Videoflo API

A platform to facilitate workflow-based applications over video calls within your own web or mobile application.

BOT AI ML Private Limited

(Subsidiary of Integra Micro Systems Pvt. Ltd.)

Bangalore

August 2021

RESTRICTIONS ON DISCLOSURE

BOT AI ML (P) Ltd.

# 4, Bellary Road, 12th KM, Jakkur,

Bangalore 560 064, Karnataka, India.

Phone: +91-80-28565801/26585802
Fax: +91-80-28565800

Website: www.botaiml.com

TABLE OF CONTENTS

1. INTRODUCTION

Videoflo is a platform to facilitate workflow-based applications over video calls within your own web or mobile application. It provides a complete stack of technologies very easy to integrate in your application. Our main goal is to allow developers to re-imagine all scenarios where in-person interaction is required and allow them to build these as workflows with real-time communications to their apps very fast and with low impact in their code.

Videoflo is fully on-premise and most of the functionality is made available via simple ReST APIs and frontend SDKs (Angular & Web Component only as of now).

2. KEY CONCEPTS

To fully understand how to create and manage sessions with participants, you need to understand these concepts:

- **App ID**: When making ReST API calls to Videoflo, your application needs to authenticate itself. Moreover, your account with Videoflo can contain multiple projects. Each project will be assigned a unique appId; this is the equivalent of username for that project.
- **Secret Key**: Each one of your projects will also be assigned an auto-generated secretKey; this is the equivalent of password for that project. The appId and secretKey should be stored securely on your server and never be shared or served with the clients or any other process.
- **Token**: This is an authentication token generated using the appId and secretKey. Your application server should generate the tokens and pass them on to frontend applications, so that the frontend can authenticate with Videoflo servers without needing access to the appId and secretKey.
- **Session**: A multi-party video conference session. Every meeting has a unique sessionId, and a set of activities to be performed during the session.
- **Participant**: A user trying to join a multi-party media session. Every participant has a participantId, an externalParticipantId that you can pass in to map the attendee to a user in your system. The sessionId, participantId and authToken are key part of the data needed to join a session, and you need to disseminate that data to all users trying to join the session.
- **Activities**: A workflow consists of multiple steps to be executed. In Videoflo such steps are called Activities. Videoflo comes with a built-in set of activities. In future, APIs will be provided for integration partners to build their own activities as well. However, current support is limited to built-in activities only.
- **Webhooks**: Videoflo needs a way to send events and data back to your application in near realtime during the session. You are expected to write HTTP endpoints that can accept POST requests and configure the URL of these endpoints against the webhook events when creating a session. Videoflo will then POST data to the appropriate endpoint as configured while creating the session.

Now that you understand these key concepts, the next important aspect to be understood is that you will be writing code and integrating at two levels:

1. Backend: This is where you would write all the code to:
    a. Generate auth tokens on behalf of your apps and participants
    b. Create and configure sessions

c. Respond to data gathered events (via webhooks) to validate data as and when it is gathered
d. Receive the final data gathered during the session
e. Retrieve call recording from Videoflo
f. Retrieve data from Videoflo (in case there was an error while processing webhook)

2. Frontend: As of now only Angular and Web Component (plain HTML5 with JS & CSS) are supported. You need to include the <videoflo-component> in your angular component / web page and pass in the authToken, sessionId & participantId parameters. The component will connect to Videoflo server and work pretty much autonomously until the workflow finishes. The frontend components are stateful and will recover gracefully from any network disconnects or page refreshes. As long as you provide the exact same sessionId & participantId, the session will resume from where it got disconnected.

3 API REFERENCE

The API definition is also available as OpenAPI document at https://demo-api.videoflo.net/api/swagger/. You can easily generate HTTP client wrappers to call our APIs in the programming language of your choice using the openapi-generator-cli tool.

3.1 GET TOKEN

Before you can start making ReST API calls to create and manage sessions, you will need to authenticate your application. For this, you will need to generate an authentication token by posting your appId and secretKey to the token endpoint. Each token is valid for xx hours; upon expiry you can generate a new token and use it to continue to make API calls. Note that we recommend generating separate token per participant, when connecting them to the call.

REQUEST URL:

```
https://demo-api.videoflo.net/token/getToken
```

METHOD:

```
POST
```

REQUEST HEADERS:

```
Content-Type: application/json
```

REQUEST BODY:

```
{
      "appId": "<your app id>",
      "secretKey": "<your secret key>"
}
```

RESPONSE BODY:

The response body will be in the following format:

```
{
  "accessToken": "<your access token>"
}
```

3.2 CREATE SESSION

Create a new Videoflo session to execute a workflow over a video conference call.

