JS Connector (xID)

xID Connect is a Javascript helper library that offers simple integration of the xID service for OIDC Clients. xID Connect supports several combinations of integration methods and message flows, each with different user experiences and considerations. The chosen integration method may cause a window, redirect, iframe or inline dialog to appear showing any dialogs relevant for the xID session. Depending on the chosen flow, HTTP endpoints at the back-end of the OIDC Client must be implemented to perform appropriate Token and Userinfo (TINFO) requests to retrieve data about the authenticated user.

The documentation of xID Connect is divided in the following sections. See also full source code examles on GitHub for the use of xID Connect.

Front-end implementation

1. Load xID Connect

Add the following HTML snippet to your OIDC Client front-end :

Example loading xID Connect library

```
<script async defer src="https://bankidapis.no/js/bid-xid_connect.bundle.min.js"></script>
```

This will make available an object `XID` attached to the global `window` object: `window.XID`

2. Initialise xID Connect with parameters

In your OIDC Client front-end, listen for the xid-loaded event, and call the initialisation API. Parameters are discussed below.

Example of xid-loaded handler

```
function onXIDLoaded() {
    // Initialise xID with required parameters
        window.XID.doInit ({
            // Merchant given client ID on the xID service
                client_id: 'myApplication',
                // The resource scopes merchant want to gain access to for the user
                scope: 'openid address',
            // Implementation method, open the session in a new window
                method: 'window',
            // Merchant backend endpoints for performing token/userinfo calls.
                token_endpoint: 'https://example.com/oauth2/token',
                userinfo_endpoint: 'https://example.com/oauth2/userinfo',
        });
}

document.body.addEventListener( 'xid-loaded', onXIDLoaded, false );
```

Parameters

The most important parameters are as follows, some of which are identical to parameters to the Authorize endpoint of the REST API.

Parameter	Description	Default
`client_id`	A string specifying the ID given when registering the OIDC Client in question with the OIDC Provider fromBankID.	
`scope`	A string of resource types (dataset) belonging to the user to request access to. Each scope / resource type must be separated by space.	`'openid'`
`method`	The chosen xID Connect integration method as further described below.	`'window'`
`response_type`	The chosen authentication response type govern message flow as further describe below.	`'code'`

Depending on `response_type` the following parameters may be important:

Parameter	Description	Default
`token_endpoint`	Absolute URL to HTTP endpoint on OIDC Client back-end to retrieve access/ID token in exchange for authorization code (if using `code` response_type).	`'/oauth/token'`
`userinfo_endpoint`	Absolute URL to HTTP endpoint on OIDC Client back-end to retrieve user information using access token.	`'/oauth/userinfo'`

Other relevant, but optional, parameters include the following, some of which are identical to parameters to the Authorize endpoint of the REST API.

Parameter	Description	Default
`redirect_uri`	Absolute URL to the HTTP endpoint on OIDC Client back-end receiving the authentication response from xID.	`(xIDs internal callback handler)`
`response_mode`	Set the format used when returning parameters from the Authorization Endpoint via `redirect_uri`	`'query'`
`noStepup`	Set to `true` to disallow step-up from xID to the BankID IDP.	`false`
`user_profile`	Set user info such as NNIN or phone number / birthdate if required for step-up to the BankID IDP.	`''`

The full list of parameter list is found in the JS API Reference.

Integration methods

There are several ways to integrate xID in your application, each with different user experiences and considerations. However, the xID functionality and appearance remain the same regardless of the integration method.

Method	Description
`window`	A popular (and default) implementation choice is `window`. When xID Connect is triggered it will open the OIDC session in a new window (pop-up). Note that an user action should trigger this session as otherwise pop-up blockers might block the window.
`inline`	A DOM element ID can be provided to `XID.doConnect()` to host an iframe which opens the OIDC session inline in your application. There is also a special integration which will display a modal dialogue overlaying your application. This method is useful when triggering xID Connect upon loading the application to avoid pop-up blockers. This method is described here XXXX.
`redirect` *(not fully tested)*	xID Connect can redirect the user away from your application to a separate web page for the entire xID login session before returning to a given callback URL. When using this method its important to set `redirect_uri` to point to a HTTP endpoint on the OIDC Client back-end which can receive authorisation code / tokens.

Message flows (response_type)

xID Connect supports each of thefmessage flows supported by the OIDC Provider as governed by the `response_type` parameter

> Currently, only the authorization code flow is fully functional with xID Connect. Support for implicit/hybrid is coming.

