INNOVATRICS

IDENTITY VERIFICATION SERVICE RIVaaS

Integration Manual

This document describes how to integrate with Innovatrics RIVaaS service.

Table of Contents

1 What is RIVaaS? 2
2 Prerequisites 2
3 How to integrate with RIVaaS 3
3.1 Concept 3
3.1.1 Redirect flow 4
3.1.2 Iframe flow 5
3.2 Integration steps 5
3.2.1 Obtain JWT from Auth0 5
3.2.2 Obtain session token from RIVaaS API 7
3.2.3 Initialization of RIVaaS App 8
Redirect flow 8
Iframe flow 8
3.2.4 User identity verification 9
3.2.5 Obtaining verification result and data 9
Verification result 9
Redirect flow 9
Iframe flow 10
Webhook event API 10
3.3 Error Handling 12

INNOVATRICS

1 What is RIVaaS?

Remote Identity Verification as a Service (RIVaaS) is a service allowing web platforms to verify the identity of their users remotely without the need of complicated integration of Innovatrics' identity verification toolkit - DOT. It is a web application that can be accessed either as a standalone web application (via redirect) or as an iframe.

2 Prerequisites

Before integrating with RIVaaS, integrator needs to contact Innovatrics representative and provide:

- verifiedUrl (`VERIFIED_URL`) - URL of the web platform to which RIVaaS will redirect the user after successful identity verification.
- rejectedUrl (`REJECTED_URL`) - URL to which RIVaaS will redirect the user when verification result is rejected or failed.
- unverifiedUrl (`UNVERIFIED_URL`) - URL to which RIVaaS will redirect the user when the verification was cancelled by user.
- callbackUrl (`CALLBACK_URL`) - URL to which RIVaaS API will send sensitive information about customer in case of successful verification.
- logoUrl (`LOGO_URL`) - URL to image of company logo of the integrator that will be displayed in RIVaaS App. (SVG for light background recommended, PNG supported too)

Also, it is good practice to provide `CUSTOMER_NAME` for customer identification.

After providing these, the integrator will obtain:

- Auth0 client id (`AUTH0_CLIENT_ID`)
- Auth0 client secret (`AUTH0_CLIENT_SECRET`)
- Auth0 issuer base URL (`AUTH0_ISSUER_BASE_URL`)
- RIVaaS service URL (`RIVAAS_SERVICE_URL`)
- RIVaaS app URL (`RIVAAS_APP_URL`)

It is necessary for the integrator to ensure the following parts of the solution:

- backend application*
- frontend application (website).

*The requirement for the backend application is there because every customer has to authenticate against Auth0. Since frontend applications are public, it should be avoided to store Auth0 client secret in them. As a result, integrator's account at RIVaaS could be exploited. Therefore, it is the backend application that is required to authenticate against Auth0 and get JWT token. This token is then used to authenticate all calls against RIVaaS API.

3 How to integrate with RIVaaS

Before integrating with RIVaaS, please have a look at the overall concept of RIVaaS integration in the Concept section.

If you are ready to integrate with RIVaaS, please proceed to the Integration steps section.

3.1 Concept

Before integrator proceeds to integrate with RIVaaS, it's essential to understand how data flows between RIVaaS and integrator's applications. From higher-level perspective, the RIVaaS ID verification process can be split into the following steps:

01 User starts remote identity verification on integration website.

02 Website requests session token from integration backend.

03 Integration backend requests JWT from Auth0 using provided client id and secret.

04 Auth0 returns JWT to integration backend.

05 Integration backend requests session token from RIVaaS API using JWT.

06 Integration backend returns session token to integration website.

07 Integration website redirects user to RIVaaS App with session token or loads RIVaaS iframe with session token.

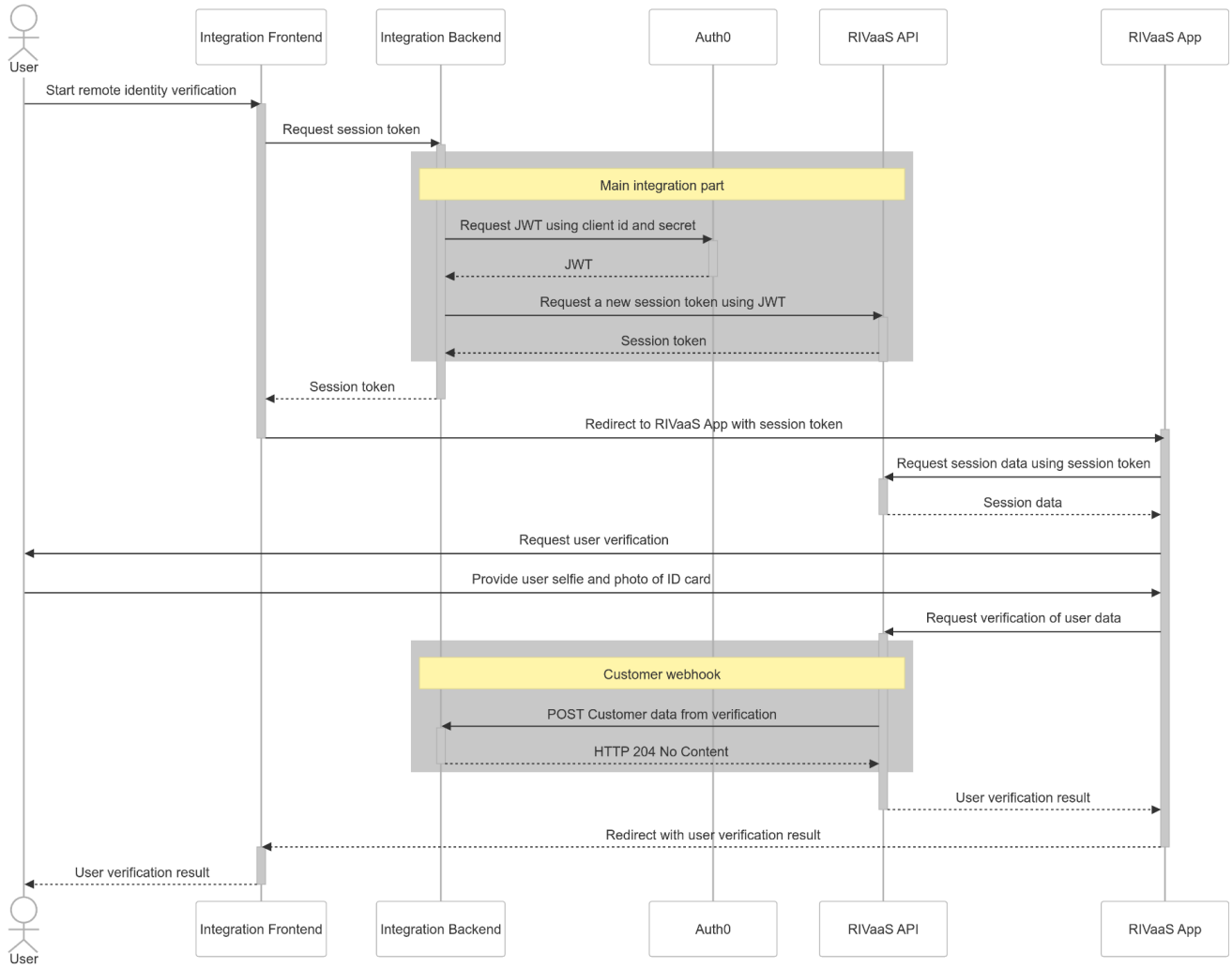
08 User verifies himself in RIVaaS App.