REQUEST URL:

```
https://demo-api.videoflo.net/videoSessions/createSession
```

METHOD:

```
POST
```

REQUEST HEADERS:

```
Content-Type: application/json
Authorization: Bearer <your auth token>
```

REQUEST BODY:

```
{
  "name": "string",
  "participants": [
    {
      "externalParticipantId": "string",
      "name": "string",
      "role": "string",
      "videoLayoutSettings": {
        "<externalParticipantId-1>": 'small',
        "<externalParticipantId-2>": 'big'
      },
    }
  ],
  "activities": [
    {
      "id": "string",
      "activityType": "string",
      "gatherFrom": [
        "string"
      ],
      "displayTo": [
        "string"
      ],
      "configuration": {},
      "onActivityDataGathered": "string",
      "onActivityAction": "string"
    }
  ],
  "webhooks": {
    "onParticipantConnected": "<your webhook endpoint url>",
    "onParticipantDisconnected": "<your webhook endpoint url>",
    "onWorkflowFinished": "<your webhook endpoint url>",
    "onRecordingAvailable": "<your webhook endpoint url>",
    "onRecordingError": "<your webhook endpoint url>"
  }
}
```

JSON fields in request body:

Field	Type	Description
name	String	Name for this Videoflo session. This string is displayed to the participants as the subject of the meeting.

participants	Participant[]	An array of participant objects who will be participating in this meeting.
externalParticipantId	String	A unique identifier that you can pass in to map the participant to a user in your system. Note that the externalParticipantId must be unique within the scope of the session. For example, if you have a customer table and an agent table, with both using autogenerated ID columns, then you need to prefix the externalParticipantId so that ID collisions do not occur.
name		Full Name of the participant. This will be displayed in the user interface and used in the email/sms notification templates.
role		Role of the participant in this call (only "customer" and "agent" are supported as of now)
videoLayoutSettings	Object (map)	Using this you can control how all participants are shown to this participant, including themselves. Each key in this object should refer to an externalParticipantId.
<externalParticipantId as key>	String	Valid values are Small / Big. This value controls whether the participants video is shown enlarged or as a small thumbnail.
activities	Activity[]	Represents the list of activities to be carried out during the call along with the activity specific configuration.
id	String	A unique identifier (within the scope of the session) for this activity. For example, you could have more than one "capture image" activities in the session. But the id should uniquely identify each activity separately such as "capturePhoto", "captureSignature" etc.
activityType	String	The Activity Type. Must be one of the valid activities as listed in 5. Activities section.
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data. For example, if you want to capture a photo of the signature card from the customer, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who will gather and review the data from other participants. For example, if you want the agent to capture a photo of the signature card, you need to specify "displayTo": ["agent"]
configuration	Object	Activity specific configuration object, if applicable, as listed in 5.1 Configuration section.
onActivityDataGathered	String	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations

		and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
webhooks	String[]	An array of URLs where Videoflo needs to post data for the events below.
onParticipantConnected	String	A webhook URL to POST the data to, when a participant gets connected to the call. Note that this connection is to the "workflow engine" and does not necessarily indicate that the participant is connected to the video/audio conference. You will receive this data in real-time and could be useful to update the "connected / disconnected" status on a per participant basis in your system.
onParticipantDisconnected	String	A webhook URL to POST the data to, when a participant gets disconnected from the call. Note that this disconnection is from the "workflow engine" and does not necessarily indicate that the participant is disconnected from the video/audio conference. You will receive this data in real-time and could be useful to update the "connected / disconnected" status on a per participant basis in your system.
onWorkflowFinished	String	A webhook URL to POST the data to, when all activities have been completed and the agent has reviewed the summary and clicked on "Finish" button. You will receive a large payload with cumulative data from all the activities during the session including images, location, ip address, OCR results, face recognition results etc. Payloads for individual activity types are listed in 5.2 Data section. Note that the call recording module might not have completed encoding the video by this time. Therefore, you will receive a separate webhook notification for this.
onRecordingAvailable	String	A webhook URL to POST the data to, when the call recording has been encoded, stored to disk and is now available for downloading. Note that you will only receive a JSON payload with the sessionId and other metadata. You will need to make a ReST API call yourself to fetch the actual recording file and store it in your system.

onRecordingError	String	A webhook URL to POST the data to, in case there was an error while beginning call recording, encoding the video or storing the video to disk. Note that this error will happen under rare conditions such as: • Necessary dependencies to encode video (ffmpeg etc.) are not installed and configured correctly on Videoflo side. • The system does not have enough memory to complete encoding the video. • Disk is full and no space is available to store the video.

RESPONSE BODY:

The response body will be in the following format:

```
{
  "sessionId": "string",
  "participants": [
    {
      "participantId": "string",
      "externalParticipantId": "string"
    }
  ]
}
```

JSON fields in response body:

Field	Type	Description
sessionId	String	
participants	ParticipantDTO[]	An array of participant objects consisting mapping of externalParticipantId and Videoflo participant ids.
participantId	String	Participant ID generated by Videoflo for this participant.
externalParticipantId	String	externalParticipantId as submitted by you in the request body.

3.3 GET ACTIVITY DATA

This endpoint allows you to download the cumulative data from all the activities during the session including images, location, ip address, OCR results, face recognition results etc. based on the activities configuration specified while creating the session.

REQUEST URL:

```
https://demo-api.videoflo.net/videoSession/getActivityData/{sessionId}
```

METHOD:

```
GET
```

REQUEST HEADERS:

```
Content-Type: application/json
Authorization: Bearer <your auth token>
```

REQUEST BODY:

```
{
  "sessionId": "<Videoflo session id>",
  "data":{},
  "createdOn": "datetime"
}
```

JSON fields in response body:

Field	Type	Description
sessionId	String	Videoflo session id
data	Object	An object with cumulative data from all the activities during the session including images, location, ip address, OCR results, face recognition results etc. See 5.2 Data to know more about the data payloads for different activities.
createdOn	Date	Date when the session was created

3.4 DOWNLOAD CALL RECORDING

Download the call recording in MP4 format.

