VideocallManager Integration Guide

release 2021Q1 (v2.0.8)

Release	Date	Description	Author	Reviewer	Approver
2.0	30-03-2021	VideocallManager v2.0.8	OLN	ACR	MSY

1. Introduction

VideocallManager is a frame built as a stand-alone solution with an easy plug-and-play integration.

This video call has two main actors:

- Agent: boi-Das user (not covered is this document)
- Client: end-user (same user who has previously done the onboarding process)

VideocallManager comprises the technologies for the reception and transmission of audio-video signals by users at different locations, for communication between Agent and Client in real time.

Veridas has added video call functionality to the available product range. This functionality allows customers to use an online video identification technology. In this way, Veridas offers technology to cover any of the scenarios envisaged in the SEPBLAC regulation in Spain, CNBV in Mexico, or other regulations.

Additionally, Veridas has certified through Dekra that its online video identification product (videocall) complies with the requirements of SEPBLAC and with article 24.1d of eIDAS. This makes it possible to prove that Veridas' identity verification technology offers a level of security equal to or higher than that of physical processes.

The use of this service requires the use of vali-Das or XpressID. It is also necessary to have contracted the boi-Das service (case management tool).

2. What 's new?

This section presents the changes that have been introduced in the first version of the VideoCall standalone service, released in 2021Q1

2.1. Improved

- Improved UI of transition steps by changing colors and the contrast between the background and the text.

2.2. Fixed

- Pre-videocall button in Safari mobile has been properly centered on the screen

3. Browser compatibility

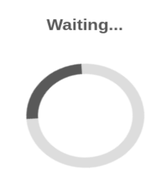
Video Call has been designed to maximize compatibility and performance across a broad spectrum of devices and browsers. The chromium-based browsers work fine.

Desktop devices			Mobile devices		
Browser Name	Minimum Version	Current Version[1]	Browser Name	Minimum Version	Current Version[1]
Chrome	57	89	Chrome for Android	57	85
Opera	44	71	Opera Mobile	46	79
Edge	16	86	iOS Safari	11.2	14
Safari	11.2	14			
Firefox	52	83			

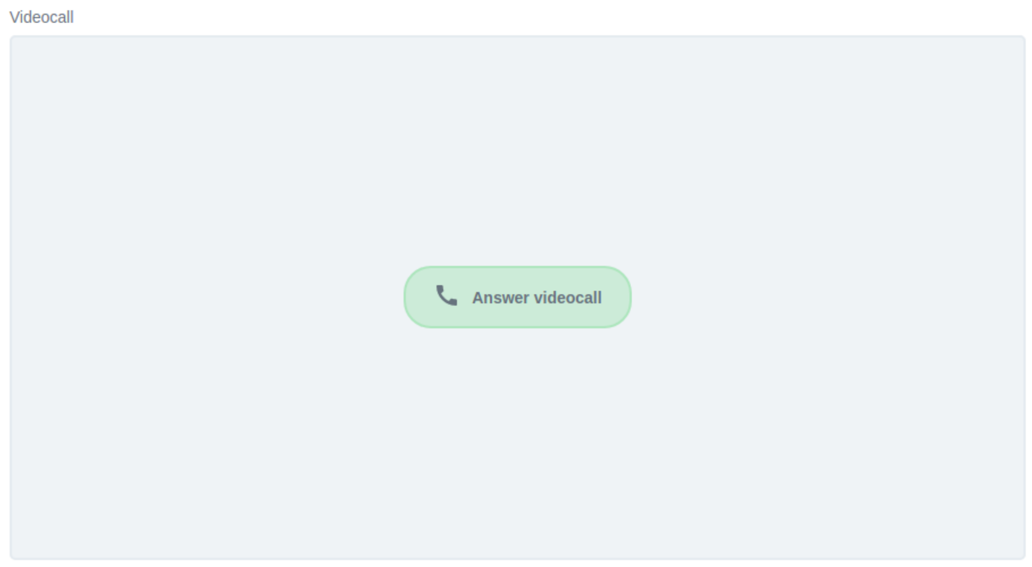
4. Video call scheme

The video call has these steps:

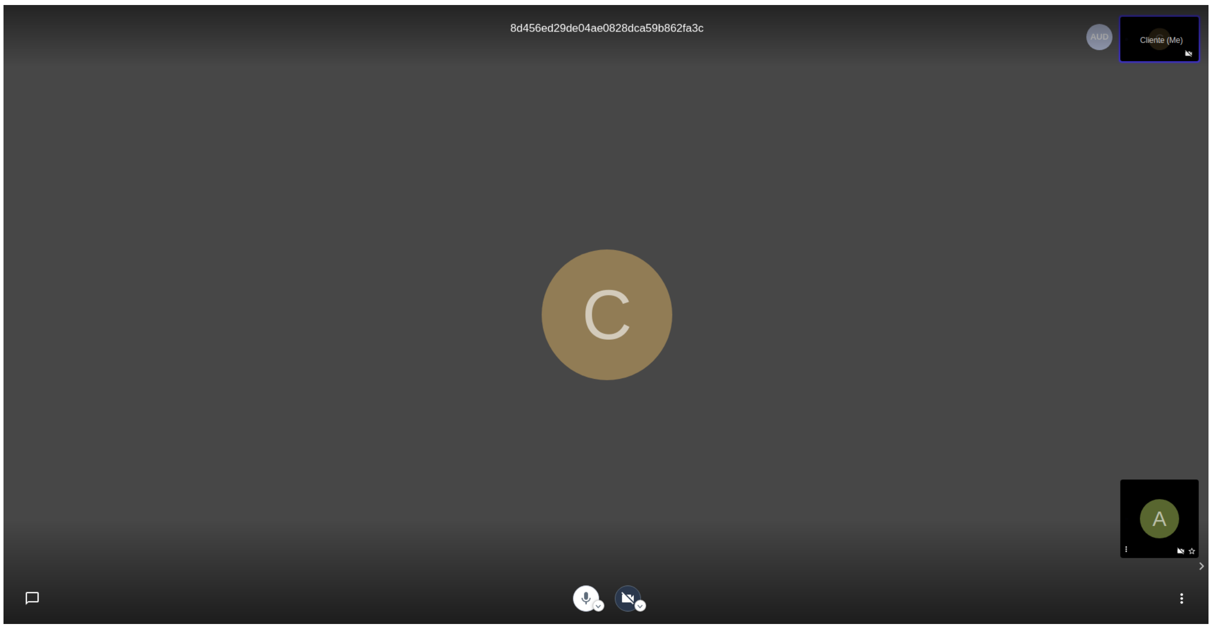
1. At a certain point, the client accesses the video call URL. At first, the client does not enter the video call, but is redirected to a waiting room, where he will wait for an agent to connect.



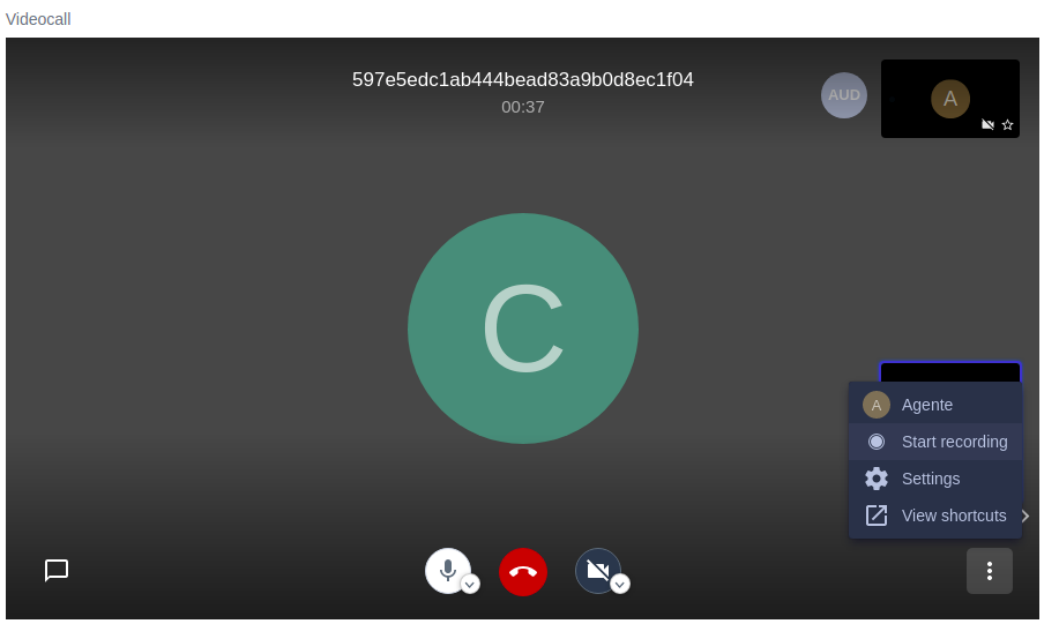
2. Once the client is in the waiting room, an agent realizes that (boi-Das warns him) and accesses the video call (the agent accesses directly without waiting in the waiting room).



