can be obtained by calling the Get JWT Public Key API.

Exception response sample

- The client_id parameter is missing or incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
"error" : "invalid_request"
}
```

- The client_id is inconsistent with the one used when obtaining authorization and retrieving the Token.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
"error" : "invalid_client"
}
```

- The grant_type parameter is incorrect.

```
HTTP/1.1 401 Unauthorized
```

- The 'code' parameter is incorrect.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "invalid_client"
}
```

- The code_verifier parameter is incorrect.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
"error" : "invalid_client"
}
```

General Authorization Code Mode

Last updated：2023-09-04 10:42:32

API Description

Upon obtaining the `code` returned by the authentication portal through the standard authorization code mode, the application system calls this API to get the Access Token and ID Token, thereby completing the login process.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Host: sample.portal.tencentciam.com
Content-Type: application/x-www-form-urlencoded

client_id=TENANT_CLIENT_ID&client_secret=TENANT_CLIENT_SECRET&grant_type=authc
```

Request Parameters

Category	Optional	Description
client_	false	The `client_id` of the application. It must be consistent with the one

id		used during authorization.
client_ secret	false	The application's `client_secret` . Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "client_secret".
grant_ type	false	Enter the fixed value `authorization_code` .
code	false	The authorization code returned during the acquisition of authorization.
redire ct_uri	false	The redirect URL after successful authorization. It must be consistent with the address specified when obtaining the authorization.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNI
  "refresh_token" : "Ugvo1lO7Se8vvIPrIOwn_eBe0hoi5-5ynR3H-aFYl0e1Gej-SfUAaBDBXkW
  "scope" : "openid",
  "id_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNDYiLC
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Note
CIAM returns an ID Token in JWT format. Please refer to the OIDC official documentation for decrypting and verifying the ID Token. Alternatively, you can use the relevant development libraries to complete the decryption and verification. The public key required for verification can be obtained by calling the Get JWT Public Key API.

Response Parameters

Category	Data Type	Description
access_toke	String	OAuth 2.0 Access Token (JWT).

n		
token_type	String	Token type, currently set to a fixed value: `Bearer` .
expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Scope of the Access Token.
refresh_token	String	OAuth 2.0 Refresh Token.
id_token	String	OIDC ID Token (JWT).

Note
CIAM returns an ID Token in JWT format. Please refer to the OIDC official documentation for decrypting and verifying the ID Token. Alternatively, you can use the relevant development libraries to complete the decryption and verification. The public key required for verification can be obtained by calling the Get JWT Public Key API.

Exception response sample

- The client_id parameter is missing or incorrect.

```
HTTP/1.1 401 Unauthorized
```

- The client_id is inconsistent with the one used when obtaining authorization and retrieving the Token.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_client"
}
```

- The grant_type parameter is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
"error" : "unsupported_grant_type",
"error_description" : "OAuth 2.0 Parameter: grant_type",
"error_uri" : "https://datatracker.ietf.org/doc/html/rfc6749#section-5.2"
}
```

- The 'code' parameter is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_grant"
}
```

Client Credentials Mode

Last updated：2023-09-04 10:42:37

API Description

Obtain an Access Token using the OAuth client credentials mode (client_credentials).

Supported Application Types

Web Applications, M2M Applications.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Host: sample.portal.tencentciam.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=TENANT_CLIENT_ID&client_secret=TENANT_CLII
```

Request Parameters

Category	Optional	Description
grant_type	false	Enter the fixed value `client_credentials`.
client_i	false	The application's `client_id`. Refer to Application Management Page

d		> **Select Specific Application** > Click on **Application Configuration** > Corresponding "Client Id".
client_secret	false	The `client_secret` of the application. Refer to Application Management Page > **Select Specific Application** > Click on **Application Configuration** > Corresponding "client_secret".
scope	true	The `scope` for which authorization is being requested, multiple `scope` should be separated by spaces.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiJmOTY5NGQ5My1kNTQxLTQ5ODUtODhkYy00MjIyOTg3MzA
  "scope" : "identity_proofing",
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	Access Token (JWT).
token_type	String	Token type, currently, the returned fixed value is `Bearer`.
expires_in	Number	Validity period of the Access Token, measured in seconds.
scope	String	Scope of the Access Token.

Note
The response in the client credentials mode does not include a Refresh Token. When the Access Token expires, the application should call this interface again to obtain a new Access Token.

Exception response sample

- The application does not exist, is not enabled, or the application key verification has failed.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_client"
}
```

