MACHNET

Confidential Document
Do not share
v1.1, Feb 21, 2023

Guide for Bank Authentication Changes

Introduction

Starting January 2021, our bank verification partner has entered into data access agreements with Chase, Capital One, JP Morgan Chase, Wells Fargo, Bank of America, and US Bank to start migrating into new API-driven financial access channels. As a part of this agreement, the bank verification will need to be done using an OAuth integration. This is an industry-standard authentication protocol for data sharing, which will give users increased control and transparency while improving overall connection stability.

Overview of Bank Add Changes

In general, there are two ways to handle bank API: OAuth and non-OAuth. The existing bank add flow utilizes a non-OAuth flow and the user stays within the bank widget throughout the authentication and authorization process.

However, the OAuth flow requires the user to authenticate their account on their bank's website. This means that the user will begin the bank add process from the existing widget however, they will be temporarily redirected to their Bank's website to login and give permission after which, they are brought back to the bank add widget to complete the flow.

Banks Currently supporting OAuth Flow

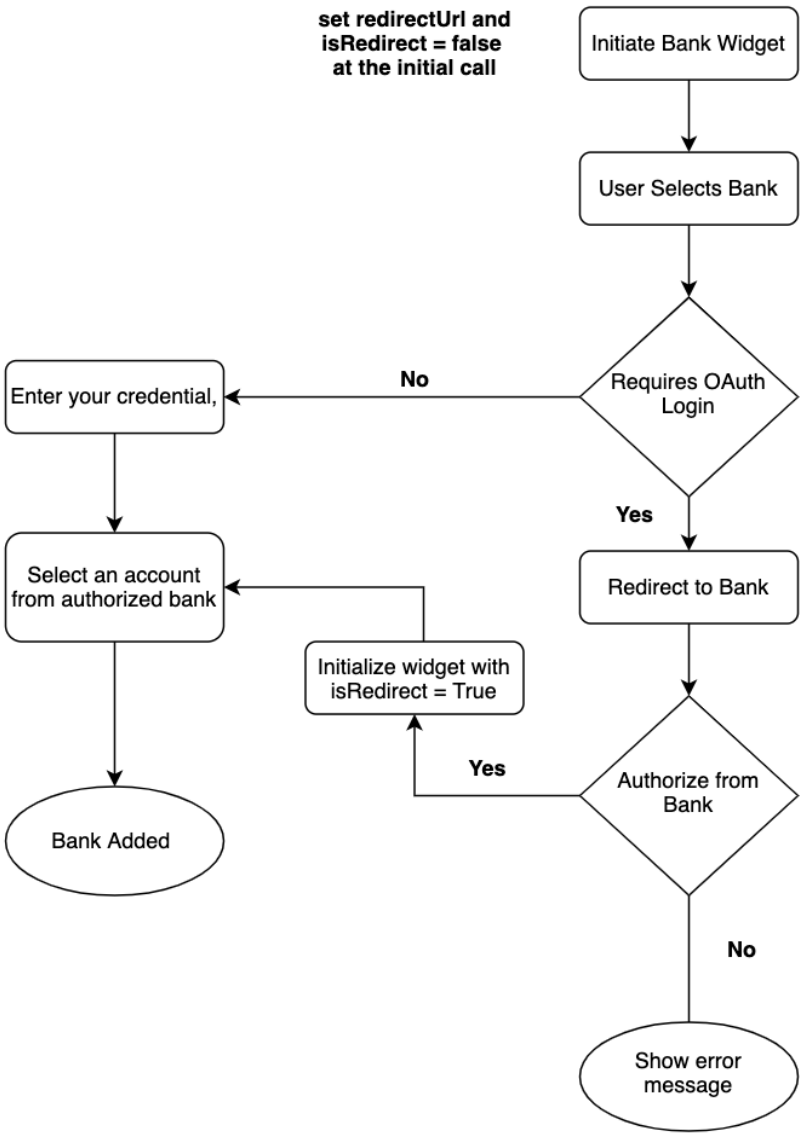
1. Chase
2. Capital One
3. Wells Fargo
4. Bank of America
5. USAA
6. U.S. Bank

7. Robinhood

Technical Changes

In order to support OAuth bank add flow, your changes will mostly be limited to your front end bank widget integration. Not all banks will support OAuth flow and you will only be able to tell whether a bank follows OAuth or Non-OAuth flow after the user has selected their bank. Therefore, we suggest you make the changes outlined below regardless of the banks you want to support.

Integration Flow Diagram



Loading Bank Widget in Web

Add new OAuth Bank

1. Open the bank widget
2. Select the bank that requires OAuth login
3. Users will be redirected to the bank website and as soon as the bank is authorized and user is redirected back to the widget iframe.
4. All redirections will be handled by the widget itself.
5. For oAuth bank relogin; it will be similar to the other banks. No changes required.

