OwnID

Security and Data Privacy Overview

V1.2

Overview

OwnID is a web based solution that allows users to use their phone to login and register instead of a password.

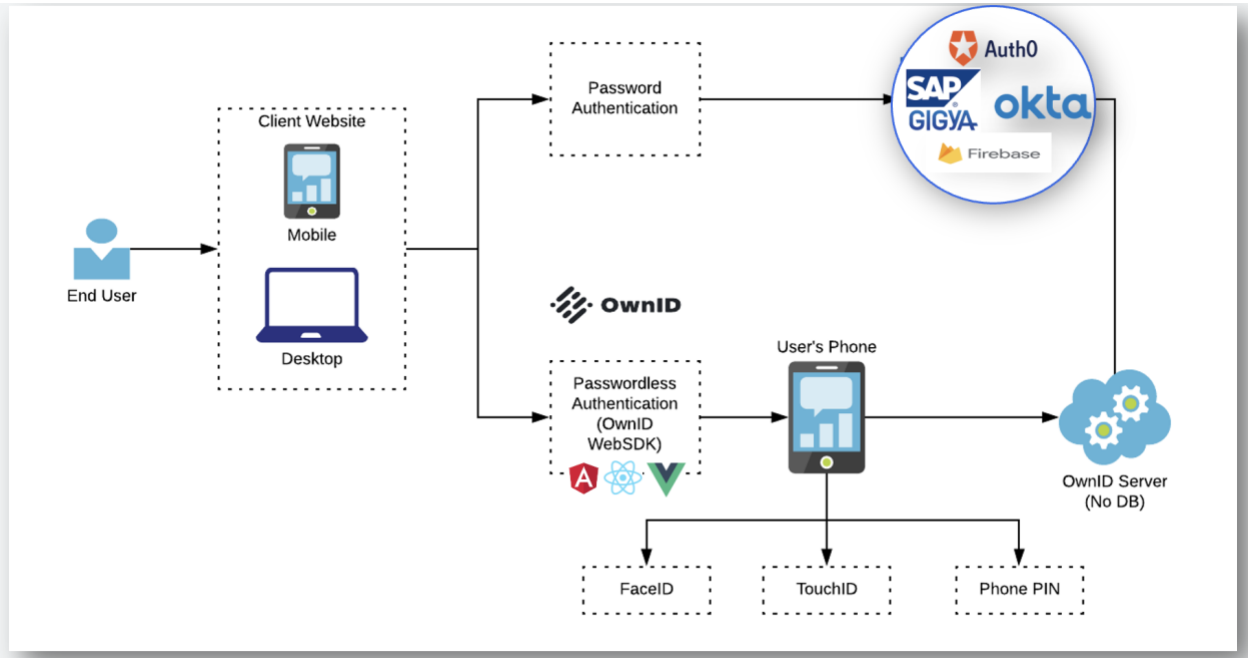
Important points to know:

- OwnID does not collect any data on the users or customers.
- OwnID communicates with the Identity Management System that the website is using.
- OwnID has out-of-the-box integration with the major CIAM solutions like SAP Customer Data Cloud (Gigya), Firebase and Amazon Cognito that takes 30 minutes and involves copying & pasting a JS code snippet.
- On desktop, a QR code is shown and the user scans it in order to use the phone to login.