- The application does not have permission to obtain a Token using the `client_credentials` mode.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "unauthorized_client"
}
```

- The scope parameter is incorrect or exceeds application permissions.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_scope"
}
```

Get User Information

Last updated：2023-09-04 10:42:44

API Description

Obtain the user information of the currently logged-in user. When invoking this API, it is necessary to carry the Access Token with `openid scope` obtained upon successful login.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
GET
```

Request path

```
/userinfo
```

Sample Request

```
GET /userinfo HTTP/1.1
Authorization: Bearer ACCESS_TOKEN_WITH_OPENID_SCOPE
Host: sample.portal.tencentciam.com
```

Request Header

Name	Description
Authorization	OAuth 2.0 Bearer Token, formatted as `Bearer <Token>`, where `Bearer` is a fixed string, `<Token>` is the Access Token with `openid scope` obtained upon successful user login, and there is a space between `Bearer` and `<Token>`.

Example of a successful response

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

```
{
  "sub" : "MOCK_USER_ID",
  "email" : "MOCK_USERNAME@example.com",
  "name" : "MOCK_NAME",
  "nickname" : "MOCK_NICKNAME",
  "zoneinfo" : "Asia/Shanghai",
  "locale" : "zh-CN"
}
```

Response Parameters

Category	Data Type	Description
sub	String	User identifier, unique within the user pool.

Note
Apart from the `sub` field which is always returned, the remaining fields to be returned are determined by the `Claims` in the application parameter configuration.

Exception response sample

- Access Token has not been carried.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request", error_description="Bearer token ı
```

- The Access Token is invalid (for instance, due to incorrect format, expiration, or cancellation).

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="Error decoding J
```

- The Access Token does not possess the `openid scope`.

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_scope", error_description="The reques
```

- No user corresponding to the Access Token was found.

```
HTTP/1.1 404 Not Found
Content-Type: application/json;charset=UTF-8

{
"error" : "user_not_found"
}
```

Update User Information

Last updated：2023-09-04 10:42:51

API Description

Update the personal information of the logged-in user. When invoking this interface, carry the Access Token with openid scope obtained upon successful login. If updating the mobile number or email, first invoke the Send OTP Verification Code interface to send a verification code to the user.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
PATCH
```

Request path

```
/userinfo
```

Request Content-Type

```
application/json
```

Sample Request

```
PATCH /userinfo HTTP/1.1
Content-Type: application/json
Authorization: Bearer ACCESS_TOKEN_WITH_OPENID_SCOPE
Host: sample.portal.tencentciam.com

{
  "nickname" : "MOCK_NICKNAME"
}
```

Request Header

Name	Description
Authorization	OAuth 2.0 Bearer Token, formatted as `Bearer <Token>`, where `Bearer` is a fixed string, `<Token>` is the Access Token with `openid scope` obtained upon successful user login, and there is a space between `Bearer` and `<Token>`.

Request Body JSON Parameters

JSON Path	Data Type	Description
phone_number	String	The user's mobile number, limited to the 11-digit mobile numbers of the three major domestic operators. When passing this parameter, you must also pass the `phone_number_otp_token` and `phone_number_otp` parameters simultaneously.
phone_number_otp_token	String	The `otp_token` returned by the server after successfully sending the SMS verification code.
phone_number_otp	String	The OTP verification code received on the user's mobile phone.
email	String	The user's email address. When passing this parameter, you must also pass the `email_otp_token` and `email_otp` parameters simultaneously.
email_otp_token	String	The `otp_token` returned by the server after successfully sending the email verification code.
email_otp	String	The OTP verification code received by the user's email.
name	String	User Name.
nickname	String	User Nickname.
zoneinfo	String	User's time zone, such as `Asia/Shanghai` or `Europe/Paris`.
locale	String	User locale information, such as `zh-CN` or `en-US`.

Note
The value of other parameters is the user attribute identifier. The attribute identifier can be viewed on the attribute details interface of the Custom Attribute Page.

Example of a successful response

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

{
 "sub" : "MOCK_USER_ID",
 "email" : "MOCK_USERNAME@example.com",
 "name" : "MOCK_NAME",
 "nickname" : "MOCK_NICKNAME",
 "zoneinfo" : "Asia/Shanghai",
 "locale" : "zh-CN"
}
```

Note
Apart from the `sub` field which is always returned, the remaining fields to be returned are determined by the `Claims` in the application parameter configuration.