09 RIVaaS API provides customer sensitive data to integration backend via webhook HTTP POST request.

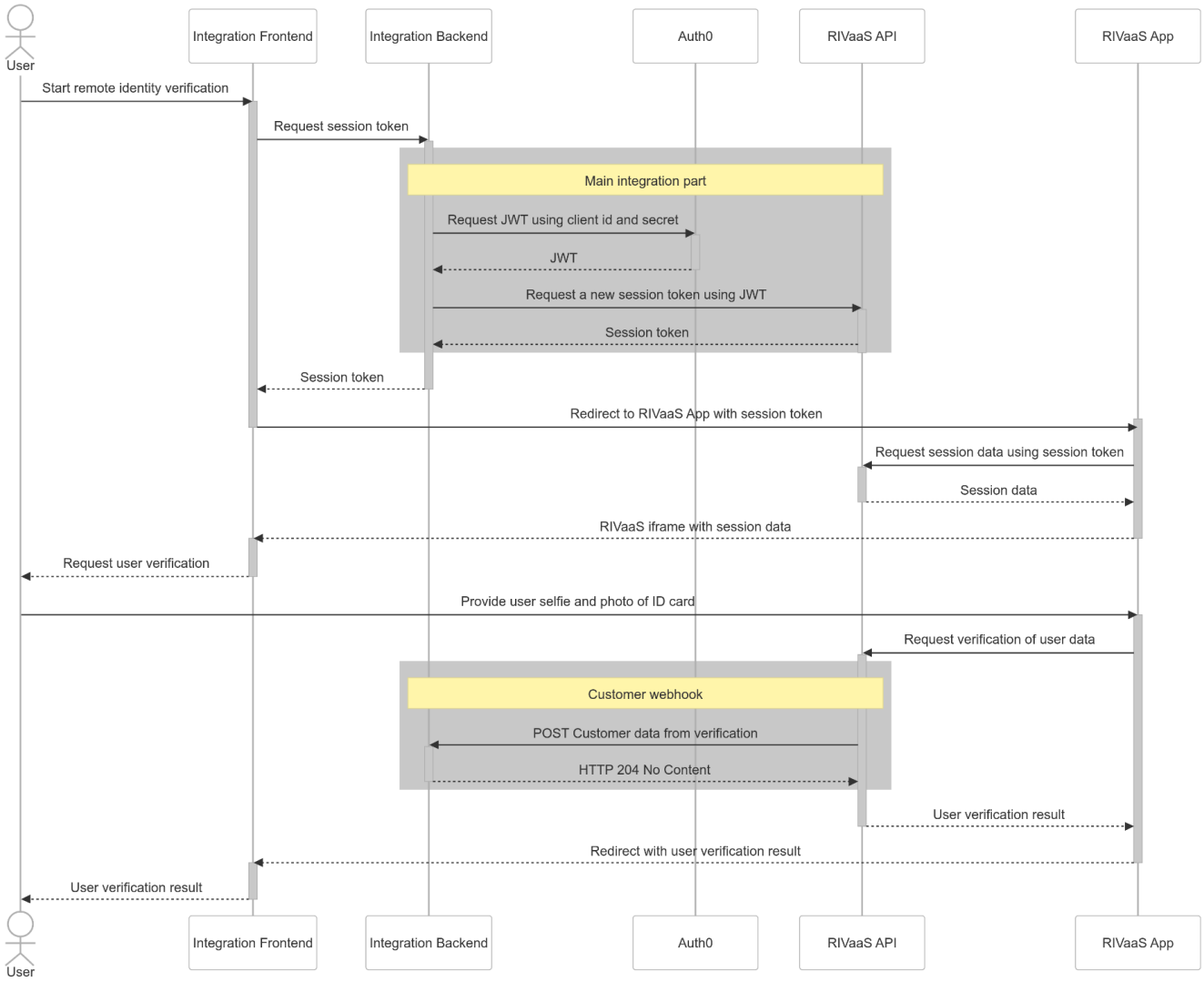
10 RIVaaS App returns user's verification result either as a redirect to specified integration website or as a message in RIVaaS iframe.

To better understand the flow, please refer to the sequence diagrams below.

3.1.1 Redirect flow



3.1.2 Iframe flow



3.2 Integration steps

3.2.1 Obtain JWT from Auth0

First step is to obtain a JWT from Auth0. This is done by calling the Auth0 API with provided client id and secret. This step is done on the integration backend.

