Message flow details

The OIDC Provider from BankID supports each of the flows (grant types) defined by the OIDC/OAuth2 standards:

- Authorization code flow
- Implicit flow
- Hybrid flow

All of these flows concern authentication of end-users followed by authorization of access to VAS services. The following flow from the OAuth2 standard concerning authorization of access to VAS services directly from an OIDC Client without involving an end-user is also supported:

- Client credential flow

The below figure provides an elaborated understanding of the message flow by showing an example of an hybrid flow. The following applies for this particular example:

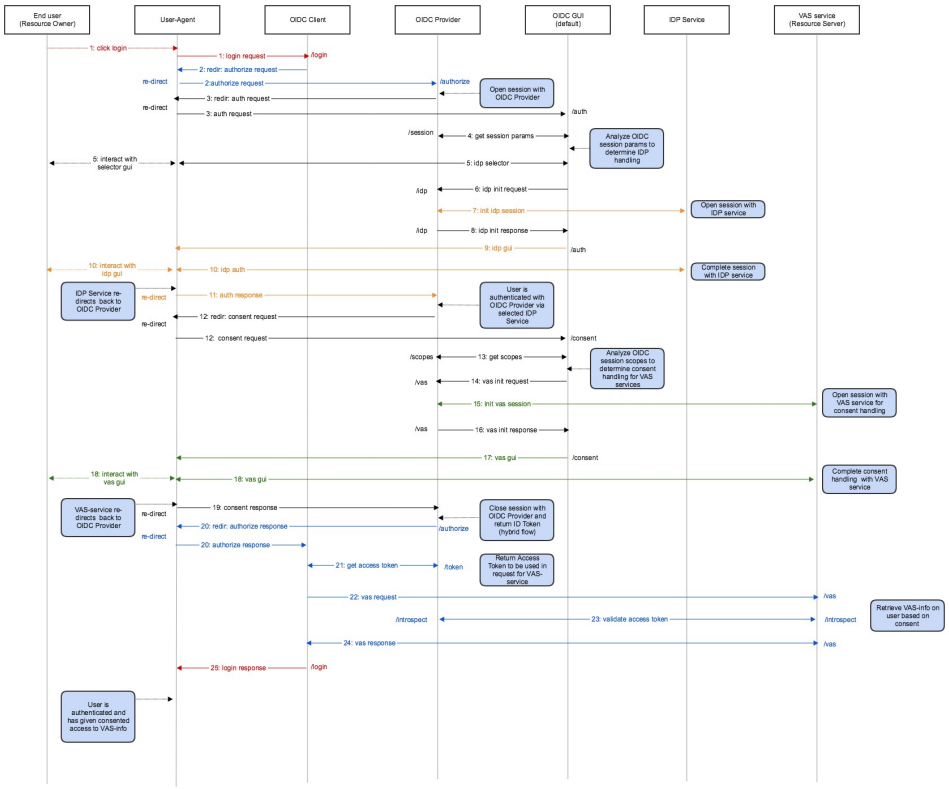
- The OIDC Client uses the default GUI component of the OIDC Provider
- The OIDC Client leaves selection of IDP Option to the end-user in a selector dialogue provided by the OIDC Provider
- The OIDC Client requests access to scopes/claims that is available via one of the supported VAS-services.

The following color coding is used:

- Red corresponds to application-specific flows for the OIDC Client
- Blue corresponds to standardized flows over the REST API according to OIDC/OAuth2 standards.
- Black corresponds to specific flows for the OIDC Provider from BankID allowing OIDC Clients to customize GUI experience
- Yellow corresponds to specific flows for the designated IDP.
- Green corresponds to specific flows for the designated VAS service (eg. the VAS Service).

An OIDC Client does by-default only involves standardized flows (blue color) over the REST API with the OIDC Provider. The exception is if the OIDC Client wants to customize GUI handling. Any custom GUI component must integrate with another REST API (black color) specific for the OIDC Provider from BankID. A custom GUI component must take care of proper integration with each of the supported IDP options (yellow color) and also any involved VAS service (green color).

Note that the below figure does not reflect the use of any JavaScript Connector to assist the OIDC Client with integration with the OIDC Provider. Using a JS Connector will save the OIDC Client from handling most of the front-end logic (blue color) associated with the message flow, thus simplyfing integration work.