Exception response sample

- The input includes an unknown attribute.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Unknown attribute(s) found."
}
```

- The input parameters include attributes that cannot be modified.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Unsupported user attribute(s) found."
}
```

- The format of the phone number is invalid.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "malformed_phone_number"
}
```

- The phone number already exists.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "duplicate_phone_number"
}
```

- The `phone_number_otp_token` is incorrect, has expired, or the parameters used during registration are inconsistent with those used when sending the verification code (for example, the phone numbers are different).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_phone_number_otp_token"
}
```

- The `phone_number_otp` is incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_phone_number_otp"
}
```

- The email format is invalid.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

```

```
{
"error" : "malformed_email"
}
```

- This email address already exists.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "duplicate_email"
}
```

- The `email_otp_token` is incorrect, has expired, or the parameters used during registration are inconsistent with those used when sending the verification code (for example, different email addresses).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp_token"
}
```

- The `email_otp` is incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp"
}
```

- Invalid input parameter value (for instance: does not comply with the user attribute regular expression).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "illegal_parameter_value"
}
```

- `bearer_token` is missing.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request", error_description="Bearer token ı
```

- `bearer_token` error.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="Error decoding J
```

- The `bearer_token` is invalid.

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_scope", error_description="The reques
```

Modify User Password

Last updated：2023-09-04 10:42:57

API Description

Change the password of the logged-in user. When calling this interface, you need to carry the Access Token with `openid scope` obtained after successful login. The new password must comply with the password policy of the account password authentication source associated with the current application, and it cannot be the same as the previous N historical passwords specified in the policy.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
POST
```

Request path

```
/change_user_password
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /change_user_password HTTP/1.1
Content-Type: application/json
Authorization: Bearer ACCESS_TOKEN_WITH_OPENID_SCOPE
Host: sample.portal.tencentciam.com

{
  "old_password" : "MOCK_PASSWORD",
  "new_password" : "MOCK_NEW_PASSWORD"
}
```

Request Header

Name	Description
Authorization	OAuth 2.0 Bearer Token, formatted as `Bearer <Token>` , where `Bearer` is a fixed string, `<Token>` is the Access Token with `openid scope` obtained upon successful user login, and there is a space between `Bearer` and `<Token>` .

Request Body JSON Parameters

JSON Path	Data Type	Description
old_password	String	Current password.
new_password	String	New password.

Example of a successful response

```
HTTP/1.1 200 OK
```

Exception response sample

- Incorrect old password.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "wrong_old_password"
}
```

- The new password is identical to the old one.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "duplicate_password"
}
```

- The new password is identical to the historical password.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "recurrent_password"
}
```

- The new password does not meet the password policy.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_new_password"
}
```

- User not found.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "user_not_found"
}
```

- The user is frozen.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "abnormal_user_status",
"error_description" : "User is frozen."
}
```

- The user is locked.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "abnormal_user_status",
"error_description" : "User is locked."
}
```

- `bearer_token` **is missing.**

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request", error_description="Bearer token ı
```

- `bearer_token` **error.**

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="Error decoding J
```

- **The** `bearer_token` **is invalid.**

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_scope", error_description="The reques
```

Reset User Password

Last updated: 2023-09-04 10:43:02

API Description

Reset the user's password. Prior to invoking this API, it is necessary to first dispatch a verification code to the user via the Send_OTP_Code interface.

Note
The new password must comply with the password policy of the account authentication source associated with the current application, and it cannot be identical to the N previous passwords specified in the policy.

Supported Application Types

Web Applications, Single-Page Applications, Mobile Apps.

Request method

```
POST
```

Request path

```
/reset_user_password
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /reset_user_password HTTP/1.1
Content-Type: application/json
Authorization: Basic VEVOQU5UX0NMSUVOVF9JRDpURU5BTlRfQ0xJRU5UX1NFQ1JFVA==
Host: sample.portal.tencentciam.com