Note: This recording will only contain the videos as seen during the call. However, additional panels such as the "activity" panel displayed to guide users through the activities is not recorded in the call.

REQUEST URL:

```
https://demo-api.videoflo.net/videoSessions/downloadCallRecording/{sessionId}
```

METHOD:

```
GET
```

REQUEST HEADERS:

```
Content-Type: application/stream
Authorization: Bearer <your auth token>
```

RESPONSE BODY:

```
application/stream
```

The response body will contain the video stream, which you can read and save to appropriate location within your system.

4. BROWSER CLIENT SDKS

We currently provide two SDKs to help build browser-based applications against Videoflo. Both these packages are hosted on GitHub packages at https://github.com/botaiml/videoflo-sdk/packages.

These packages can be installed in your frontend application via npm. Follow instructions at https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-npm-registry#authenticating-to-github-packages to authenticate your NPM cli with GitHub and be able to install @botaiml scoped libraries from NPM.

If you do not want to go through the entire GitHub documentation, execute the following command to authenticate your npm cli with GitHub:

```
npm login --scope=@botaiml --registry=https://npm.pkg.github.com
```

Note1: The above command will further prompt for your GitHub username, password and public email id. Once authentication is successful, it will create a .npmrc file in your home directory with your login token for future use (your password is not stored).

Note2: If you have enabled two-factor authentication for your GitHub account, npm login will keep showing "invalid username / password" error. This is because npm cli is not capable of performing two-factor authentication. To work around this issue, please create a personal access token for your account and use this token as the password for "npm login" command.

Note3: that by specifying `--scope` option, we have restricted npm to use GitHub package registry only for packages with a `"@botaiml/"` prefix. For all other packages, it will continue to use the default npm registry.

4.1 ANGULAR LIBRARY

1. Make sure you installed Angular CLI by executing `npm install -global @angular/cli` command.
2. Ensure that you are not already in an Angular workspace folder.
3. Run the CLI command `ng new` and provide the name `video-kyc`, as shown here:
   `ng new video-kyc`
4. The `ng new` command prompts you for information about features to include in the initial application project. Accept the defaults by pressing the Enter or Return key.
5. Wait for Angular CLI to complete configuration and installation of all packages in your workspace.
6. Go to the workspace directory:
   `cd video-kyc`
7. Videoflo angular library depends on Angular Material library. Therefore add Angular Material to your workspace by following instructions at https://material.angular.io/guide/getting-started.
8. Install videoflo-angular package using the following command:
   `npm install --save @botaiml/videoflo-angular`
9. Open app.module.ts and ensure the following imports are present:

```
import { CommonModule } from '@angular/common';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { VideofloModule } from '@botaiml/videoflo-angular';

imports: [
  CommonModule,
  FormsModule,
  ReactiveFormsModule,
  BrowserModule,
  BrowserAnimationsModule,
  VideofloModule
]
```

10. Now generate a new component for customer to join the call using the following command:
    `ng generate component customer-join`
11. Now open customer-join.component.html file and add the following HTML:

```
<div style="width:100vw; height:100vh;">
```

```
  <videoflo-component
    [apiUrl]="videofloApiUrl"
    [sessionId]="sessionId"
    [participantId]="customerParticipantId"
    [token]="videoFloAccessToken"
    (error)="onVideofloError($event)"
    (leaveSession)="onVideofloLeaveSession($event)">
    </videoflo-component>
</div>
```

You just added the Videoflo component to your newly generated component. Note the parent <div> encapsulating the videoflo-component. This div's style is currently set to occupy 100% of both view width and height. While this works for demo purposes, you will have to make sure that the parent element in your application is appropriately sized and position. Videoflo-component will take up all the width and height of its parent element but will not "force takeover" the entire page. So it is up to you appropriately style and size the parent element.

12. Now open customer-join.component.ts file and add the following code:

```
export class CustomerJoinComponent implements OnInit {
  public videofloApiUrl: string = 'https://demo-api.videoflo.net/api';
  public sessionId: string;
  public customerParticipantId: string;
  public videofloAccessToken: string;

  constructor() {
  }

  ngOnInit(): void {
    //TODO: Add http client call to fetch the sessionId, participantId & token from
the backend and assign to class variables
  }

  onVideofloError(error: any) {
    //TODO: handle the error here. Could be connection errors to workflow engine or
other runtime errors as well.
    console.error(error);
  }

  onVideofloLeaveSession(eventData: any) {
    //TODO: eventData should contain a boolean variable that should tell you whether
the session was ended by completing the Video KYC workflow or by abruptly ending the
call
    console.info(eventData);
  }
}
```

13. Repeat steps 10 to 13 to generate a component for agent to login.

4.2 WEB COMPONENT

TODO

5. ACTIVITIES

The following Activity Types are supported in a Videoflo session:

- Welcome: Allows you to display a welcome message to the customer before beginning the Video KYC process.
- GeolocationVerification: Capture the user's location by invoking browser's geolocation api.
- IpAddressVerification: Lookup the user's IP address to identify the location and identify possible threats such as VPN usage, IP being listed as an abuser etc.