Loading Bank Widget in React Native Application

Add OAuth Bank Flow

1. Initialize the **`isRedirect`** to **`false`** in the state
   `const [isRedirect,setIsRedirect] = useState(false);`
2. Initialize the bank widget with following configuration

```
elementId: 'widget-root',
senderId: senderId,
width: '100%',
height: "height",
type: "bank",
stylesheet: "css path",
token: widgetToken,
locale: locale,
isRedirect: `${isRedirect}` // initialise to false by default
```

3. Open the bank widget
4. Select the bank that requires OAuth login
5. Users will be redirected to the bank website and as soon as the bank is authorized it will be redirected back to the app.
6. Once redirected back to the app we need to check the following condition on the **`onLoadEnd`** event of the Webview.

```
<WebView
    source={{html: <HTML View>}
    onMessage={onMessage}
    onLoadEnd = {(event) => {
      const nativeEvent = event.nativeEvent;
   //if you have set redirectUrl above, you need to compare below with redirectURL
      const isOrigin = nativeEvent.url.includes('sandbox.api.machpay.com');
      if(nativeEvent.url && isOrigin){
         setRedirect(true);
      }
    }}
/>
```

7. In **`onLoadEnd`**, if the origin is **sandbox.api.machpay.com** isRedirect state will be set to true.
8. Create a new onMessage function, this takes the event as a parameter . We should parse data that is passed from event.nativeEvent. This returns status and type. Type can be bank or card and status provides the widget event.

```
const onMessage = (e: EventObject) => {
   const {status, type} = JSON.parse(e.nativeEvent.data);
   console.log(status, type)

    // handle status and type accordingly.

}
```

```
elementId: 'widget-root',
senderId: senderId,
width: '100%',
height: “height”,
type: “bank”,
stylesheet: “css path”,
token: widgetToken,
locale: locale,
isRedirect: `${isRedirect}` // true, here is the difference
```

9. Now re-initializing a bank with **`isRedirect=true`** will open the widget again with the same authorized bank to complete the add bank process.
10. Finally, select the balance and type to complete the process

***Note**: There is no way of telling if you need to follow the OAuth flow until the user has selected their bank. Therefore, we suggest you to pass the redirectUrl whenever you are opening the bank widget.*

OAuth bank Relogin

1. When the bank status is unverified, save the unverified bank id in the state
   `const [bankId, setBankId] = useState(<Login required Bank ID>);`
2. Initialize the isRedirect to **`false`** in the state
   `const [isRedirect,setIsRedirect] = useState(false);`
3. Initialize bank widget with following configuration

```
elementId: 'widget-root',
senderId: senderId,
width: '100%',
height: “height”,
type: “bank”,
stylesheet: “css path”,
token: widgetToken,
locale: locale,
bankId: `${bankId}`
isRedirect: `${isRedirect}` // initialise to false by default
```

4. Open the bank widget
5. Select the bank that requires OAuth login
6. Users will be redirected to the bank website and as soon as the bank is authorized it will be redirected back to the app.
7. Once redirected back to the app we need to check the following condition on the **`onLoadEnd`** event of the Webview.

```
env widgetOrigin = 'sandbox.api.machpay.com';

<WebView
    source={{html: <HTML View>}
    onLoadEnd = {(event) => {
        const nativeEvent = event.nativeEvent;
//if you have set redirectUrl above, you need to compare below with redirectURL
        const isOrigin = nativeEvent.url.includes(widgetOrigin);

        if(nativeEvent.url && isOrigin){
            setRedirect(true);
        }
     }}
/>
```

8. In **`onLoadEnd`**, if the origin is sandbox.api.machpay.com **`isRedirect`** state will be set to true.

```
elementId: 'widget-root',
senderId: senderId,
width: '100%',
height: “height”,
type: “bank”,
stylesheet: “css path”,
token: widgetToken,
locale: locale,
redirectUrl: window.location.href,
isRedirect: `${isRedirect}` // true, here is the difference
```

9. Now re-initializing a bank with `isRedirect=true` will open the widget again with the same authorized bank to complete the add bank process.
10. Finally, select the balance and type to complete the process

***Note**: There is no way of telling if you need to follow the OAuth flow until the user has selected their bank. Therefore, we suggest you to pass the redirectUrl whenever you are opening the bank widget.*

OAuth Testing

Test Data

For Oauth Flow: Use **Platypus OAuth Bank** in bank widget. The dummy institution does not direct you to a real bank's site, but allows you to grant, deny, or simulate errors from the placeholder OAuth page instead. You can test OAuth on Sandbox even if it has not yet been enabled OAuth flows for your account.

For Non OAuth Flow you can use any other bank. There is no change in the Username and Password for Non OAuth flow.

Username: user_custom
Password: {}

Test Cases

Ensure that you are able to replicate the following scenarios after adding support for the redirect url/OAuth flow:

1. Users should be able to add Non OAuth Bank.
2. Users should be able to add OAuth Bank.
3. Users should be able to update credentials for Non OAuth banks.
4. Users should be able to update credentials for Oauth banks.