{
  "password" : "MOCK_PASSWORD",
  "email" : "MOCK_EMAIL@163.com",
    "email_otp" : "MOCK_EMAIL_OTP",
    "email_otp_token" : "MOCK_EMAIL_OTP_TOKEN"
  }
```

Request Header

Name	Description
Authorization	HTTP Basic Authentication request header, formatted as Basic <credentials>, where Basic is a fixed string, and <credentials> is calculated as base64(url_encode(client_id) + ":" + url_encode(client_secret)). A single space separates Basic and <credentials>.

Request Body JSON Parameters

JSON Path	Data Type	Description
client_id	String	The client_id of the application. It must be consistent with the one used when sending the verification code.
client_secret	String	The client_secret of the application. This parameter must be passed for Web applications. It is not required for Single Page Applications and Mobile Apps.
password	String	New password.
email	String	The user's email address. This parameter is passed when sending the OTP verification code via email.
email_otp_token	String	The otp_token returned by the server after the email verification code is successfully sent.
email_otp	String	The OTP verification code received by the user's email.
phone_number	String	The user's mobile number. This parameter is passed when sending the SMS OTP verification code.
phone_number_otp_token	String	The otp_token returned by the server after the successful dispatch of the SMS verification code.
phone_number_otp	String	The OTP verification code received on the user's mobile phone.

Example of a successful response

Password reset successful.

```
HTTP/1.1 200 OK
```

Exception response sample

- The new password is identical to the historical password.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "recurrent_password"
}
```

- The new password does not meet the password policy.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_new_password"
}
```

- No user corresponding to the email was found.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "user_not_found"
}
```

- The user is in a frozen state, thus password reset is not feasible.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "abnormal_user_status",
"error_description" : "User is frozen."
}
```

- The email_otp_token is either incorrect, expired, or the parameters used during the password reset are inconsistent with those used when sending the verification code (for instance, the email addresses differ).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp_token"
}
```

- The email_otp is either incorrect or has expired.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "bad_email_otp"
}
```

Identity Verification
Two-factor Real-name Verification

Last updated：2023-09-04 10:43:08

API Description

This API conducts real-name verification of natural persons based on two elements: ID number and name. The document type is limited to the ID card of mainland China residents.

Note
Before invoking this API, you need to configure the template information on the Template Settings > **Identity Verification Template** page and activate the corresponding identity verification service.

Supported Application Types

Web Applications, M2M Applications.

Request method

```
POST
```

Request path

```
/idproofing/verify-by-id-name
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /idproofing/verify-by-id-name HTTP/1.1
Content-Type: application/json
Authorization: Bearer ACCESS_TOKEN_WITH_IDPROOFING_SCOPE
Host: sample.portal.tencentciam.com

{
  "id" : "110101190001010000",
  "name" : "John Doe"
}
```

Request Header

Name	Description
Authorization	OAuth 2.0 Bearer Token, formatted as `Bearer <Token>` , where `Bearer` is a fixed string, `<Token>` is an `Access Token` with the `identity_proofing` scope, and there is a space between `Bearer` and `<Token>` .

Note
Before accessing this API, call the Get Token in Client Credential Mode API to obtain the required Access Token.

Request Body JSON Parameters

JSON Path	Data Type	Description
id	String	ID Number. Restricted to the ID cards of mainland China residents.
name	String	Full Name.

Successful Verification Response Example

```
HTTP/1.1 200 OK
```

Failed Verification Response Example

- The ID number and name do not match.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "verification_failure",
"error_description" : "Verification failed: provided info do not match."
}
```

- The format of the ID number or name is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Malformed input parameter"
}
```

- The configuration of the CIAM real-name authentication service is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "CIAM ID proofing has not been configured yet"
}
```

- The identity verification service is unavailable.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "Third party ID Proofing service unavailable"
}
```

- Access Token has not been carried.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request", error_description="Bearer token ı
```

- The Access Token is invalid (for instance, due to incorrect format, expiration, or cancellation).

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="Error decoding J
```

- **The Access Token does not possess the `identity_proofing` scope.**

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_scope", error_description="The reques
```

Three-element Real-name Verification

Last updated: 2023-09-04 10:43:13

API Description

This API conducts real-name verification for natural persons based on three elements: ID number, name, and mobile phone number. The ID type is limited to the ID card of mainland China residents, and the mobile phone number is limited to the three major domestic operators.