  Note: The above 3 activities will be deprecated in an upcoming release and will be moved to a new feature called "preCallChecks". The pre-call checks will allow you to show a welcome message, obtain user's consent (I Agree checkbox with your custom message), verify and ensure permission to access location, camera & mic are given. You can also add your own custom checklist items such as asking the user if they are in a well-lit and quiet room etc. This will reduce the call time and the burden on the agent to prep the customer for the call.

- QnA: Ask the customer a few random questions to ensure that the agent is speaking to the actual customer.
- MatchHeadPose: Ask the customer to turn their head in a few random directions such as turn-right, turn-left etc., as prompted by the system, to ensure that the person facing the camera is a live person.
- FaceRecognition: Match the face of the customer attending against an image already available within your system. For example, you could match the face of the person in the call against their Aadhar photograph.
- PanRecognition: Ask the customer to display their PAN Card to the camera and perform OCR on it. As of now we only support PAN, however we will add support for additional OVDs such as Voter ID, Aadhar, Driver's License etc. in future releases.
- CaptureImage: Ask the customer to show any document or object of interest and capture an image. Useful to capture specimen signatures etc.
- ManualPrompt: Manually prompt the agent with some instructions during the call.

5.1 CONFIGURATION

5.1.1 WELCOME

The Welcome activity is designed to show a welcome message to the customer before proceeding with the actual KYC process.

Configuration Example:

```
{
  "id": "welcome",
  "activityType": "Welcome",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "configuration": {
    "title": "Welcome to Video KYC",
    "description": "Video KYC is an alternative to in-person KYC using which banking and financial services customers can complete their KYC process remotely through a video call."
  }
}
```

JSON fields:

Field	Type	Description

id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	Welcome
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data. For example, in case of Video KYC, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the data gathered in this activity. Although, in case of Welcome activity, this field is used to specify which role should be signalled once the user accepts the welcome message. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
configuration	Object	Welcome activity configuration object with the Title and Description to be shown to the user.
title	String	Title to be displayed in the Welcome card shown to the customer.
description	String	Detailed message to be displayed in the Welcome card body shown to the customer. Note: You can add HTML markup to this field, and it will be rendered as HTML inside the card body. However, any scripts will be stripped out before rendering.

5.1.2 GEO LOCATION VERIFICATION

GeolocationVerification activity will invoke the Geolocation API in the browser to fetch the latitude/longitude information and then further fetch the full address using Google Reverse Geocoding API.

Configuration Example:

```
{
  "id": "geolocationVerification",
  "activityType": "GeolocationVerification",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>"
}
```

JSON fields:

Field	**Type**	**Description**
id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	GeolocationVerification
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data.

		For example, in case of Video KYC, the customer’s geo-location needs to be captured. Hence specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the data gathered in this activity. For example, in case of Video KYC, the agent needs to view and verify the data. Hence specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.

5.1.3 IP ADDRESS VERIFICATION

IpAddressVerification activity will lookup the address of the participant’s IP and fetch the location and threat analysis for that IP.

Configuration Example:

```
{
  "id": "ipVerification",
  "activityType": "IpAddressVerification",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>"
}
```

JSON fields:

Field	Type	Description
Id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	IpAddressVerification
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data. For example, in case of Video KYC, the customer’s IP Address metadata needs to be captured. Hence specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the data gathered in this activity.

		For example, in case of Video KYC, the agent needs to view and verify the data. Hence specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.

5.1.4 Q&A

The QnA activity prompts the agent with randomized questions she can ask the customer. The questions and expected answers should be pre-populated with data from your system. This activity can be useful in ensuring that the video of the customer is a livestream of a real person in front of the camera. A pre-recorded video will most certainly fail to match a random question prompt.

Configuration Example:

```
{
  "id": "randomQuestions",
  "activityType": "QnA",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>",
  "configuration": {
    "title": "Question/Answers",
    "description": "Please answer a few questions for us.",
    "noOfQuestionsToAsk": 3,
    "qnaPairs": [
      {
        "question": "What is your Father's name?",
        "expectedAnswer": "Father's Name",
      },
      ... more questions here ...
    ]
  }
}
```

JSON fields:

Field	Type	Description
Id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	QnA
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data.

		For example, in case of Video KYC, agent needs to ask the question and get answer from customer. Hence specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the data gathered in this activity. In case of Q&A activity, this field is used to specify which role can see the question along with expected answer. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
configuration	Object	QnA activity configuration object.
title	String	Title to be displayed in the activity card shown to the customer.
Description	String	Detailed message to be displayed in the activity card body shown to the customer. Note: You can add HTML markup to this field, and it will be rendered as HTML inside the card body. However, any scripts will be stripped out before rendering.
noOfQuestionsToAsk	Int	How many questions to ask the customer. For example, you can provide 5 question/answer pairs but ask only 3 random questions out of these 5.
qnaPairs	QnaPair[]	An array of questions along with expected answers
question	String	The question to ask
expectedAnswer	String	Expected answer for this question

5.1.5 MATCH HEAD POSES

The MatchHeadPoses activity lets you prompt the user to pose their head in specific direction and automatically verify the pose using AI model. This can be useful in ensuring that the video of the customer is a livestream of a real person in front of the camera. A pre-recorded video will most certainly fail to match a random prompt for a specific head-pose.