NOTE

JWT token is not one time token. It is valid for one week so we strongly recommend to store it in a secure way and refresh it when needed.

In the example below, we use fetch function to call Auth0 API in JavaScript. Please note that this is just an example and you can use any other library to call Auth0 API.

```
const AUTH0_ISSUER_BASE_URL = '<AUTH0_ISSUER_BASE_URL>';
const AUTH0_CLIENT_ID = '<AUTH0_CLIENT_ID>';
const AUTH0_CLIENT_SECRET = '<AUTH0_CLIENT_SECRET>';

try {
  const response = await fetch(`${AUTH0_ISSUER_BASE_URL}/oauth/token`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      audience: 'https://verify-identity.innovatrics.com/service',
      grant_type: 'client_credentials',
      client_id: AUTH0_CLIENT_ID,
      client_secret: AUTH0_CLIENT_SECRET,
    }),
  });

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  const data = await response.json();
  // use data
} catch (error) {
  // Refresh token if needed
  // handle error
}
```

It is a good practice to implement refresh token mechanism in case the JWT expires.

3.2.2 Obtain session token from RIVaaS API

Second step is to obtain a session token from the RIVaaS API. This is done by calling the RIVaaS API with the JWT obtained in previous step. This step is also done on the integration backend.

In the example below, is the fetch function used to call the RIVaaS API in JavaScript. Please note that this is just an example and you can use any other library to call the RIVaaS API.

```
const RIVAAS_SERVICE_URL = 'https://<RIVAAS_SERVICE_URL>';
const ACCESS_TOKEN = '<JWT_FROM_AUTH0>';
const LOCALE = 'en'; //desired localization for RIVaaS App
try {
  const response = await fetch(`${RIVAAS_SERVICE_URL}/api/v1/session`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${ACCESS_TOKEN}`,
    },
    body: JSON.stringify({
      configuration: {
        locale: LOCALE,
      },
    }),
  });

  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }

  const data = await response.json();
  // use data
} catch (error) {
  // handle error
}
```

3.2.3 Initialization of RIVaaS App

RIVaaS App is a web application that can be accessed either as a standalone web application (redirect) or as an iframe. In both cases, RIVaaS App needs to be initialized with the session token obtained in the previous step.

Redirect flow

In case of the redirect flow, RIVaaS App is initialized by redirecting the user to RIVaaS App with the session token as a query parameter.
Example below is written in vanilla JavaScript, but you can use any other library to redirect.

```
const RIVAAS_APP_URL = 'https://<RIVAAS_APP_URL>';
const SESSION_TOKEN = '<SESSION_TOKEN>';

return (window.location.href =
`${RIVAAS_APP_URL}/?sessionToken=${SESSION_TOKEN}`);
```

Iframe flow

In case of the iframe flow, RIVaaS App is initialized by loading RIVaaS iframe with session token and `viewType` as a query parameter.
Example below is written in vanilla JavaScript, but you can use any other library to create the iframe.

```
const RIVAAS_APP_URL = 'https://<RIVAAS_APP_URL>';
const SESSION_TOKEN = '<SESSION_TOKEN>';
const VIEW_TYPE = 'iframe';

const iframeUrl =
`${RIVAAS_APP_URL}?sessionToken=${SESSION_TOKEN}&viewType=${VIEW_TYPE}`;

const iframeElement = document.createElement('iframe');
iframeElement.src = iframeUrl;
iframeElement.width = '100%';
iframeElement.height = '100%';
iframeElement.allow = 'camera';
iframeElement.style.border = 'none'; // add this line if you want to remove border
around iframe
```

```
document.body.appendChild(iframeElement);
```

3.2.4 User identity verification

This is the part where the main functionality happens. User is now in RIVaaS App and he needs to verify his identity. This is done by providing selfie and a photo of his ID card. After user provides these, RIVaaS App will upload them to the RIVaaS API for verification. After verification is done, RIVaaS App will either redirect the user back to the integration website with the verification result or post the result through postMessage in case of the iframe.