How it works

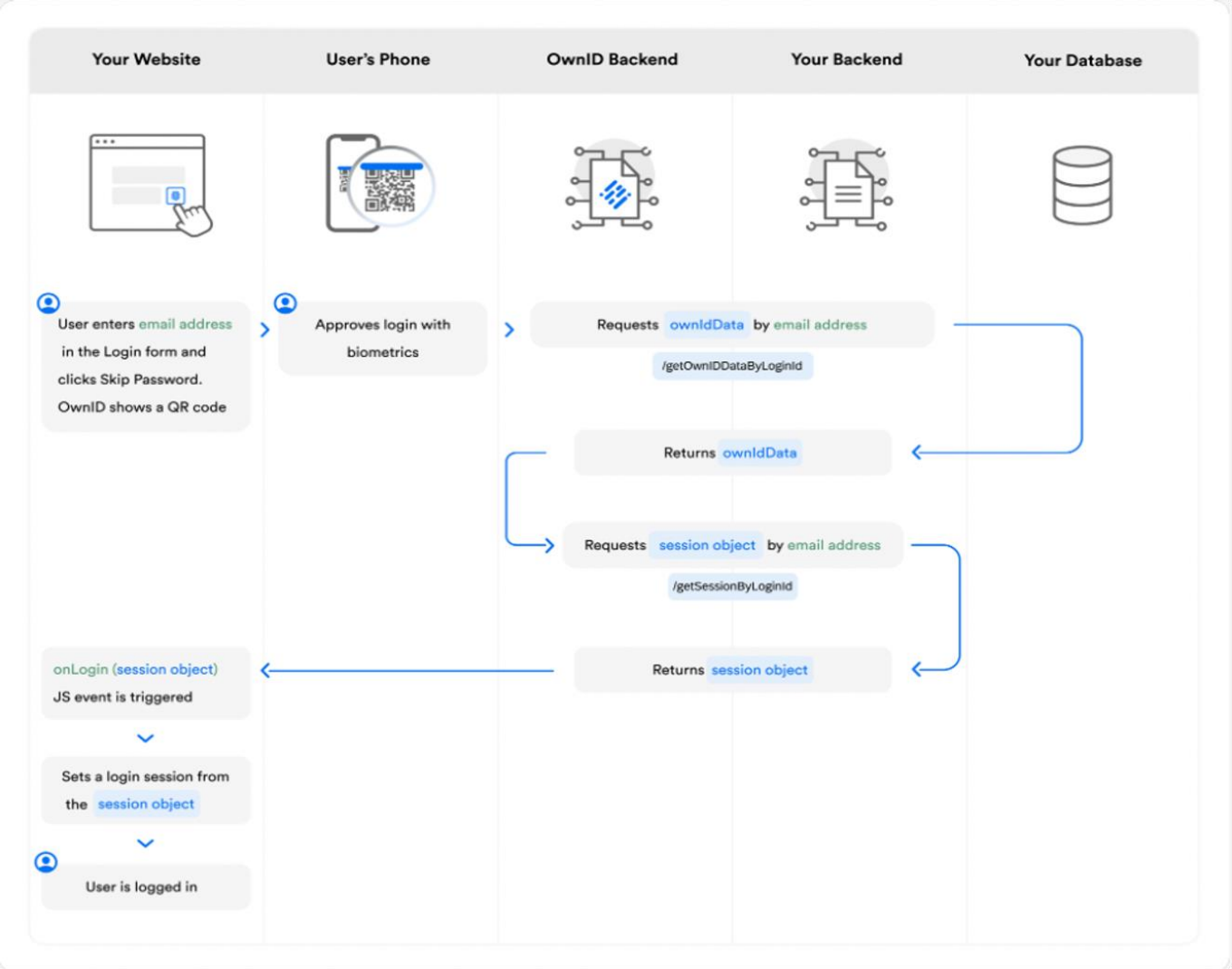
The components involved in the OwnID flow are:

- Client's website/ mobile app integrated with OwnID widget
- User's mobile phone executing OwnID WebApp
- Identity Management System that the customer is using (e.g: SAP CDC, Auth0, Firebase, in-house)