Authorisation code flow

`response_type: 'code'`

This is the most secure and recommended method as the `client_secret` is not leaked into the client application. It is also the default method used with xID Connect.

A time limited one-time code, authorisation code, is granted to `redirect_uri` which in turn can be used to fetch an access token. The access token can in turn be used to fetch user information.

1. `code` is returned as parameter in response to `redirect_uri`: `https://example.com/callback?code=oMsCeLvIaQm6bTrgtp7`
2. The `code` is exchanged for Access Token and ID Token by sending a HTTP POST request to the xID IDP token endpoint as explained here
3. Consented user information is retrieved by sending a HTTP GET request to the xID IDP userinfo endpoint as explained here

xID Connect automatically handles the client code for the token exchange and provides API for getting user information for the current session via `window.XID.doGetUserInfo`

Never embed or send the `client_secret` to the client application.

Implicit flow

`response_type: 'token'`

This flow returns an Access Token directly in the response to `redirect_uri`, bypassing the client_secret/code exchange from the Authorisation code flow, allowing the request userinfo directly. Meant for OIDC Clients without backend.

Hybrid flow

`response_type: 'hybrid'`

Combination of code grant and implicit flow. Authorisation code and tokens can be delivered to `redirect_uri`.

> For increased security in a production environment, it is highly encouraged to use `nonce` and `state` parameters when interacting with the xID service.

3. Start xID login

To start the xID login process call the `window.XID.doConnect()` function giving a callback function.

The callback function is provided with the returning accessToken and any error messages if something has gone wrong. For example, by user cancellation.

Example calling doConnect and logging accessToken

```
// Start xID login providing a callback
window.XID.doConnect( function (err, accessToken) {
        if ( err ) {
            // handle xID connect error
    }
    else {
        // client is now connected, store ‘accessToken’ if needed
        doGetXidUserInfo();
    }
});
```

4. Get user info

A common callback action is to fetch user information as the access token stored in the current xID Connect session.

xID Connect provides the API function `window.XID.doGetUserInfo()` for this task.

Example calling doGetUserInfo() in a callback for doConnect()

```
function doGetUserInfo() {
        window.XID.doGetUserInfo( function (err, user) {
                if ( err ){
                        // handle error
                }
                else {
                        // handle user
                }
        });
}
```

Back-end implementation

When using authorisation code flow or hybrid flow with xID Connect the library expects the OIDC Client to implement back-end functions to handle OAUTH requests to receive tokens and user info from xID IDP. The OIDC Client back-end is needed in order to safely store the required parameter `client_secret` away from front-end components of the OIDC Client. xID Connect will by default automatically call these HTTP endpoints in this mode.

If you use the implicit flow, there is no need for any OIDC Client back-end.

Token endpoint

When a code is received, xID Connect will call the token endpoint to receive an `access_token`

xID Connect will store this token in the current client session. For increased security, you may avoid sending this token back to xID Connect request.

URL	As specified in configuration parameter `token_endpoint`	
Request mode	`POST` with parameters as `application/x-www-form-urlencoded` data	
Request parameters	`grant_type`	Grant type is always `authorization_code`
	`code`	Value from response of the foregoing Authorize request
	`redirect_uri`	Redirect URI used in the foregoing Authorize request
	`client_id`	Not supported since the OIDC clients must always authenticate

xID IDP Response /oauth/token	**JSON**
	`{`
	`  access_token: "654fe6f11ad61ceb1697d643b5fc59",`
	`  token_type: "Bearer",`
	`  expires_in: 3600,`
	`  scope: "openid phone address",`
	`  id_token: "...."  // JWT`
	`}`	

For documentation on the corresponding response to xID IDP see Token

Userinfo endpoint

Using the access token (which also could be stored on the server) to access user information xID IDP.

