# Authenticate users

Connect to user wallets and authenticate your users using @stacks/connect.

---

Authentication is a fundamental part of many web applications, ensuring that users are who they claim to be and that their data is secure. With the Stacks blockchain, user authentication involves connecting to users' wallets and managing their sessions securely.

The `@stacks/connect` package provides the tools needed to integrate this functionality seamlessly into your web app.

In this guide, you will learn how to:

1. Install the `@stacks/connect` package.
2. Connect to a user's wallet.
3. Manage authentication state.
4. Access user data.

---

## Install the @stacks/connect package

Stacks Bitcoin

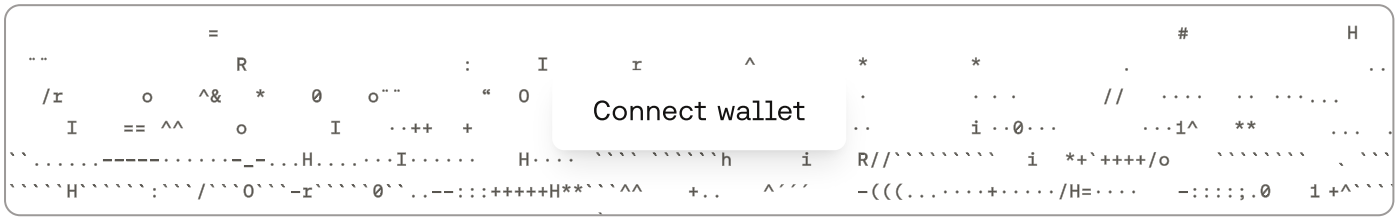
```
$ npm install @stacks/connect
```

## Connect to a user's wallet

After installing the `@stacks/connect` package, you can use the `connect` function to initiate a wallet connection. This will trigger a popup that allows users to select and connect their wallet.

```
import { connect } from '@stacks/connect';

async function authenticate() {
  const response = await connect();
  // response contains the user's addresses
}
```




The `connect` function stores the user's addresses in local storage by default, making it easy to persist the user's session across page reloads and browser sessions.

> You can customize the connection behavior by passing options to the `connect` function, such as forcing wallet selection or specifying default providers. See the advanced usage section for more details.

## Manage authentication state

You can manage the user's authentication state using the following functions:

```
import { connect, disconnect, isConnected } from '@stacks/connect';

// Check if user is connected
const authenticated = isConnected();

// Connect to wallet
await connect();

// Disconnect user
disconnect(); // clears local storage and selected wallet
```

## Access user data

Once connected, you can access the user's data using the `getLocalStorage` function or make specific requests using the `request` method:

```
import { getLocalStorage, request } from '@stacks/connect';

// Get stored user data
const userData = getLocalStorage();
// {
//   "addresses": {
//     "stx": [
//       { "address": "SP2MF04VAGYHGAZWGTEDW5VYCPDWWSY08Z1QFNDSN" },
//     ],
//     "btc": [
//       { "address": "bc1pp3ha248m0mnaevhp0txfxj5xaxmy03h0j7zuj2upg34mt7s7e32q
//     ]
//   }
// }

// Get detailed account information
const accounts = await request('stx_getAccounts');
// {
//   "addresses": [
//     {
//       "address": "SP2MF04VAGYHGAZWGTEDW5VYCPDWWSY08Z1QFNDSN",
//       "publicKey": "02d3331cbb9f72fe635e6f87c2cf1a13cd...",
//       "gaiaHubUrl": "https://hub.hiro.so",
//       "gaiaAppKey": "0488ade4040658015580000000dc81e3a5..."
//     }
//   ]
// }
```

> **Note**
>
> For a list of all available methods, see the reference page.

---

## Next steps

### Broadcast transactions

Learn how to sign and broadcast transactions.

### Deep dive into post-conditions

Learn how to build post-conditions to secure your smart contracts.

/-/