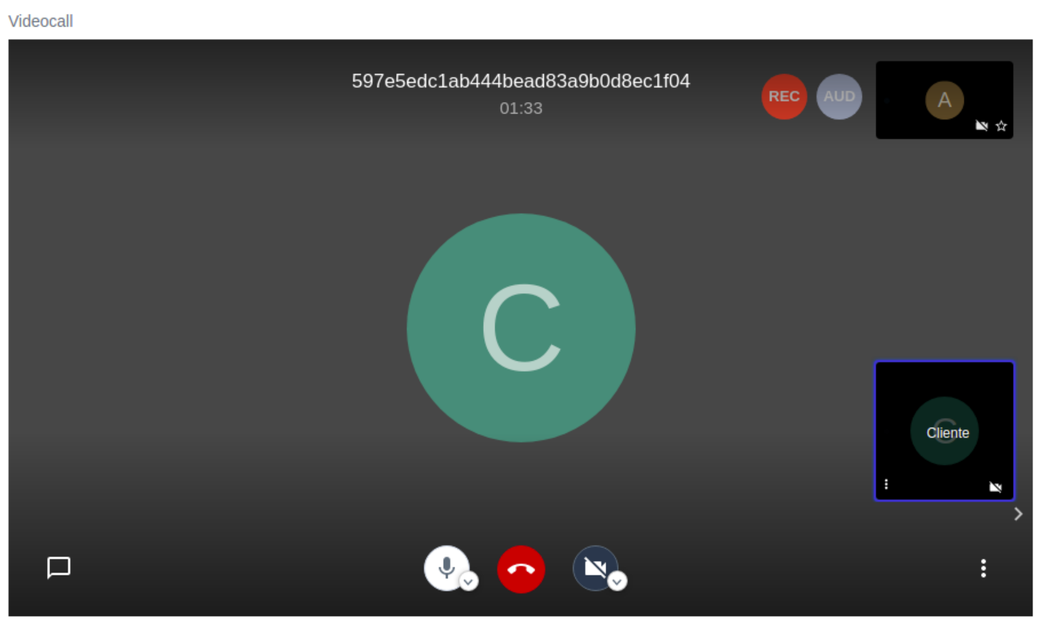
3. That's when the client is redirected to the video call.



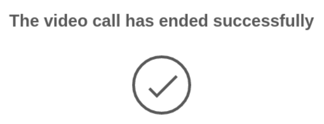
4. Once both are on the video call, the agent can talk to the client freely and he can start recording the meeting.



5. The agent can end the video call at any time with a button (only the agent has this button, so the client can't use it).