URL	As specified in configuration parameter `userinfo_endpoint`
Request mode	`POST` with parameters as `application/x-www-form-urlencoded` data

Request parameters	`access_token`	Access token to be used as authorization to access xID IDP endpoint
	`token_type`	How access token shall be passed to xID OIDC endpoint (

xID IDO response /oauth/userinfo	**JSON**
	
	`{`
	` "sub": "9578-6000-4-127698",`
	` "iss": "https://preprod.bankidapis.no",`
	` "iat": 1485866449,`
	` "exp": 1485870048,`
	` "preferred_username": "Testesen, Test",`
	` "name": "Testesen, Test",`
	` "given_name": "Test",`
	` "family_name": "Testesen",`
	` "birthdate": "1980-03-09",`
	` "nnin": "09038000010"`
	`}`	

For documentation on the corresponding response to xID IDP see Userinfo (TINFO)

JS API Reference

Configuration

Configuration is set by passing an object to XID.doInit() or the config parameter of XID.doConnect(). The following parameters are supported, some of which are identical to parameters to the Authorize endpoint of the REST API.

Parameter	Description	Default
`client_id`	A string specifying the client ID given when registering to the xID central service.	
`scope`	A string of resource types (dataset) belonging to the user to request access to. Each scope / resource type must be separated by space.	`'openid'`
`method`	The chosen xID Connect integration method, explained here.	`'window'`
`response_type`	The chosen authentication response type, explained here. Ex. `'code'` or `'token'`	`'code'`
`response_mode`	Set the format used when returning parameters from the Authorization Endpoint via `redirect_uri`	`'query'`
`token_endpoint`	Absolute URL to HTTP endpoint on merchant server-side to retrieve access/ID token in exchange for authorization code (if using `code` response_type).	`'/oauth/token'`
`userinfo_endpoint`	Absolute URL to HTTP endpoint on merchant server-side to retrieve user information using access token.	`'/oauth/userinfo'`
`redirect_uri`	HTTP endpoint receiving the authentication response from xID	
`oauth_url`	Absolute URL to the xID IDP OAUTH endpoint. Ex. https://preview.bankidapis.no/oauth Likely never applicable to change.	`(default OAUTH endpoint for the xID service)`
`client_type`	Preset the OIDC client type, one of `XID`, `BID`, `BIM` or (empty).	`'XID'`
`forceClientPrompt`	Set to `true` to force the user to always confirm the xID client usage Corresponds with the login hint `forceconfirm`	`false`
`skipClientPrompt`	Set to `true` to always skip the xID client usage confirmation Corresponds to the login hint `directConsent`	`false`

`noStepup`	Set to `true` to disallow step-up to the BankID IDP.	`false`
`user_profile`	Set user info such as NNIN or phone number / birthdate if required for step-up to the BankID IDP.	`''`
`state`	Increase security towards cross-site request forgery by verifying this value in the requests and responses	`'untouched'`
`nonce`	Provide a nonce value for securing the integrity of the `id_token`	
`grant_type`	This field always contains the value authorization_code, as defined in the OAuth 2.0 specification.	`'authorization_code'`

Methods

XID.doInit(config{...})

Sets the configuration for the current session towards xID IDP.

Parameter	Description	Default	Required
`config`	See Configuration above		x

XID.doConnect (callback(err, data), [config{..}, onActionCallback, inlineOnLoadCallback, inlineModalWindow, inlineElementID])

Starts xID login session with the given configuration set with `doInit()`.

If an access token is already active, the callback is directly called without calling xID IDP. To avoid this, call XID.doLogout() and try again.

Parameter	Description	Default	Required
`callback`	Function to handle response from Authorize call. Arguments are: • err - error messages, if any • data - returned object with accessToken		x
`config`	Config parameters can be provided which will override session parameters for this session only.	`{}`	
`onActionCallback`	Called for each action in xID iframe	`null`	
`inlineOnLoadCallback`	Called onload for injected iframe	`null`	
`inlineModalWindow`	Set to true to activate special inline login modal window. Only active when used with `inline` integration method.	`false`	
`inlineElementID`	ID of DOM element to inject xID login iframe into.	`null`	

XID.doGetUserInfo (callback(err, user), [accessToken, tokenType, responseType])

Parameter	Description	Default	Required
`callback`	Function to handle response from userinfo call. Arguments are: • err - error messages, if any • user - user object with data		x
`accessToken`	Optionally provide own accessToken.	`null`	
`tokenType`	Optionally provide own tokenType.	`null`	
`responseType`	Set to `token` if userInfo request should go directly to oauth endpoint instead of through the middleware.	`'code'`	

XID.doLogout()

Removes the current sessions access token and resets xID Connector state.

XID.doReset()

Removes the local xID cookies for the current user. (advanced functionality)

Events

Name	Description
`xid-loaded`	Triggered on `document.body` element when xID Connect is loaded and ready to receive API calls