Web implementation with b.connect Web SDK

The Web SDK offers a way to simplify the integration of the b.connect button by handling its rendering and the redirection to the b.connect authentication service. It also offers a way to perform the authentication in a separate popup window and handles the associated interactions (such as passing messages between windows and closing the popup).
However, the Web SDK does not handle the Authorization Code that is returned from the OIDC authentication of the user. It is the service provider's responsibility to perform the necessary calls to the b.connect API to:

- Exchange the authorization code for an Access Token and ID Token
- Verify the signatures
- Verify the coherency
- (optional) Request b.connect CICO token

Those steps must be performed server side in accordance with the OpenID Service Provider specification and best practices. As those are OpenID connect standard processing, Service Provider should not have difficulties for using third party library or product that can handle those actions.

Web SDK initialization & configuration

The SDK attaches itself to the JavaScript `window` under the namespace "bconnect".
There are two methods to initialize the SDK and render the b.connect button:

- a declarative one, where configuration is provided through HTML tags;
- a programmatic one, which requires calling JavaScript functions directly but provides more flexibility

Additionally, when using the programmatic method, the b.connect sign-in redirection can be triggered programmatically.
The `initialize(config)` method should be called to provide the necessary configuration.
The call to `initialize(config)` returns an object with a `render(element, config)` and `signin()` methods.

Web SDK initialization

The Web SDK needs to be initialized with a configuration Service Provider specific to the Service Provider web site. This configuration manages both OIDC standard related parameters and b.connect Service Provider specific ones.
When using the programmatic method, the configuration is passed by calling the `initialize(config)` method. The call to `initialize(config)` return an object with a `render(element, config)` and `signin()` methods.

Web SDK parameters related to OIDC standard:
`base_url`: (Optional) The base URL on which to call b.connect OIDC authorization code endpoint /authorize?client_id=... If not provided the script will use its origin. To be used only if you encounter script origin detection issue.

b.connect - Communication, diffusion, reproduction, utilisation, exécution ou représentation de ce document interdits quel qu'en soit le support, sans l'accord de b.connect SAS

`client_id`: (Mandatory) The client id provided by b.connect after Service Provider application registration on b.connect administration portal.
`code_challenge`: (only if PKCE implementation) A Base64 encoded SHA-256 hashed code used to implement the PKCE verification. If you are not implementing PKCE, please let this field blank.
`nonce`: (Optional) String value used to associate a Client session with an ID Token, and to mitigate replay attacks, thus usage recommended.
`redirect_uri`: (Mandatory) The URL to redirect to after the authentication.
`scope`: (Optional) The scopes requested. If not provided Web SDK defaults to a single scope "openid".
`state`: (Recommended) A state Service Provider specific to the service provider, that will be returned as a parameter appended to the redirect_uri. To prevent attack and malicious user redirects, it is highly recommended for Service Provider to provide a value which must be different for each transactional call to b.connect authentication endpoint.

Web SDK parameters Service Provider specific to b.connect service:
`ux_mode`: (Optional) Two values are allowed: "redirect" and "popup'. Default to "redirect".
`auth_method`: (Optional) May be used to require a certain authentication method from the user on b.connect. Only supports the “active” option. When not set, b.connect decides on the authentication method by itself.
`callback`: (Optional) Only useful when the ux_mode is set to "popup". The name of a global function that will be called when the popup end event is sent from the child window (see popup mode).

B.connect Button Customization

The b.connect button rendered by the SDK can be customized by passing a configuration to the `render(element, config)` method:
`button_type`: (Optional) The following values are allowed: "standard", "icon”, "connect”, "create”. Defaults to "standard".



`icon` `standard` `connect` `create`

`button_size`: (Optional) The size of the button, one of “small”, “medium” or “large”.

Usage

Declarative Method

This method can be used for simple use cases and the use of the standard b.connect button.
After the DOM is loaded, the SDK will search for Service Provider specific configuration nodes and render a button. The nodes are searched based on the following values:

- "bconnect_onload" - The *id* of the element which will have the data attributes necessary for the SDK configuration.
- "bconnect_signin" - The *class name* to add on to an element where the b.connect button will be rendered.

On those nodes, the configuration parameters are passed by prefixing their name with `data-`.
The following HTML snippet illustrates how this works (with PKCE implementation case):

```
<html>
<body>
<h1>Demo</h1>
<script src="https://bconnect.net/a/sdk/web/v1/bconnect-sdk.min.js" async defer></script>
<script>
    function callback(msg) {
        document.getElementById("responsePre").innerHTML = `code = ${msg.code}`;
    }
</script>
<div id="buttonDiv">
    <div id="bconnect_onload"
        data-client_id="test"
        data-redirect_uri="http://my.site.com/auth/callback"
        data-scope="email given_name family_name openid cico"
        data-nonce="Tfq9ZgpbaibZ32Fvqp"
        data-state="jbp2BaVAcpaFVWA23fQA2fqPECB"
        data-code_challenge="E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM"
        data-ux_mode="popup"
        data-callback="callback">
    </div>
    <div class="bconnect_signin"
        data-button_type="standard"
        data-button_size="large">
    </div>
</div>
<pre id="responsePre"></pre>
</body>
</html>
```

