

Parties API

Step By Step

Sign up for a New Account

Step 1: Sign Up (duh)

* First Name

* Last Name

* Organization

My Applications

Step 2: Set Up (My Apps)

01

Create an
app in My Apps

02

Discover APIs

03

View
documentations

04

Try it

Add Application

Postman substitution



No description provided

Joels Testing



No description provided

Mashape Endpoint



No description provided

Authentication [\(learn more about DBS Authentication\)](#)

Note:

1. Please choose at least one authentication method.
2. Provision of **Trusted Partner** (Machine to Machine authentication) is subject to strict approval process (during application promotion).

☐ Trusted Partner ⓘ

Step 2.1: Authentication and Redirects

☒ DBS Customer ⓘ

☒ Internet Banking Credentials ⓘ

☒ Card ⓘ

Redirect URLs - one per line (maximum 10 redirect URL's are allowed of size 1024 each)

http://localhost:5500/index.html

Customer Client Id ⓘ

acce3a35-ec78-4541-95bb-938d9d346ef2

Customer Client Secret ⓘ

48b09a03-d4e4-49d3-a4bf-32a10f5a06b5

Customers



Step 2.2: Pick your API and browse documentation

Channel Preferences

Generates a list of customer preferences for a given channel, including channel profile details for a party and recently accessed entities.

[View Docs](#) | [Try It](#)

Parties

Enables the retrieval of party information, and provides details on the demographics regarding a specific party.

[View Docs](#) | [Try It](#)

Return the profile for a given party.

PARAMETERS

Path Parameters ?

→ partyId string **Required**
Tokenised Party Identifier

Header Parameters ?

→ clientId string **Required**
Client Identifier

→ accessToken string **Required**
Access Token

→ accept-version string
Specifies the acceptable version of the message set. If not specified, the latest version of the API will be considered. Example: 1.2

RESPONSE SAMPLES

• 200 Party's profile is returned • 500 Internal Server Error

```
{
  - "retailParty": {
    "partyId": "string",
    + "retailDemographic": { - },
    + "employmentDetl": { - },
    + "contactDetl": { - }
  },
  - "corporateParty": {
    "partyId": "string",
    + "corporateDemographic": { - },
    + "contactDetl": { - }
  }
}
```

VERSION HISTORY

AUTHENTICATION

PAGINATION

FREQUENTLY ASKED QUESTIONS

KNOWN ISSUES

THROTTLING (RATE LIMITS)

PARTIES

[GET](#) Retrieve Party's profile

Customers



Channel Preferences

Generates a list of customer preferences for a given channel, including channel profile details for a party and recently accessed entities.

[View Docs](#) | [Try It](#)



Step 3: Deploy!

Parties

Enables the retrieval of party information, and provides details on the demographics regarding a specific party.

[View Docs](#) | [Try It](#)

VERSION HISTORY

AUTHENTICATION

PAGINATION

FREQUENTLY ASKED QUESTIONS

KNOWN ISSUES

THROTTLING (RATE LIMITS)

PARTIES

[GET](#) Retrieve Party's profile

Return the profile for a given party.

PARAMETERS

Path Parameters ?

→ partyId string **Required**
Tokenised Party Identifier

Header Parameters ?

→ clientId string **Required**
Client Identifier

→ accessToken string **Required**
Access Token

→ accept-version string
Specifies the acceptable version of the message set. If not specified, the latest version of the API will be considered. Example: 1.2

RESPONSE SAMPLES

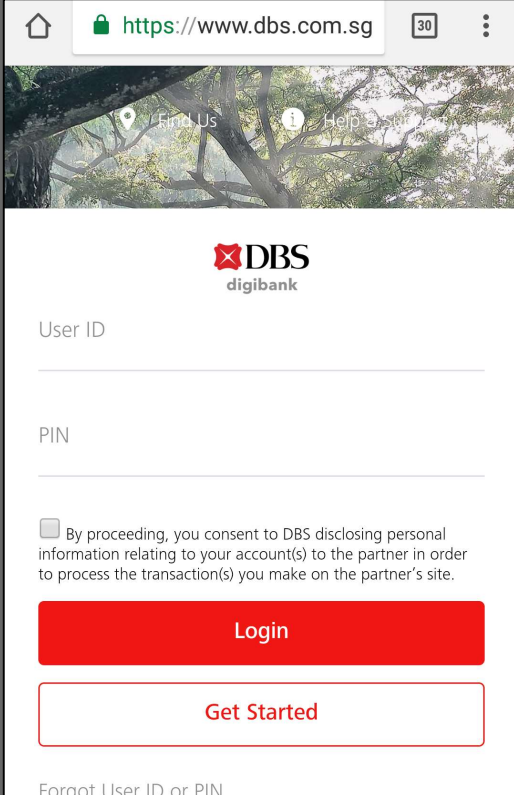
200 Party's profile is returned 500 Internal Server Error

```
{
  - "retailParty": {
    "partyId": "string",
    + "retailDemographic": { - },
    + "employmentDetl": { - },
    + "contactDetl": { - }
  },
  - "corporateParty": {
    "partyId": "string",
    + "corporateDemographic": { - },
    + "contactDetl": { - }
  }
}
```