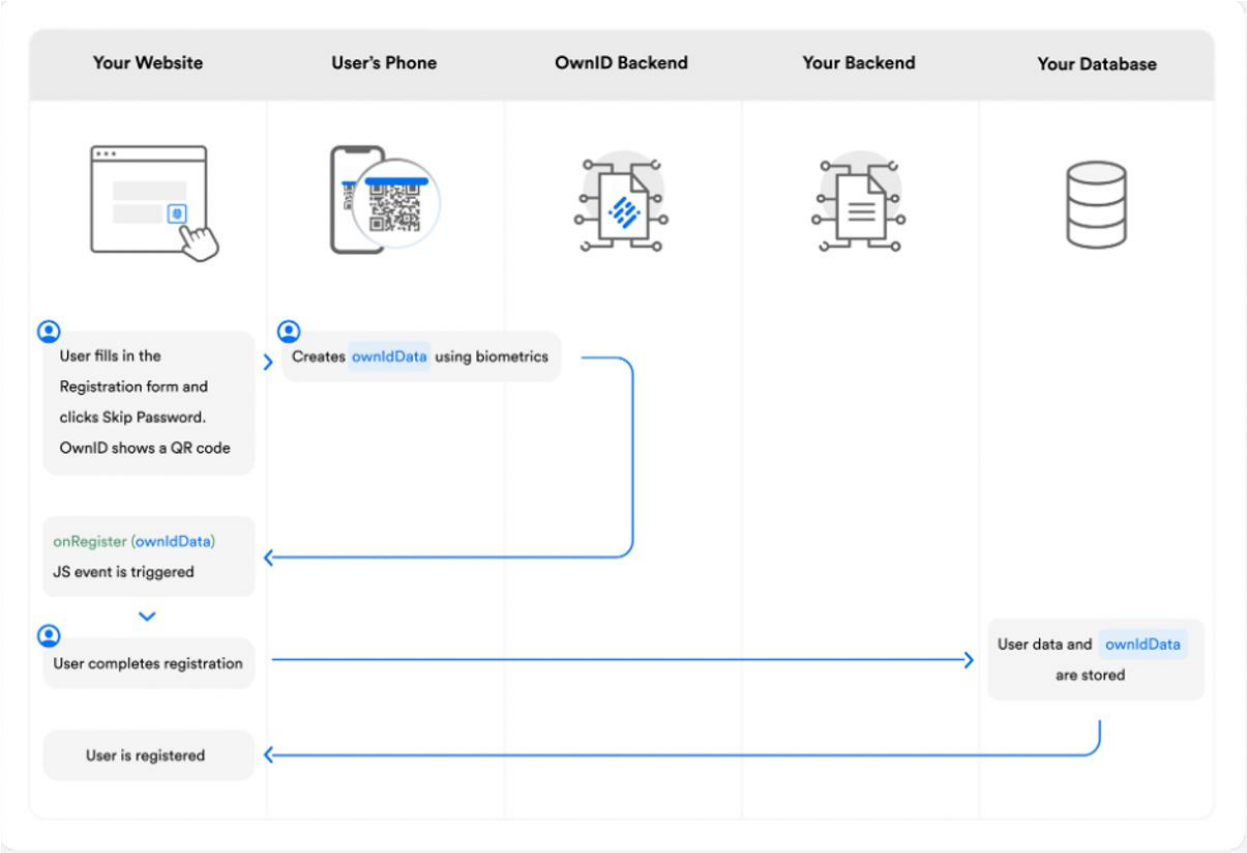


1. OwnID widget sends an authentication request to the phone by scanning a QR code (on desktop) or clicking a “Skip Password” link (on mobile)
2. WebApp is initialized on the phone. It looks up credentials which are stored on the phone (FIDO2 by default)
3. WebApp communicates with the OwnID server to authenticate the user in the Identity Management System
4. OwnID widget gets a login token from the OwnID server

Login Flow - High Level



Registration Flow - High Level



Data Privacy

The user's phone is the method of login. Therefore, the phone is where the credentials are stored. Credentials can be stored in:

- By default, Biometrics using FIDO2 (depending on device support)

OwnID is replacing the user's password with credentials that are created and stored on the user's mobile phone. The public part of the credentials is then sent over to the server to be stored in the Identity Management System.

OwnID does not store any user data.

We do not collect or process Personal Data when users sign up and login.

When users sign up or login, they can use their phone's biometrics instead of a password by clicking on a "Skip Password" button.

When users click on "Skip Password" from their phone, they are requested to use their phone unlock mechanism (e.g. Face ID) in order to authenticate.

On a desktop, a QR code is presented. Users scan the code with their phone and then authenticate with their phone biometrics. After that, the flow continues on the desktop.

Application-Level Security

Data Encryption

We force all network exchange between our SDK and servers to take place over TLS. Our certificates are signed with SHA-256 ECDSA.

Client and Server Hardening

Exposed server endpoints are recurrently tested for vulnerabilities using multiple types of scanning software as well as manual testing. Request-handling code paths have frequent user re-authorization checks, payload size restrictions, rate limiting where appropriate, and other request verification techniques. All requests are logged and made searchable to operations staff.

Client code utilizes multiple techniques to ensure that using OwnID's WebSDK and Console application is safe and that requests are authentic, including

- IFRAME sandboxing
- XSS and CSRF protection

Monitoring and Testing

We use internal and third-party systems to monitor the confidentiality, integrity, and availability of our platform. If an incident occurs, a team of engineers is alerted immediately.