Configuration Example:

```
{
  "id": "matchHeadPoses",
  "activityType": "MatchHeadPoses",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>"
  "configuration": {
    "title": "",
    "description": "",
    "noOfAttempts": 2,
    "poses": ["faceleft", "faceright", "faceup", "facedown"]
  }
}
```

JSON fields:

Field	Type	Description
id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	MatchHeadPoses
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will gather data. For example, in case of Video KYC, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can begin capture, see and review the data gathered in this activity. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
configuration	Object	MatchHeadPosesConfig object
Title	String	Title to be displayed in the activity card shown to all the participants.
description	String	Detailed message to be displayed in the activity card body shown to all the participants. Note: You can add HTML markup to this field, and it will be rendered as HTML inside the card body. However, any scripts will be stripped out before rendering.
noOfAttempts	Int	Number of images to capture before submitting the images to AI model. The pose is considered to match if 2/3rds of the images result in a successful match with the prompted pose.
poses	String[]	An array of poses to prompt to the customer. Valid values are:

		• faceleft • faceright • faceup • facedown • facestraight • tiltright • tiltleft Note: the array provided will be shuffled and the poses prompted will not be in the same order as given in this parameter.

5.1.6 FACE RECOGNITION

The FaceRecognition activity lets you, capture the photo of the customer from their camera and compare it against a photo provided by your system. For example, if your system already has Aadhar (or any other ID proof) data for the customer, you could provide the this photograph as the source image to compare against the customer's live image. This can be useful in ensuring that the person who has joined the call is the same person as per the ID proof available in your system.

Configuration Example:

```
{
  "id": "faceRecognition",
  "activityType": "FaceRecognition",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>",
  "configuration": {
    "title": "Face Recognition",
    "face1": {
      "sourceType": "Base64",
      "value": "data:image/jpeg;base64,<image data encoded in base64>",
      "caption": "Photo in Aadhar Card",
      "displayTo": ["1"]
    },
    "face2": {
      "sourceType": "Camera",
      "caption": "Photo from Live Stream",
      "capturerExternalId": "1",
      "instructionTitle": "Capturing your photo",
      "instructionDescription": "Please hold your face straight and look towards the
camera",
      "capturerInstructionTitle": "Capture Customer's Photo",
      "capturerInstructionDescription": "Make sure that the customer is faced towards
the camera and the full-frontal face is visible"
    }
  }
}
```

JSON fields:

Field	Type	Description
Id	String	A unique identifier (within the scope of the session) for this activity.

activityType	String	FaceRecognition
gatherFrom	String[]	An array of string representing the roles of the participants from whom this activity will capture the photo. For example, in case of Video KYC, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can capture, compare and review the face recognition results. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
configuration	Object	FaceRecognitionConfig object
title	String	Title to be displayed in the activity card shown to all the participants.
Face recognition requires to face images to work. Therefore, you will have to specify image source configuration for face1 and face2 respectively. The image source configuration is quite flexible and lets you specify image sources such as URL, Upload, Camera & Base64 image.		
face1	Object	ImageSourceConfig object
sourceType	String	Type of image source. Valid values are: • Camera • URL • Base64 (image encoded as base64 and included in the config payload) • Upload (will display a file upload control)
value	String	A valid value based on the image source type specified. • URL: the URL of the image. Note: We do not support any form of authentication. The URL given here must be reachable from the internet and GET verb. • Base64: image encoded as base64. Note: The value should conform to the data: URI format and include header information on image format. For example: `data:image/jpeg;base64`

		• Camera: value is ignored • Upload: value is ignored
caption	String	Caption to be displayed under the image thumbnail
displayTo	String[]	An array of roles to whom the image is visible within the activity card. For example, you can specify only the agent's external id and not include customer's id to hide the Aadhar image from customer but display it to agent.
instructionTitle	String	Title of the instruction card displayed to the participant whose photo is being captured.
instructionDescription	String	Description to be shown in the instruction card displayed to the participant whose photo is being captured.
capturerExternalId	String	External ID of the participant who can click on the capture and compare button. In case of Video KYC, it will be externalParticipantId of the agent.
capturerInstructionTitle	String	Title of the instruction card displayed to the participant whose is capturing the photo.
capturerInstructionDescription	String	Description to be shown in the instruction card displayed to the participant who is capturing the photo.

5.1.7 PAN RECOGNITION

The Welcome activity is designed to show a welcome message to the customer before proceeding with the actual KYC process. This activity can be configured as below.