Step 3.1 : Redirect to OAuth

DBS Login

```
const clientID = '9e8b5831-6ea9-4b62-8e4e-4bb671b17d5a'  
const clientSecret = 'b55c828a-4ad6-4c9a-b8f9-98a1e88b1d87'  
  
const redirectURI =  
`http://localhost:${portNumber}/PostmanSub/postman.html`  
const authURL =  
`https://www.dbs.com/sandbox/api/sg/v1/oauth/authorize?client_id=  
${clientID}&redirect_uri=${redirectURI}&scope=Read&response_type=  
code&state=0399`
```



The screenshot shows the DBS digibank login interface. At the top, there's a navigation bar with a home icon, the URL <https://www.dbs.com.sg>, a tab count of 30, and a menu icon. Below the navigation bar is a header image with links for 'Find Us' and 'Help Us'. The main content area features the DBS digibank logo. There are two input fields: 'User ID' and 'PIN'. Below these fields is a consent checkbox with the text: 'By proceeding, you consent to DBS disclosing personal information relating to your account(s) to the partner in order to process the transaction(s) you make on the partner's site.' There are two buttons: a red 'Login' button and a white 'Get Started' button with a red border. At the bottom, there is a link for 'Forgot User ID or PIN'.

Step 3.2 : Get Access Code (*≠ Access Token*)



code=eGJu7ue0MHT3QXABEYf4LtmfMbY%3D

Step 3.3.1 : Get Access Token

```
const clientID = '9e8b5831-6ea9-4b62-8e4e-4bb671b17d5a'  
const clientSecret = 'b55c828a-4ad6-4c9a-b8f9-98a1e88b1d87'  
const authCode = btoa(`${clientID}:${clientSecret}`) [base64 encoding]  
const code = eGJu7ue0MHT3QXABEYf4LtmfMbY%3D
```

```
let fetchToken = {  
  method: 'POST',  
  body: `code=${code}&grant_type=authorization_code&redirect_uri=${redirectURI}`,  
  headers: {  
    'authorization': `Basic ${authCode}`,  
    'content-type': 'application/x-www-form-urlencoded',  
    'cache-control': 'no-cache',  
    'accept': 'application/json'  
  }  
}
```

```
let response = await fetch(`https://www.dbs.com/sandbox/api/sg/v1/oauth/tokens`,  
fetchToken)
```

Step 3.3.2 : Get Access Token

```
let parsed = await response.json()
console.log(parsed)
```

```
2. expire_in: "1538540733659"
```

```
3. party_id: "SVcwMzY="
```

5.token_type:"bearer"

Step 3.3.3 : Decode Access Token

```
access_token:"eyJhbGciOiJIUzI1NiJ9.eyJpc3MiIDogImh0dHBzOi8vY2FwaS5kYnMuY29tIiwiaWF0IiA6IDE1Mzg1MzcxMzM2NTksICJleHAiIDogMTUzODU0MDczMzY1OSwic3ViIiA6ICJTVmN3TXpZPSIsInB0eXR5cGUiIDogMSwiY2xuaWQiIDogIjllOGI1ODMxLTZlYTktNGI2M....."
```



```
function parseJwt (access_token) {  
  var base64Url = token.split('.')[1];  
  var base64 =  
    decodeURIComponent(atob(base64Url).split('').map(function(c) {  
      return '%' + ('00' +  
        c.charCodeAt(0).toString(16)).slice(-2);  
    })).join('');  
  
  return JSON.parse(base64);  
};
```



```
{  
  "iss": "https://capi.dbs.com",  
  "iat": 1538537133659,  
  "exp": 1538540733659,  
  "sub": "SVcwMzY=",  
  "ptytype": 1,  
  "clnid": "9e8b5831-6ea9-4b62-8e4e-4bb671b17d5a",  
  "clntype": "2",  
  "access": "1FA",  
  "scope": "2FA-SMS",  
  "aud": "https://capi.dbs.com/access",  
  "jti": "5475037324376589625",  
  "cin": "Q0IOMDAwMDAx"  
}
```

Step 3.4.1: Call your API of Choice! (Parties)

```
const accessToken =
"eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJkbG90dHBzOj8vY2FwaS5kYnMuY29tIiwiaWF0IiA6IDE1Mzg1MzcwMzY1MzY1ODMxLTZlYTktNGI2M.....

const clientID = '9e8b5831-6ea9-4b62-8e4e-4bb671b17d5a'
const partyID = 'Q010MDAwMDAx'

let fetchbalanceData = {
  method: 'GET',
  headers: {
    'accessToken': accessToken,
    'clientID': clientID,
    'accept': 'application/json'
  }
}

fetch(`https://www.dbs.com/sandbox/api/sg/v2/parties/${partyID}`, fetchbalanceData
)
```

Step 3.4.2: Call your API of Choice! (Parties)

```
▼ {retailParty: {...}} ⓘ  
  ▼ retailParty:  
    ► contactDet1: {phoneDet1: Array(2), emailDet1: Array(1), addressDet1: Array(2)}  
    ► employmentDet1: {employerName: "Mcdonalds", jobTitle: "Manager", occupationGroup: "Service", occupation: "Service"}  
      partyId: "Q010MDAwMDAx"  
    ▼ retailDemographic:  
      dateOfBirth: "1982-08-20"  
      ethnicGroup: "Chinese"  
      gender: "M"  
      maritalStatus: "Married"  
      nationality: "SG"  
      ► partyDoc: [{...}]  
      ► partyName: {fullName: "John Rivas", salutation: "Mr", alias: "John"}  
        residenceCountry: "SG"  
      ► __proto__: Object  
    ► __proto__: Object  
  ► __proto__: Object
```