Programmatic Method

This method is used to provide more flexibility. For example, it allows you to use a custom button or to control exactly when the signin redirection is triggered.
Use the `signin()` method to perform a sign-in from a custom handler:
The following HTML snippet illustrates how this works (with PKCE implementation case):

```
  <script src="https://bconnect.net/a/sdk/web/v1/bconnect-sdk.min.js" async></script>
  <script>
    onload = () => {
        document.getElementById("bconnect-signin-btn").onclick = event => {
            bconnect.initialize({
```

```
                client_id: "acme_client_id",
                redirect_uri: "https://my.site.com/auth/callback",
                scope: "openid email given_name family_name cico",
                nonce: "Tfq9ZgpbaibZ32Fvqp“,
                state: "jbp2BaVAcpaFVWA23fQA2fqPECB",
                code_challenge: "E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM"
            }).signin();
        }
    }
  </script>
</head>
<body>
    <button id="bconnect-signin-btn">Sign In with B.Connect</div>
</body>
```

Note: The `signin()` method return a promise that is only useful in the ux_mode === "popup" mode.

Use the `render(element, config)` method to render a button inside the provided element. A click on this button internally calls the `signin()` method.

```
  <script src="https://bconnect.net/a/sdk/web/v1/bconnect-sdk.min.js" async></script>
  <script>
    onload = () => {
      const bconSdk = bconnect.initialize({
        client_id: "acme_client_id",
        redirect_uri: "https://my.site.com/auth/callback",
        scope: "openid email given_name family_name cico",
        nonce: "Tfq9ZgpbaibZ32Fvqp“,
        state: "jbp2BaVAcpaFVWA23fQA2fqPECB",
        code_challenge: "E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM"
      });
      bconSdk.render(document.getElementById("buttonDiv"), {
        button_type: "standard",
        button_size: "large"
      });
    }
  </script>
</head>
<body>
    <button id="buttonDiv">  </div>
</body>
```

UX Mode

There are two modes that control how the b.connect authentication will be performed, from a user experience perspective. They are controlled by the value of the `ux_mode` parameter set upon initialization:

1. "redirect" – the browser is redirected to the b.connect authentication service directly
2. “popup” – the browser opens a popup window that loads the b.connect authentication service

b.connect - Communication, diffusion, reproduction, utilisation, exécution ou représentation de ce document interdits quel qu'en soit le support, sans l'accord de b.connect SAS

Redirect mode

In the redirect mode The SDK will simply perform a change of window location upon clicking the button. The user is redirected to the Service Provider website at the end of the authentication process, with the authorization code as a query parameter.

Popup mode

In the popup mode, the SDK will open a new child window and load the b.connect authentication service inside. This mode requires a callback to be set when initializing the SDK, so that the parent window can "receive" the authorization code. Internally, the callback is the one passed to the `signin(callback)` method (when using the declarative method or programmatic method with `render(element, config)`).

```
        <script src="https://bconnect.net/a/sdk/web/v1/bconnect-sdk.min.js" async></script>
        <script>
            window.onload = function () {
                const bconSdk = window.bconnect.initialize({
                    client_id: 'acme_client_id',
                    redirect_uri: "https://my.site.com/auth/callback",
                    scope: "openid email given_name family_name cico",
                    nonce: "Tfq9ZgpbaibZ32Fvqp“,
                    state: "jbp2BaVAcpaFVWA23fQA2fqPECB",
                    code_challenge: "E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM",
                    ux_mode: 'popup',
                    callback: msg => {
                        console.log(msg.code)
                    }
                });

                bconSdk.render(document.getElementById("buttonDiv"));
            }
        </script>
        <div id="buttonDiv"></div>
      </body>
    </html>
```

Additionally, in order to close the child window and return the authorization code to the callback (or the promise of `signin`), a call to `bconnect.popup.end()` should be performed in the Service Provider page loaded from the the `redirect_uri`, in the child window.

```
        <script src="https://bconnect.net/a/sdk/web/v1/bconnect-sdk.min.js" async></script>
        <script>
            window.onload = function () {
                bconnect.popup.end();
            }
        </script>
        <div id="buttonDiv"></div>
      </body>
    </html>
```

This will send a message to the parent window and close the child window.

Thank you!

Technical support contact:

Lionel Lorimier

CTO b.connect

lionel.lorimier@bconnect.net

+33 6 70 24 49 87

b.connect - Communication, diffusion, reproduction, utilisation, exécution ou représentation de ce document interdits quel qu'en soit le support, sans l'accord de b.connect SAS



Copyright © 2025 b.connect
All rights reserved.
b.connect and its logo are trademarks of b.connect.