3.2.5 Obtaining verification result and data

In this part, user provided necessary data for verification. Now, RIVaaS will provide data from verification to integrator. Overall status of verification will be sent to integrator via redirect (Redirect flow) or postMessage (iframe flow). Sensitive data of verified user will be sent to integrator via webhook event API.

Verification result

Both flows provide the same verification result. The verification result consists of the following parameters:

- `timestamp` - Timestamp of verification result used for verification of result integrity
- `status` - Overall status of verification (`success` | `failed`)

Redirect flow

In case of the redirect flow, RIVaaS App will redirect user back to the integration website with the verification result as a query parameter. Please note that the user might not be redirected back to the same page where he started the remote identity verification. He will be redirected to the page specified in `VERIFIED_URL`, `REJECTED_URL` or `UNVERIFIED_URL`.
Result of user verification is in query parameters.

Successful verification result example:

```
https://www.rivaas-customer.com/happy-path
  ?timestamp=1702039347154
```

```
  &status=success
```

Iframe flow

In case of iframe flow, RIVaaS App will post verification result to the integration website using `postMessage` API. See more about `postMessage` API [mdn documentation](#).
Please note that the verification result is posted to the same page where the user started remote identity verification.
Result of user verification is posted as a message containing the verification result.

Example of `postMessage` listener:

```
window.addEventListener('message', (event) => {
  if (event.origin !== 'https://www.rivaas-customer.com') {
    return;
  }

  const { data } = event;

  // process data
});
```

Please note that `event.origin` will be parsed from `VERIFIED_URL` provided to Innovatrics initially.

Webhook event API

Before user will be redirected back to the integration website or the verification result will be posted through `postMessage`, the integration backend (API) will receive the customer sensitive data from RIVaaS API via webhook event.
WebHook event is represented as an (response) object with the type of event and data. With webhook event API, we can send various data. Right now, we have only one type of webhook event `VERIFICATION_SUCCEEDED`

WebHookEvent type

```
enum WebHookEventType {
  VERIFICATION_SUCCEEDED = 'VERIFICATION_SUCCEEDED',
}
```

WebHookEvent object type

```
type WebHookEvent<T> = {
  event: WebHookEventType;
  data: T;
};
```

Integration backend has to handle this request and store the data in a secure way. The integration backend should also return HTTP 204 No Content to RIVaaS API to confirm that the data was received and stored.

`VERIFICATION_SUCCEEDED` webhook event data will contain the following parameters:

- `sessionToken` - Identifier of the session
- `result` - Overall status of verification (will be always `VERIFIED`)
- `timestamp` - Timestamp of verification result used for verification of result integrity
- `customer` - Object containing sensitive data of verified
  - `address` - Object containing address of verified user
    - `fullAddress` - Full address (*e.g. 5 4th Street New York*) of verified user
  - `documentNumber` - Document number of verified ID card
  - `dateOfBirth` - Date of birth of verified user
  - `dateOfExpiry` - Date of expiry of verified ID card
  - `fullName` - Full name of verified user
  - `placeOfBirth` - Place of birth of verified user
  - `selfie` - Base64 encoded selfie of verified user

`VERIFICATION_SUCCEEDED` webhook event data type

```
export type VerificationSucceededResult = {
  customer: {
    address: {
      fullAddress?: string;
    };
    documentNumber?: string;
    dateOfBirth?: string;
    dateOfExpiry?: string;
    fullName?: string;
```

```
    placeOfBirth?: string;
    selfie: string;
  };
  timestamp: string;
  result: SESSION_STATE;
  sessionToken?: string;
};
```

3.3 Error Handling

All the errors that could occur in integration are returned as HTTP status codes and will be handled by RIVaaS App. Integrator has to handle only the errors that might occur during initialization of the RIVaaS App or when retrieving the verification result.

For security reasons, RIVaaS App does not return any error details to integration website. Every error has a "tracing ID". This ID is unique for every error and it is used for debugging purposes. If any error is encountered, please contact Innovatrics providing the tracing ID in the error message to analyze the issue.