Additional Policies

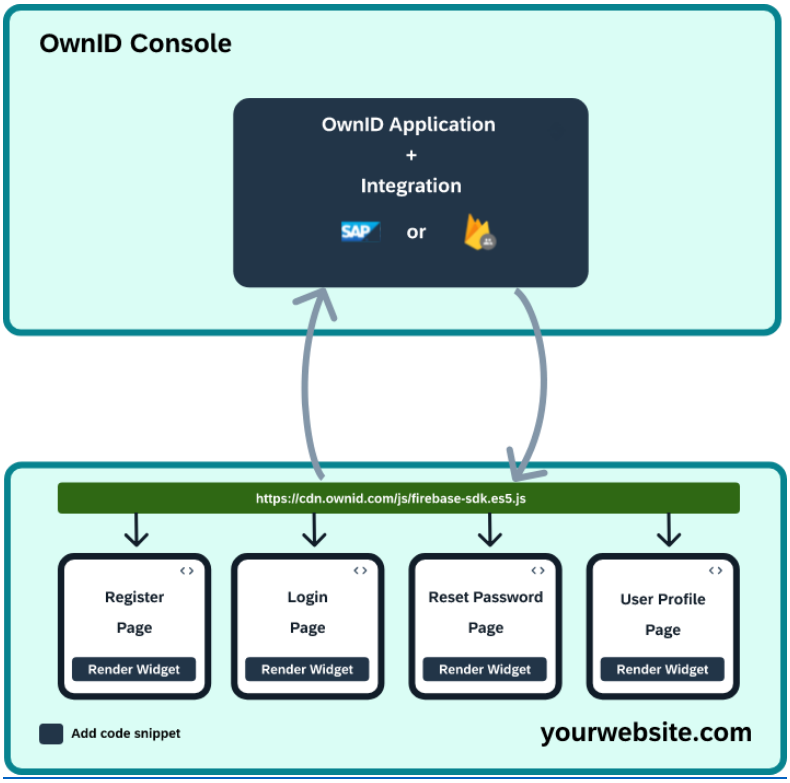
Code Reviews and Production Release

All changes to source code destined for production systems are subject to a pre-commit code review by a qualified engineering peer that includes security, performance, and potential-for-abuse analysis.

Prior to updating production services, all contributors to the updated software version are required to attest that their changes are working as intended on staging servers.

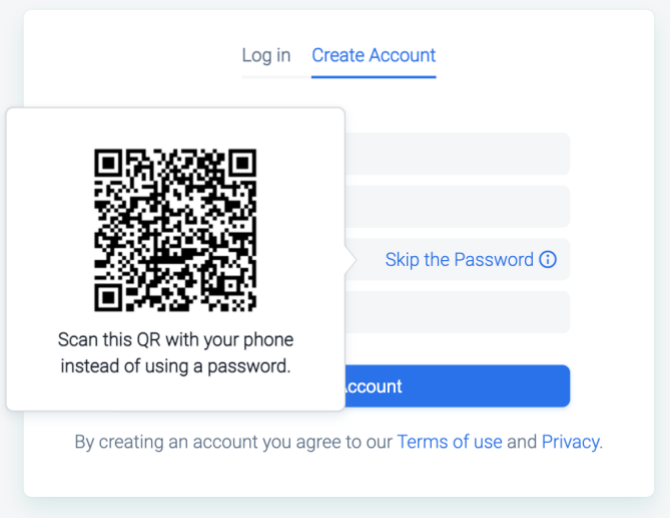
Integration & setup

1. A customer's website developer registers on OwnID Console (console.ownid.com) and is presented with a UI to create a new integration. This effectively creates a DB record on the OwnID's server that defines a new tenant configuration.
2. The developer enters the identity management credentials in order for the OwnID server to talk with the identity management system. This configuration is stored on the tenant's DB record.
3. Finally, the developer gets instructions on how to integrate the OwnID widget SDK into the website's authentication pages - a copy & paste of javascript snippets that include the OwnID web SDK.



Registration flow

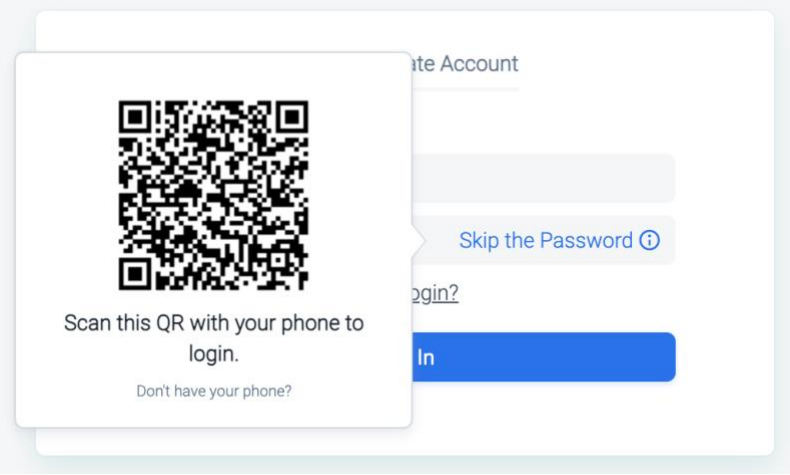
1. An end user navigates to the customer's website and goes to the registration page. A "Skip Password" link in the password field of the registration form gives the user the option of using their mobile phone to register without a password.
2. A click on the link initiates a call to OwnID server that returns a response with the Webapp URL, context and nonce:
    a. Webapp URL – A URL pointing to the location where the OwnID web app is hosted (app.ownid.com)
    b. Context – A UUID that should be set in every request to the server so cached parameters that are part of the register/login can be looked up
    c. Nonce – A parameter for the server to identify the client
3. On desktop, the data received is used to generate a QR Code that the user is instructed to scan with a phone in order to execute the web app.
   On mobile, the data received is used to execute the web app in a new tab.
4. Once the web app is executed, OwnID web SDK starts a polling process to the OwnID server to get the status of the execution in order to know when the user has finished registration.
5. The web app asks the user to identify with biometrics by using the browser's native WebAuthN API (FIDO2). The WebAuthN structure is sent to the OwnID server with the context and the web app closes.
6. The web SDK's polling process gets the WebAuthN structure, stores it under the user's profile and completes the registration process with the identity management system.



