Web SDK - Quick Start Guide

The following guide helps you integrate the HyperKYC Web SDK and launch your first workflow in minutes.

> **ADDITIONAL RESOURCE**
>
> - Complete Sample Project: A ready to use example project with all the code you need to get started with the Web SDK quickly.

Step 1: Add SDK to Your Project

Add the following script to your HTML page:

HTML

```
<script src="https://hv-web-sdk-cdn.hyperverge.co/hyperverge-web-sdk@<actual_SDK_VERSION>/src/sdk.min.js"></script>
```

> **PRO-TIP**
>
> Replace <SDK_VERSION> with the latest version listed in the Web SDK Changelog .

Step 2: Initialize and Launch the SDK

Initialize the SDK and launch a verification workflow in your JavaScript code:

JavaScript

```
const hyperKycConfig = new HyperKycConfig(
  "<ACCESS_TOKEN>",        // Short-lived token from your backend
  "<WORKFLOW_ID>",         // Workflow ID from HyperVerge Dashboard
  "<TRANSACTION_ID>",      // Unique identifier for this application
  true                     // Optional: true to show landing page, default is false
);
// Register callback to handle SDK result when verification completes
const handler = (HyperKycResult) => {
  // Log result for debugging
  console.log(
    `status=${HyperKycResult.status} code=${HyperKycResult.code} message=${HyperKycResult.message}`
  );
  // Handle verification outcome
  switch (HyperKycResult.status) {
    case "auto_approved":
      // All checks passed - update UI, proceed
      break;
    case "auto_declined":
      // Verification failed - show rejection UI
      break;
    case "needs_review":
      // Ambiguous result - show pending review UI
      break;
    case "user_cancelled":
      // User exited flow - handle gracefully
      break;
    case "error":
      // Technical failure - show retry option
      break;
  }
};
// Launch the HyperVerge SDK with configuration
await HyperKYCModule.launch(hyperKycConfig, handler);
```

The following table describes each parameter in the configuration:

Parameter	Description	Source
accessToken	Short-lived token from your backend	Generate Access Tokens
workflowId	Workflow identifier	HyperVerge Dashboard
transactionId	Unique session identifier	Generated by your backend

That's it! You've launched your first HyperKYC workflow.

Step 3: Handle Results & Test the Flow

The callback in Step 2 returns one of these statuses:

Status	Description
auto_approved	User verified successfully
auto_declined	Application rejected automatically
needs_review	Flagged for manual review
user_cancelled	User exited before completion
error	SDK or network issue

For detailed response formats, error codes, and field descriptions, see the SDK Response documentation .

> Test: Build & run your app, trigger the launcher, complete a sample journey, and check the log output to confirm integration.

Next Steps

Explore advanced capabilities:

- Additional Configurations: See the Integration Guide for detailed configuration options including optional configurations, UI customization, language settings, and dark mode support.

- Error Codes & Troubleshooting: Refer to the Error Response Details for detailed error codes and descriptions.
- Integrate Results Webhook: Receive backend updates when journeys complete using the Results Webhook API.
- Real-time Event Notifications: Track user progress with Real-time Event Notifications
- Sample Project: Check out the Complete Sample Project for a ready-to-use example with all the code you need to get started quickly.

Recommendations

Follow these best practices to ensure a secure and smooth integration:

> **IMPORTANT**
>
> - Validate camera & microphone permissions before SDK launch (if your workflow requires these)
> - Do not send SDK results directly to your backend for decisioning. To avoid potential man-in-the-middle (MITM) attacks, integrate the Results Webhook instead to securely receive verified outcomes from HyperVerge servers.
> - Ensure your domain and IP addresses are whitelisted for security. You can manage IP whitelisting through CORS Whitelisting on the HyperVerge One dashboard.