Note
Before invoking this API, you need to configure the template information on the Template Settings > **Identity Verification Template** page and activate the corresponding identity verification service.

Supported Application Types

Web Applications, M2M Applications.

Request method

```
POST
```

Request path

```
/idproofing/verify-by-id-name-phone
```

Request Content-Type

```
application/json
```

Sample Request

```
POST /idproofing/verify-by-id-name-phone HTTP/1.1
Content-Type: application/json
Authorization: Bearer ACCESS_TOKEN_WITH_IDPROOFING_SCOPE
```

```
Host: sample.portal.tencentciam.com

{
  "id" : "110101190001010000",
  "name" : "John Doe",
  "phoneNumber" : "13610000000"
}
```

Request Header

Name	Description
Authorization	OAuth 2.0 Bearer Token, formatted as `Bearer <Token>` , where `Bearer` is a fixed string, `<Token>` is an `Access Token` with the `identity_proofing` scope, and there is a space between `Bearer` and `<Token>` .

Note
Before accessing this API, call the Get Token in Client Credential Mode API to obtain the required Access Token.

Request Body JSON Parameters

JSON Path	Data Type	Description
id	String	ID Number. Restricted to the ID cards of mainland China residents.
name	String	Full Name.
phoneNumber	String	Mobile numbers from the three major domestic operators (11 digits).

Successful Verification Response Example

```
HTTP/1.1 200 OK
```

Failed Verification Response Example

- The three elements: ID number, name, and mobile phone number do not match.

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

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

{
"error" : "verification_failure",
"error_description" : "Verification failed: provided info do not match."
}
```

- The format of the ID number, name, or mobile phone number is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "invalid_request",
"error_description" : "Malformed input parameter"
}
```

- The configuration of the CIAM real-name authentication service is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "CIAM ID Proofing is misconfigured"
}
```

- The identity verification service is unavailable.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error" : "misconfigured",
"error_description" : "Third party ID Proofing service unavailable"
}
```

- Access Token has not been carried.

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: Bearer error="invalid_request", error_description="Bearer token
```

- The Access Token is invalid (for instance, due to incorrect format, expiration, or cancellation).

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="Error decoding J
```

- The Access Token does not possess the `identity_proofing` scope.

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_scope", error_description="The reques
```

Get JWT Public Key

Last updated：2023-09-04 10:43:19

API Description

The JWT Public Key is utilized to validate ID Tokens and Access Tokens in JWT format.

Note
JWT keys are auto-generated upon the creation of a user directory, with each user directory possessing a unique key.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
GET
```

Request path

```
/oauth2/jwks
```

Sample Request

```
GET /oauth2/jwks HTTP/1.1
Host: sample.portal.tencentciam.com
```

Example of a successful response

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

{
  "keys" : [ {
    "kty" : "RSA",
    "e" : "AQAB",
    "kid" : "f9694d93-d541-4985-88dc-42229873008e",
    "n" : "wYmf-IL7_pXqEjtfHme7KqS06hRQ0BzhTzORjgwnsJD_CPexMHQAd82vZfOQioW9o
  } ]
}
```

Response Parameters

Category	Data Type	Description
keys	Array	An array containing public keys.
keys[].kty	String	Key types, such as RSA.
keys[].kid	String	Key Identifier.
keys[].e	String	RSA Public Key.
keys[].n	String	RSA Public Key.

Refresh Token

Last updated：2023-09-04 10:43:24

API Description

Obtain a new access_token using the refresh_token.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
POST
```

Request path

```
/oauth2/token
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/token HTTP/1.1
Host: sample.portal.tencentciam.com
Content-Type: application/x-www-form-urlencoded

client_id=TENANT_CLIENT_ID&client_secret=TENANT_CLIENT_SECRET&grant_type=refres
```

Request Parameters

Category	Optional	Description
client_id	false	The `client_id` of the application. It must be consistent with the one used during authorization.

client_secret	false	The application's `client_secret`. It can be viewed on the Basic Information page of the Tenant Management Platform.
grant_type	false	Enter the fixed value `refresh_token`.
refresh_token	false	The `refresh_token` returned when obtaining the token.