Example:

```
{
  "id": "panCapture",
  "activityType": "PanRecognition",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>",
  "configuration": {
    "title": "Pan Recognition",
    "responseRequired": true,
    "requiredFields": ["pan_num", "name", "dob", "father_name"],
    "optionalFields": ["face_image", "signature_image", "pan_image"],
    "image": {
      "sourceType": "Camera",
      "caption": "Captured Pan Card Image",
      "capturerExternalId": "1",
      "instructionTitle": "Capturing your pan photo",
      "instructionDescription":  "Please  hold  your  pan  straight  and  show  to  the
camera",
      "capturerInstructionTitle": "Capture Customer's PAN Card Photo",
      "capturerInstructionDescription": "Make sure that the PAN Card is faced towards
the camera, is being held in correct orientation and all the fields are visible and
legible"
    }
  }
}
```

JSON fields:

Field	Type	Description
id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	PanRecognition
gatherFrom	String[]	An array of string representing the roles of the participants from whom PAN image should be captured. For example, in case of Video KYC, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the PAN image and OCR data. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.
onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
configuration	Object	PanRecognitionConfig object
title	String	Title to be displayed in the activity card shown to all the participants.
requiredFields	String[]	An array of mandatory fields to be recognized by OCR. Valid values are: • pan_num • name • dob • father_name • face_image (photograph extracted from PAN in base64) • signature_image (signature image extracted from PAN in base64) • pan_image (image cropped to only include the PAN card area in base64)
optionalFields	String[]	An array of non-mandatory fields to be recognized by OCR.

Note: Any field not specified in both requiredFields and optionalFields will be ignored and not extracted by the OCR engine.		
image	Object	ImageSourceConfig object
sourceType	String	Type of image source. Valid values are: • Camera • URL • Base64 (image encoded as base64 and included in the config payload) Upload (will display a file upload control)
Value	String	A valid value based on the image source type specified. • URL: the URL of the image. Note: We do not support any form of authentication. The URL given here must be reachable from the internet and GET verb. • Base64: image encoded as base64. Note: The value should conform to the data: URI format and include header information on image format. For example: `data:image/jpeg;base64` • Camera: value is ignored Upload: value is ignored
caption	String	Caption to be displayed under the image thumbnail
displayTo	String[]	An array of roles to whom the image is visible within the activity card. For example, you can specify only the agent's external id and not include customer's id to hide the captured image from being visible to customer but display it to agent.
instructionTitle	String	Title of the instruction card displayed to the participant whose photo is being captured.
instructionDescription	String	Description to be shown in the instruction card displayed to the participant whose photo is being captured.
capturerExternalId	String	External ID of the participant who can click on the capture and compare button. In case of Video KYC, it will be externalParticipantId of the agent.
capturerInstructionTitle	String	Title of the instruction card displayed to the participant whose is capturing the photo.
capturerInstructionDescription	String	Description to be shown in the instruction card displayed to the participant who is capturing the photo.

5.1.8 CAPTURE IMAGE

The Welcome activity is designed to show a welcome message to the customer before proceeding with the actual KYC process. This activity can be configured as below.

Example:

```
{
  "id": "signatureImage",
  "activityType": "CaptureImage",
  "gatherFrom": ["customer"],
  "displayTo": ["agent"],
  "onActivityDataGathered": "<onActivityDataGathered webhook url>",
  "onActivityAction": "<onActivityAction webhook url>",
  "configuration": {
    "title": "Capture Specimen Signature",
    "description": "Please sign on a whitepaper and show it to the camera. Ensure that the signature is clearly visible and large enough before clicking on capture."
    "options": {
      "sourceType": "Camera",
      "caption": "Captured Specimen Signature",
      "capturerExternalId": "1",
      "instructionTitle": "Capturing your specimen signature",
      "instructionDescription": "Please hold the specimen signature straight and show to the camera",
      "capturerInstructionTitle": "Capture Customer's specimen signature ",
      "capturerInstructionDescription": "Make sure that the specimen signature is faced towards the camera, is being held in correct orientation and is clearly visible and legible"
    }
  }
}
```

JSON fields:

Field	Type	Description
id	String	A unique identifier (within the scope of the session) for this activity.
activityType	String	CaptureImage
gatherFrom	String[]	An array of string representing the roles of the participants from whom the image needs to be captured. For example, in case of Video KYC, you need to specify "gatherFrom": ["customer"]
displayTo	String[]	An array of string representing the roles of the participants who can see and review the image captured in this activity. For example, in case of Video KYC, you need to specify "displayTo": ["agent"]
onActivityDataGathered	String (optional)	A webhook URL to POST the data to, when data for the activity has been gathered. You will receive this data in real-time and will be able to perform your own validations and return the response in JSON format. Videoflo will display any validation errors or warning in the frontend and enable/disable the "continue" button based on the response received. If an empty response with HTTP status 200 is received, then the data is deemed acceptable, and the workflow will be allowed to continue normally.

onActivityAction	String (optional)	A webhook URL to POST the data to, when agent accepts / rejects data for the activity.
configuration	Object	CaptureImageConfig object.
title	String	Title to be displayed in the activity card shown to the customer.
description	String	Detailed message to be displayed in the activity card body shown to the customer. Note: You can add HTML markup to this field, and it will be rendered as HTML inside the card body. However, any scripts will be stripped out before rendering.
options	Object	ImageSourceConfig object
sourceType	String	Type of image source. Valid values are: • Camera • URL • Base64 (image encoded as base64 and included in the config payload) Upload (will display a file upload control)
value	String	A valid value based on the image source type specified. • URL: the URL of the image. Note: We do not support any form of authentication. The URL given here must be reachable from the internet and GET verb. • Base64: image encoded as base64. Note: The value should conform to the data: URI format and include header information on image format. For example: `data:image/jpeg;base64` • Camera: value is ignored Upload: value is ignored
caption	String	Caption to be displayed under the image thumbnail
displayTo	String[]	An array of roles to whom the image is visible within the activity card. For example, you can specify only the agent's external id and not include customer's id to hide the captured image from being visible to customer but display it to agent.
instructionTitle	String	Title of the instruction card displayed to the participant whose photo is being captured.
instructionDescription	String	Description to be shown in the instruction card displayed to the participant whose photo is being captured.
capturerExternalId	String	External ID of the participant who can click on the capture and compare button. In case of Video KYC, it will be externalParticipantId of the agent.
capturerInstructionTitle	String	Title of the instruction card displayed to the participant whose is capturing the photo.
capturerInstructionDescription	String	Description to be shown in the instruction card displayed to the participant who is capturing the photo.