The following actors are involved in the message flow for the shown example:

- End-user - The user owns resources that the OIDC Client requests access to. Some resources may require an explicit consent from the user before the OIDC Client is granted access.
- User-Agent - A browser, or a browser window in an application, allowing the user to navigate the OIDC Client and interact with the other parties involved via re-direction of requests through the User-Agent.
- OIDC Client - The application that needs to assure the identity of the end-user and request access to various resources.
- OIDC Provider - The platform from BankID that provides an OpenID Connect / OAuth2 interface in front of a range of Identity Providers (IDPs) and Value Added Services (VAS).
- OIDC GUI - A service that is responsible for all GUI handling associated with OIDC Provider. The OIDC Provider comes with a default GUI service that is used unless it is overridden by the OIDC Client.
- IDP Service - A designated IDP selected by the end-user among all IDP options supported by the OIDC Provider (in other cases the OIDC Client may select the designated IDP option).
- VAS Service - A service that returns info on the end-user beyond what is returned directly in the ID Token by the OIDC Provider itself. The service performs consent handling before actual data is retrieved subsequently.

The example flow consists of the following steps, some of which are optional.

1. The end-user navigates the OIDC Client via the User-Agent and selects a login action.
2. The OIDC Client redirects the User-Agent to the OIDC Provider with a standardized Authorize request. The OIDC Client will regain control in step 16 at a designated redirectURL.
3. The OIDC Provider opens a session and redirects the User-Agent with an authentication request to the designated URL for GUI handling, which in this case corresponse to the default GUI component. Parameters to the request identifies the session in progress.
4. The GUI component requests parameters from the OIDC Provider for the session in progress to determine if the OIDC Client has pre-selected a specific IDP or if a selector dialog should be shown to the end-user
5. A IDP selector dialog is shown to the end-user.
6. The GUI component sends an init request to the OIDC Provider for the designated IDP Service
7. The OIDC Provider sends a corresponding init request to the designated IDP Service which opens a session and responds with the necessary parameters to lauch the GUI for the designated IDP.
8. The OIDC Provider returns the necessary parameters to the GUI component
9. The GUI component delivers the GUI for the selected IDP to the User-Agent
10. The end-user interacts with the IDP GUI, which in turn communicates with the IDP Service. Note that the OIDC Client is kept out of this dialogue to prevent any replay attack from any malicious OIDC Client.
11. After completing the session with the IDP Service, the User-Agent is redirected back to the OIDC Provider with an authentication response. The end-user is now authenticated.
12. The OIDC Provider redirects the User-Agent to the GUI component for consent handling
13. The GUI component requests from the OIDC Provider the set of scopes and claims that has been requested by the OIDC Client for the session in progress. For the shown example supplied scopes as associated with the VAS Service.
14. The GUI component sends an init request to the OIDC Provider for the VAS service
15. The OIDC Provider sends a corresponding init request to the VAS Service which opens a session for consent handling and responds with the necessary parameters to lauch the GUI for the VAS Service.
16. The OIDC Provider returns the necessary parameters to the GUI component
17. The GUI component delivers the GUI for the VAS Service to the User-Agent
18. The end-user interacts with the VAS GUI, which in turn communicates with the VAS Service.
19. The User-Agent is redirected back to the OIDC Provider after consent handling for the VAS Service. The ID Token for the authenticated user is now being composed according to the reqested scopes.
20. The ID Token is returned to the OIDC Client in an standardized Authorize response via a redirect of the User-Agent, corresponding to a hybrid OAuth2 flow. An intermediate authorization code is also returned in this step that is used in the next step to request any Access Token.
21. The OIDC Client sends a standardized Token request to exchange the authorization code from the previous step for an Access Token with the OIDC Provider. For the shown example an Access Token is returned by the OIDC Provider that grants access to the VAS Service over an endpoint in the correspoding Resource Server in the next step. Note that the token request does not go through the User-Agent for security reasons.
22. The OIDC Client sends the Access Token from the previous step 21 in a request to the VAS-specific Resource Serverto get access to the VAS Service. The Access Token is a bearer token that provides proof of authorization by the end-user.
23. The VAS service validates the Access Voken via a standardized Introspect request.
24. After sucessfull validation of the Access Token, the VAS info in question is returned to the OIDC Client in a VAS-specific response.
25. The OIDC Clients returns to the User-Agent a page showing the reponse of the login request along with any additional VAS-specific information that was retrieved.