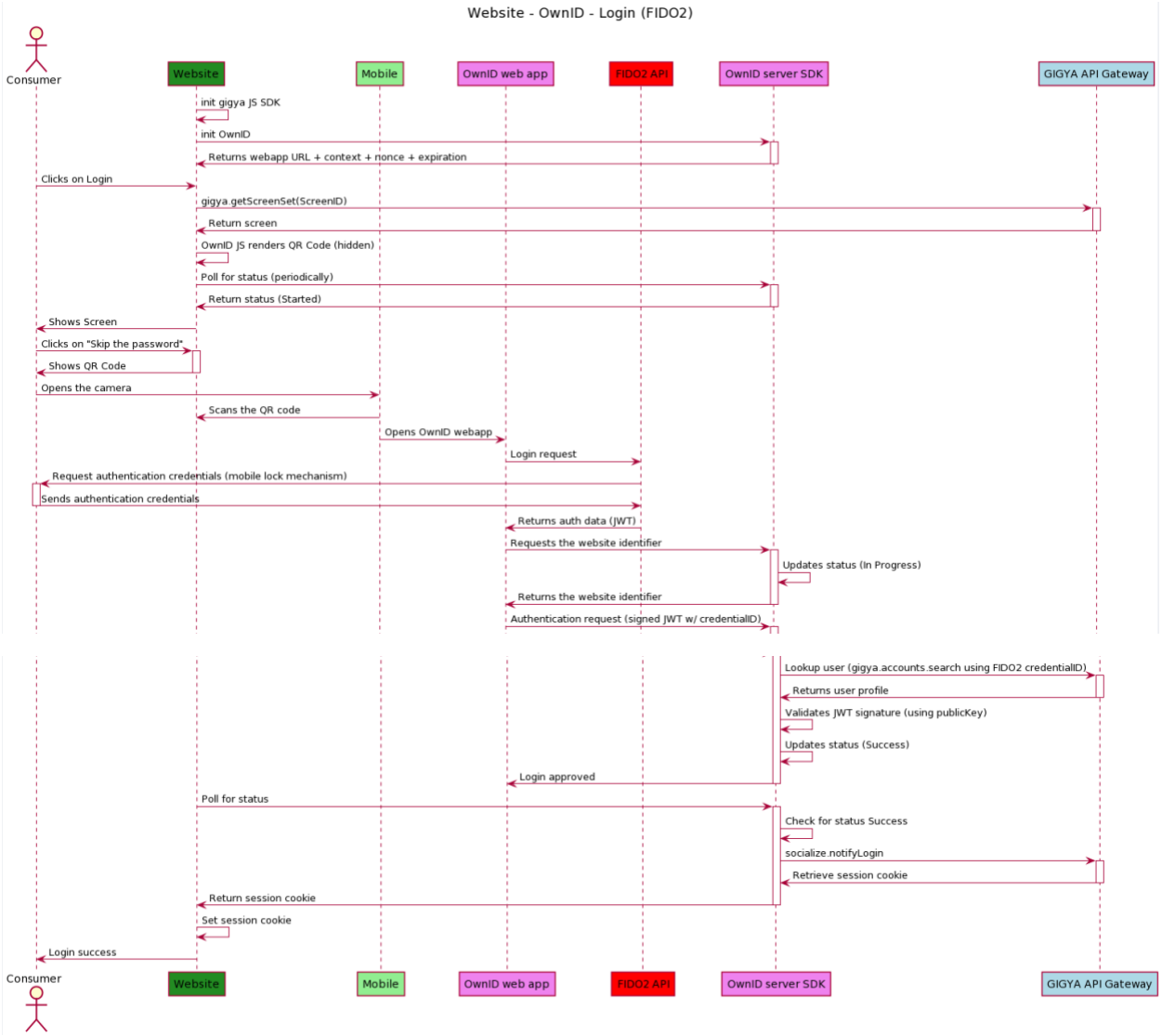

Website - OwnID - Registration (FIDO2)
Consumer
Website
Mobile
OwnID webapp
FIDO2 API
OwnID sever
GIGYA API Gateway
init gigya JS SDK
init OwnID
Returns webapp URL + context + nonce + expiration
Clicks on Register
gigya.getScreenSet(ScreenID)
Return screen
OwnID JS renders QR Code (hidden)
Poll for status (periodically)
Return status (Started) + context
Shows Screen
Fills registration form
Clicks on "Skip the password"
Shows QR Code
Opens the camera
Scans the QR code
Opens OwnID webapp
Authentication request
Request authentication credentials (mobile lock mechanism)
Send authentication credentials
Registration details (public key)
Request the website identifier and website publicKey
Updates status (In Progress)
Returns website identifier and website publicKey (JWT)
Validates JWT (using website publicKey)
Sends public key (JWT)
Updates status (Success)
Returns OK
Device ready
Poll for status (periodically)
Return status (Success) + context + payload (publicKey)
Completes registration form
Submit registration form
gigya.accounts.register()
including data.ownidConnections fields
Registration completed successfully
Consumer
Website
Mobile
OwnID webapp
FIDO2 API
OwnID sever
GIGYA API Gateway

Login Flow

1. An end-user navigates to the customer's website and goes to the login page. He is presented with an option to use a mobile phone to login without a password by clicking on a "skip password" link on top of the password field in the login form
2. A click on the link initiates a call to OwnID API that returns a response with the Webapp URL, context and nonce:
    a. Webapp URL – A URL pointing to the location where the OwnID web app is hosted (app.ownid.com)
    b. Context – A UUID that should be set in every request to the server so cached parameters that are part of the register/login can be looked-up
    c. Nonce – A parameter for the server to identify the client
3. On desktop, the data received is used to generate a QR Code that the user is instructed to scan with a phone in order to execute the web app.
   On mobile, the data received is used to execute the web app in a new tab.
4. Once the web app is executed, the OwnID web SDK starts a polling process to the OwnID server to get the status of the execution in order to know when the user has finished the login process.
5. The web app asks the user to identify with biometrics by using the browser's native WebAuthN API (FIDO2). The WebAuthN structure is sent to the OwnID server and the OwnID server finds the user in the identity management system by using the signed WebAuthN structure.
6. OwnID server gets a session key for the user from the identity management system and stores it under the context of the original request.
7. The web SDK's polling process gets the session key and uses it to log the user in through the identity management system.