5.2 DATA

All the data gathered across all the activities configured for a given session is encapsulated in a JSON object. This section describes the format of this JSON object.

Each key in the data object represents the activity itself. Furthermore, each key within an activity represents the participant from whom the data was gathered. The value of this "participant" key contains the actual object with the data.

Example data object:

```
{
  "<your activity id>": {
    "<participant id>": {
      "payload": {
        ... activity data here ...
      },
      "accepted": true,
      "acceptedBy": "<agent participant id>"
    }
  }
}
```

JSON fields:

Field	Type	Description
<your activity id>	Object	<your activity id> corresponds to the ID specified by you while configuring the activities for the session. It is NOT the activity type. This design allows you to use the same activity type multiple times within a session; for example, you can now use multiple CaptureImage activities within a session if you want to. Just give them different meaningful IDs.
<participant id>	Object	<participant id> corresponds to the Videoflo Participant ID from whom the data was gathered/captured. In case of Video KYC, this will be the customer’s participant id. The idea behind using participant id as key is to allow for gathering data from multiple participants as part of a workflow. This way, each participant’s data is encapsulated separately.
payload	Object	Contains the data gathered as part of the activity corresponding to the activityId key.
accepted	Bool	A Boolean value indicating whether the data gathered in the activity was accepted by the agent.
acceptedBy	String	Participant ID of the agent who "accepted" data for this activity. Note: If "accepted" is false, then this field denotes "rejectedBy”. We are looking to change this to a more neutral name. Suggestions are welcome! 😊

The following sub-sections detail the payload to expect based on the ActivityType.

5.2.1 WELCOME

Although you will find an object for Welcome activity, the object will be empty, since there is no data being gathered. This object will be deprecated and removed in a future release once we have preCallChecks implemented.

5.2.2 GEO LOCATION VERIFICATION

Example payload:

```
{
  "accuracy": 21,
  "results": []
}
```

JSON fields:

Field	Type	Description
accuracy	Float	Accuracy (in meters) of the geolocation lookup as reported by the browser.
results	GeocodingResponse[]	An array of geocoding response from Google Geocoding API. More details on response format can be found at https://developers.google.com/maps/documentation/geocoding/overview#GeocodingResponses

5.2.3 IP ADDRESS VERIFICATION

Example payload:

```
{
      "ip": "49.206.8.192",
      "is_eu": false,
      "city": "Bengaluru",
      "region": "Karnataka",
      "region_code": "KA",
      "country_name": "India",
      "country_code": "IN",
      "continent_name": "Asia",
      "continent_code": "AS",
      "latitude": 12.9634,
      "longitude": 77.5855,
      "postal": "560002",
      "calling_code": "91",
      "flag": "https://ipdata.co/flags/in.png",
      "emoji_flag": "IN",
      "emoji_unicode": "U+1F1EE U+1F1F3",
      "asn": {
            "asn": "AS24309",
            "name": "Atria Convergence Technologies Pvt. Ltd.  Broadband Internet
Service Provider INDIA",
            "domain": "actcorp.in",
            "route": "49.206.8.0/21",
            "type": "isp"
      },
      "languages": [
            {
```

```
                        "name": "Hindi",
                        "native": "हिन्दी"
                  },
                  {
                        "name": "English",
                        "native": "English"
                  }
            ],
            "currency": {
                  "name": "Indian Rupee",
                  "code": "INR",
                  "symbol": "Rs",
                  "native": "টকা",
                  "plural": "Indian rupees"
            },
            "time_zone": {
                  "name": "Asia/Kolkata",
                  "abbr": "IST",
                  "offset": "+0530",
                  "is_dst": false,
                  "current_time": "2021-07-25T23:31:49.216024+05:30"
            },
            "threat": {
                  "is_tor": false,
                  "is_proxy": false,
                  "is_anonymous": false,
                  "is_known_attacker": false,
                  "is_known_abuser": false,
                  "is_threat": false,
                  "is_bogon": false
            },
            "count": "10"
}
```

Detailed information on the fields is available at https://ipinfo.io/developers/data-types

5.2.4 Q&A

The payload will contain an array of Q&A response objects, one object per question asked.

Example payload:

```
[
  {
    "question": "Could you please confirm your current address?",
    "expectedAnswer": "#67, 7th A Cross, Attur Layout, Bangalore 560064",
    "isAnswered": true,
    "isAnswerCorrect": true
  },
  ... responses for other questions ...
]
```

JSON fields:

Field	Type	Description
question	String	The question that was asked.

expectedAnswer	String	Expected answer for this question, as specified in the activity configuration while creating the session.
isAnswered	Boolean	Whether or not the question was answered by the customer.
isAnswerCorrect	Boolean	Whether or not the answer given was right.
Note: This object will be extended further and will contain more data once speech recognition is implemented.		