Example of a successful response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNI
  "refresh_token" : "MOCK_REFRESH_TOKEN",
  "scope" : "openid",
  "id_token" : "eyJraWQiOiJkNDliYzUwNS01NTcyLTRlZDYtOWU0OC0zODhjM2Q0NGJiNDYiLC
  "token_type" : "Bearer",
  "expires_in" : 299
}
```

Response Parameters

Category	Data Type	Description
access_token	String	Refreshed OAuth 2.0 Access Token (JWT).
refresh_token	String	Refreshed OAuth 2.0 Refresh Token.
scope	String	Scope of the Access Token.
id_token	String	Refreshed OIDC ID Token (JWT).
token_type	String	Token type, currently set to a fixed value: `Bearer`.
expires_in	Number	Validity period of the Access Token, measured in seconds.

Exception response sample

The refresh_token parameter is incorrect.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
  "error" : "invalid_grant"
}
```

Revoke Token

Last updated: 2023-09-04 14:19:47

API Description

Revoke OAuth 2.0 Token. If an access_token is provided, only that access_token will be revoked. If a refresh_token is provided, both the refresh_token and its associated access_token will be revoked.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
POST
```

Request path

```
/oauth2/revoke
```

Request Content-Type

```
application/x-www-form-urlencoded
```

Sample Request

```
POST /oauth2/revoke HTTP/1.1
Host: sample.portal.tencentciam.com
Content-Type: application/x-www-form-urlencoded

client_id=TENANT_CLIENT_ID&client_secret=TENANT_CLIENT_SECRET&token=MOCK_AC(
```

Request Parameters

Category	Optional	Description

client_ id	false	The application's `client_id` . It must be consistent with the one used when obtaining authorization and acquiring the Token.
client_ secret	false	The application's `client_secret` . It can be viewed on the Basic Information page of the Tenant Management Platform.
token	false	Value of the access_token or refresh_token.

Example of a successful response

```
HTTP/1.1 200 OK
```

Exception response sample

The client_id is inconsistent with the one used during login and token acquisition.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8

{
  "error" : "invalid_client"
}
```

Get OpenID Provider Configuration

Last updated: 2023-09-04 15:04:19

API Description

Applications can simplify local configurations by obtaining the configuration information of the OIDC authorization server through this interface. For detailed configuration information and its meaning, please refer to the OpenID Connect Discovery Standard.

Supported Application Types

Web applications, single-page applications, mobile apps, M2M applications, and mini program applications.

Request method

```
GET
```

Request path

```
/.well-known/openid-configuration
```

Sample Request

```
GET /.well-known/openid-configuration HTTP/1.1
Host: sample.portal.tencentciam.com
```

Example of a successful response

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

{
  "issuer" : "https://sample.portal.tencentciam.com",
  "authorization_endpoint" : "https://sample.portal.tencentciam.com/oauth2/authorize",
  "token_endpoint" : "https://sample.portal.tencentciam.com/oauth2/token",
  "token_endpoint_auth_methods_supported" : [ "client_secret_basic", "client_secret_post
  "jwks_uri" : "https://sample.portal.tencentciam.com/oauth2/jwks",
  "response_types_supported" : [ "code" ],
  "grant_types_supported" : [ "authorization_code", "client_credentials", "refresh_token" ]
  "subject_types_supported" : [ "public" ],
  "id_token_signing_alg_values_supported" : [ "RS256" ],
  "scopes_supported" : [ "openid" ]
}
```

Response Parameters

Category	Data Type	Description
issuer	String	OIDC Issuer.
authorization_endpoint	String	Service address for obtaining authorization via OAuth 2.0.
token_endpoint	String	OAuth 2.0 Token Retrieval Service URL.
jwks_uri	String	Service URL for obtaining JWT public key.
grant_types_supported	Array	Supported OAuth 2.0 Grant Types.
response_types_supported	Array	Obtain the Response Type supported by the OAuth 2.0 authorization service.
token_endpoint_auth_methods_supported	Array	Authentication methods supported by the OAuth 2.0 Token Acquisition Service.
subject_types_supported	Array	Supported OIDC Subject Identifier Types.
id_token_signing_alg_values_supported	Array	Supported JWS Signature Algorithms.
scopes_supported	Array	Supported OAuth 2.0 Scopes.