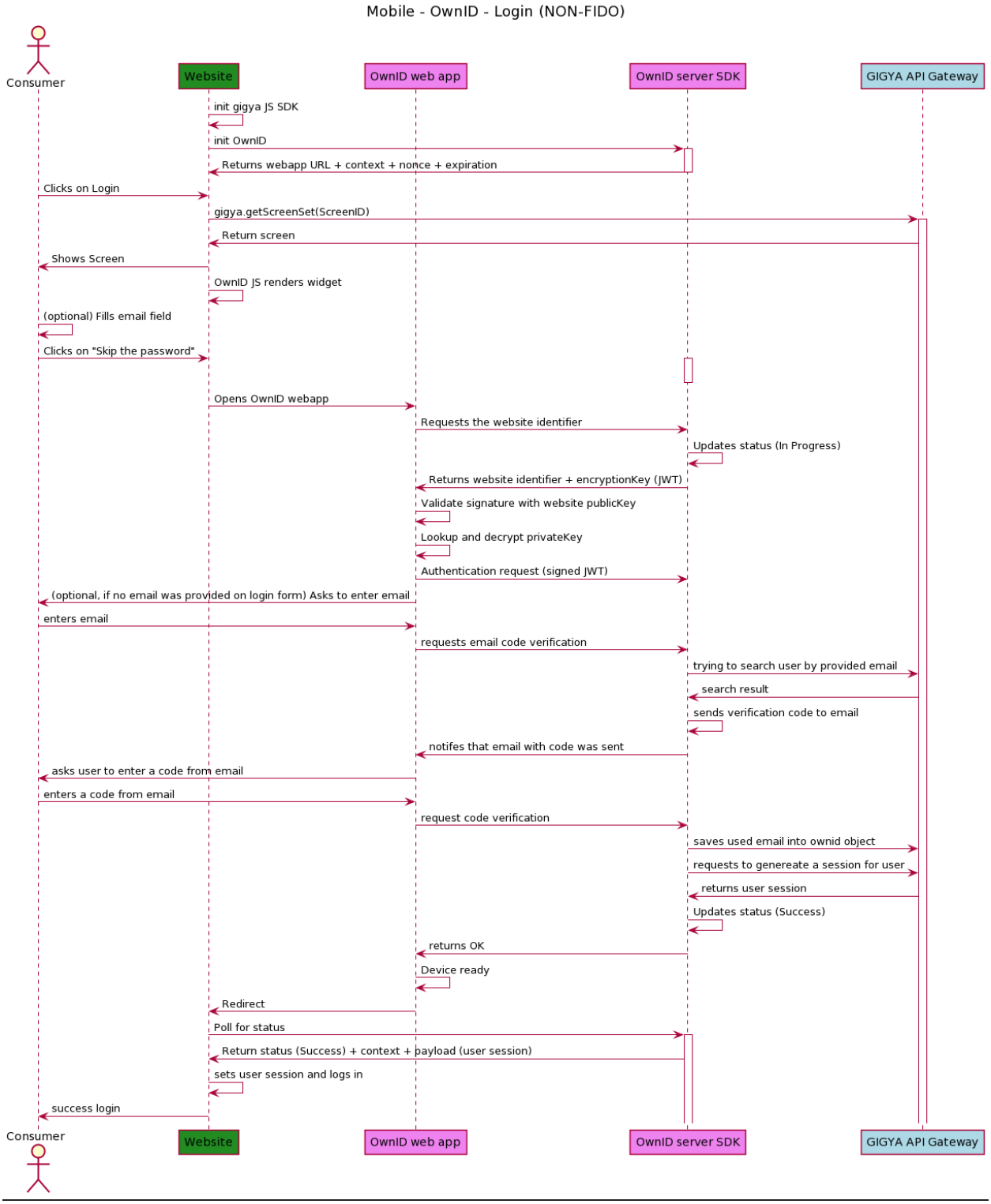


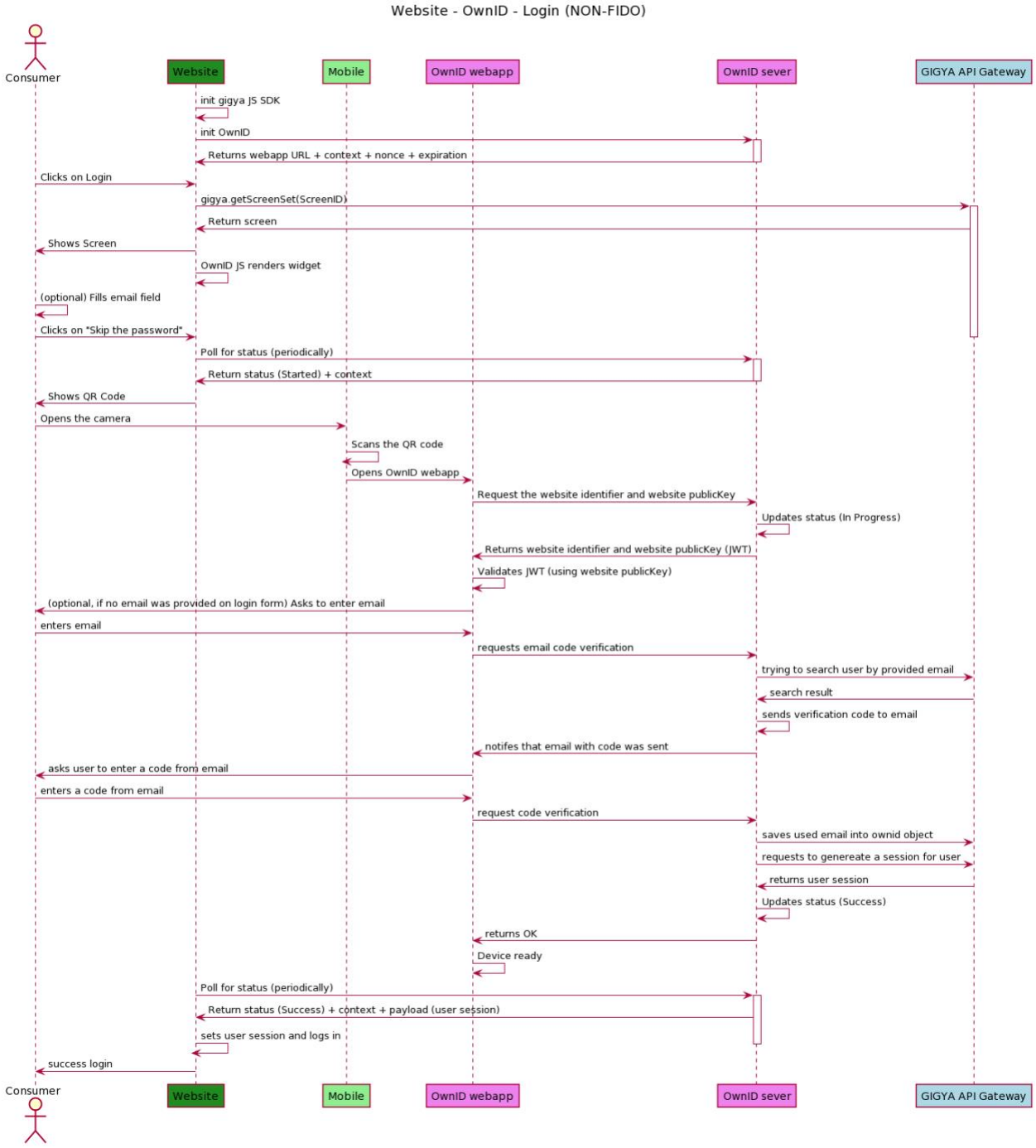
The web SDK has identical SDKs for android & iOS in order to integrate with native apps in the same way.