5.2.5 MATCH HEAD POSES

The payload will contain an array of MatchHeadPoseResponse objects, one for each head pose that was prompted. For each head pose, multiple images are captured and submitted to avoid false positives. The number of images captured is controlled by noOfAttempts parameter in the activity configuration.

Example payload:

```
[
  {
    "pose": "facedown",
    "response": {
      "success": true,
      "errorCode": null,
      "errorMessage": null,
      "results": [
        {
          "imageId": 1,
          "result": true,
        },
        {
          "imageId": 2,
          "result": false,
          "errorCode": 1001,
          "errorMessage": "Face Not Found",
        },
        ... result for all other images ...
      ]
    }
  }
  ... data for other poses ...
]
```

JSON fields:

Field	Type	Description
pose	String	Name of the pose, whose results this object represents.
response	Object	MatchHeadPoseResponse object
success	Boolean	A Boolean value indicating whether the overall match head-pose test was successful or not. If 2/3rds of the head-poses match correctly, it is deemed that the test was passed by the customer.
errorCode	Int	Optional and nullable. Will contain error code only in case of an error.
errorMessage	String	Optional and nullable. Will contain user friendly error message only in case of an error.
results	ResultItem[]	
imageId	Int	1-based serial number of the image, within this head-pose.

result	Boolean	A Boolean value indicating whether this image is matching the expected head pose specified in the request.
errorCode	Int	Optional and nullable. Will contain error code only in case of an error. For example, 1001 for FACE_NOT_FOUND
errorMessage	String	Optional and nullable. Will contain user friendly error message only in case of an error. For example, “Face not found”

5.2.6 FACE RECOGNITION

The payload will contain an object with both image1 & image2 represented in base64 along with the face matching result.

Example payload:

```
{
  "image1": {
    "base64Image": "data:image/jpeg;base64,"
  },
  "image2": {
    "base64Image": "data:image/jpeg;base64,"
  },
  "faceMatchingResult": {
    "success": true,
    "isMatching": true,
    "distance": 0.8605557084083557,
    "errorCode": null,
    "errorMessage": null,
    "confidence": 68
  }
}
```

JSON fields:

Field	Type	Description
image1	Object	Image object
base64Image	String	Base64 representation of the image buffer
image2	Object	Image object
base64Image	String	Base64 representation of the image buffer
faceMatchingResult	Object	Object containing the face match api response
isMatching	Boolean	A Boolean value indicating whether the face found in the two images are of the same person.
distance	float	The Euclidean distance between the two faces. This should be below 1.05 for the faces to be considered same.
confidence	Int	A number between 0 to 100 representing the confidence percentage of the model.
Success	Boolean	A Boolean indicating whether the face match API call was successful. If this is false, it usually indicates a) problem with decoding the image, or b) face match api error
errorCode	Int	Optional and nullable. Will contain error code only in case of an error.
errorMessage	String	Optional and nullable. Will contain user friendly error message, only in case of an error.

5.2.7 PAN RECOGNITION

The payload will contain the original image submitted along with the result received from the PAN recognition API.

Example payload:

```
{
  "image": "data:image/jpeg;base64,",
  "result": {
    "success": true,
    "pan_num": "<OCR result for PAN number field>",
    "name": "<OCR result for name field>",
    "father_name": "<OCR result for father name field>",
    "dob": "<OCR result for dob field>",
    "face_image": "data:image/jpeg;base64,",
    "pan_image": "data:image/jpeg;base64,",
    "signature_image": "data:image/jpeg;base64,"
    "errorCode": null,
    "errorMessage": null
  }
}
```

JSON fields:

Field	Type	Description
Image	String	Original image captured in base64 format
Result	Object	Response object returned by the PAN Recognition API
success	Boolean	A Boolean indicating whether the PAN ocr api call was successful. If this is false, it usually indicates a) problem with decoding the image, or b) PAN ocr api error
pan_num	String	PAN number field as recognized by OCR
name	String	Name field as recognized by OCR
father_name	String	Father name field as recognized by OCR
dob	Date	DOB field as recognized by OCR
face_image	String	Face Image extracted out of the PAN card, in base64
pan_image	String	Cropped PAN card from the original image, in base64. Since the original image can contain the background, you can retrieve the cropped PAN card image using this field.
signature_image	String	Signature image extracted out of the PAN card, in base64.
errorCode	Int	Optional and nullable. Will contain error code only in case of an error.
errorMessage	String	Optional and nullable. Will contain user friendly error message, only in case of an error.

5.2.8 CAPTUREIMAGE

The payload will contain an object encapsulating the base64Image value of the captured image.

Example payload:

```
{
  "base64Image": ""
}
```

6. CONTACT DETAILS

Raghavendra K, Head, Engineering raghuk@botaiml.com +91 73489 44871	Mahanthesha H Product Manager mahantheshah@botaiml.com +91 96114 84959	Lakshmana B Poojary Team Lead lakshmanabp@botaiml.com +91 78922 65321