Non Fido Login Flow

1. An end-user navigates to the customer's website and goes to the login page. He is presented with an option to use a mobile phone to login without a password by clicking on a "skip the password" link on top of the password field in the login form
2. A click on the link initiates a call to OwnID API that returns a response with the Webapp URL, context and nonce:
    a. Webapp URL – A URL pointing to the location where the OwnID web app is hosted (app.ownid.com)
    b. Context – A UUID that should be set in every request to the server so cached parameters that are part of the register/login can be looked-up
    c. Nonce – A parameter for the server to identify the client
3. On desktop, the data received is used to generate a QR Code that the user is instructed to scan with a phone in order to execute the web app.
   On mobile, the data received is used to execute the web app in a new tab.
4. Once the web app is executed, OwnID web SDK starts a polling process to the OwnID server to get the status of the execution in order to know when the user has finished the login process.
5. The web app asks the user to identify with biometrics by using the browser's native WebAuthN API (FIDO2). In case the device isn't supported by WebAuthN, then we use the fallback mechanism and send the user one time code to the user's email.
6. The user enters the one time code from his email, and then the OwnID server verifies the code and enables the session.
7. OwnID server gets a session key for the user from the identity management system and stores it under the context of the original request.
8. The web SDK's polling process gets the session key and uses it to log the user in through the identity management system.


Mobile - OwnID - Login (NON-FIDO)
Consumer
Website
OwnID web app
OwnID server SDK
GIGYA API Gateway
init gigya JS SDK
init OwnID
Returns webapp URL + context + nonce + expiration
Clicks on Login
gigya.getScreenSet(ScreenID)
Return screen
Shows Screen
OwnID JS renders widget
(optional) Fills email field
Clicks on "Skip the password"
Opens OwnID webapp
Requests the website identifier
Updates status (In Progress)
Returns website identifier + encryptionKey (JWT)
Validate signature with website publicKey
Lookup and decrypt privateKey
Authentication request (signed JWT)
(optional, if no email was provided on login form) Asks to enter email
enters email
requests email code verification
trying to search user by provided email
search result
sends verification code to email
notifes that email with code was sent
asks user to enter a code from email
enters a code from email
request code verification
saves used email into ownid object
requests to genereate a session for user
returns user session
Updates status (Success)
returns OK
Device ready
Redirect
Poll for status
Return status (Success) + context + payload (user session)
sets user session and logs in
success login
Consumer
Website
OwnID web app
OwnID server SDK
GIGYA API Gateway


Website - OwnID - Login (NON-FIDO)
Consumer
Website
Mobile
OwnID webapp
OwnID sever
GIGYA API Gateway
init gigya JS SDK
init OwnID
Returns webapp URL + context + nonce + expiration
Clicks on Login
gigya.getScreenSet(ScreenID)
Return screen
Shows Screen
OwnID JS renders widget
(optional) Fills email field
Clicks on "Skip the password"
Poll for status (periodically)
Return status (Started) + context
Shows QR Code
Opens the camera
Scans the QR code
Opens OwnID webapp
Request the website identifier and website publicKey
Updates status (In Progress)
Returns website identifier and website publicKey (JWT)
Validates JWT (using website publicKey)
(optional, if no email was provided on login form) Asks to enter email
enters email
requests email code verification
trying to search user by provided email
search result
sends verification code to email
notifes that email with code was sent
asks user to enter a code from email
enters a code from email
request code verification
saves used email into ownid object
requests to genereate a session for user
returns user session
Updates status (Success)
returns OK
Device ready
Poll for status (periodically)
Return status (Success) + context + payload (user session)
sets user session and logs in
success login
Consumer
Website
Mobile
OwnID webapp
OwnID sever
GIGYA API Gateway