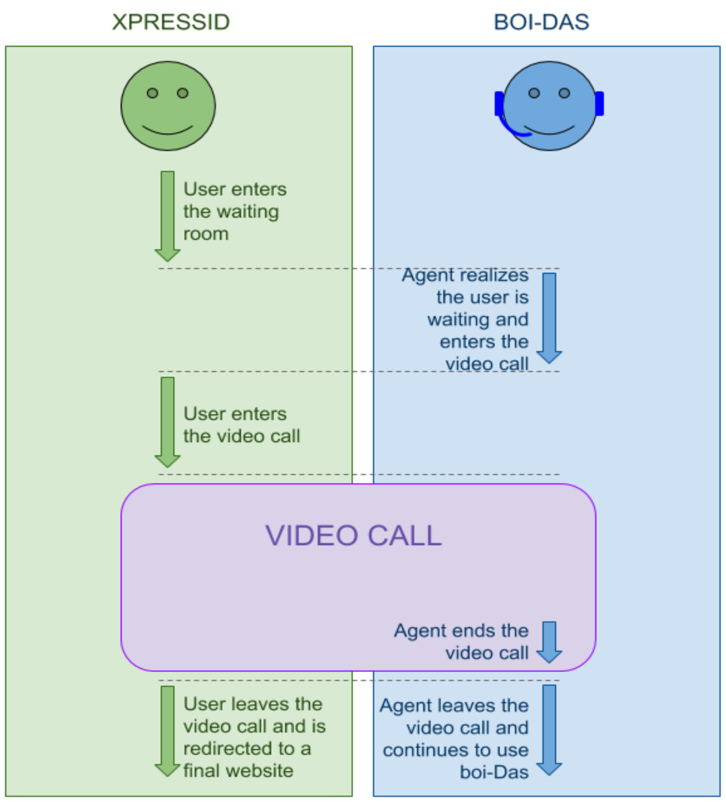


6. When the agent has finished the video call, the client will be directed to another website where he will see a message saying the video call has ended successfully.



The video call has ended successfully

Below is a diagram of the video call flow between client and agent.


XPRESSID
BOI-DAS
User enters the waiting room
Agent realizes the user is waiting and enters the video call
User enters the video call
VIDEO CALL
Agent ends the video call
User leaves the video call and is redirected to a final website
Agent leaves the video call and continues to use boi-Das

5. Integration

5.1. Client credentials

To use VideocallManager, the following data will be provided by Veridas:

1. VIDEOCALL_URL, which varies depending on the environment (sandbox or production), region, etc
2. CLIENT_ID (unique for each client)
3. CLIENT_SECRET (unique for each client)

The “CLIENT_ID” and “CLIENT_SECRET” will be used to interact with the VideocallManager Authentication API REST.

Moreover, to comply with Veridas security measures, the following information should be provided by the client:

- Client server IPs where VideoCall is planned to be embedded

Note: this security measure only applies to production environments and not to the sandbox.

5.2. Authentication API

The first step to embed VideocallManager in an iframe, is to obtain an access token. For this purpose, VideocallManager Authentication API must be used:

Request:

- URL: https://*VIDEOCALL_URL*/o/token/
- Method: POST
- Content-Type: multipart/form-data (or application/x-www-form-urlencoded)
- Data:
  - 'grant_type': 'client_credentials'
  - 'client_id': *CLIENT_ID*
  - 'client_secret': *CLIENT_SECRET*
- Actually client_id and client_secret parameters can be also sent in the 'Authorization' header instead of the body.

Response:

- Status: 200
- Data: {'access_token': *ACCESS_TOKEN*, 'expires_in': 3600, 'token_type': 'Bearer', 'scope': 'read write groups'}

The response will provide an access token that will allow VideocallManager to be embedded. This access token will be used as a query parameter when embedding the iframe, and it's unique for each final users (two different user should not use the same token).

Here are some examples of this request using CURL:

```
# multipart/form-data
curl --request POST 'https://VIDEOCALL_URL/o/token/' --form
'client_id=CLIENT_ID' --form 'client_secret=CLIENT_SECRET' --form
'grant_type=client_credentials'
_
# multipart/form-data with Authorization header
curl --request POST 'https://VIDEOCALL_URL/o/token/' --user
'CLIENT_ID:CLIENT_SECRET' --form 'grant_type=client_credentials'

# application/x-www-form-urlencoded
curl --request POST 'https://VIDEOCALL_URL/o/token/' --header
'Content-Type: application/x-www-form-urlencoded' --data-urlencode
```

```
'client_id=CLIENT_ID' --data-urlencode 'client_secret=CLIENT_SECRET'
--data-urlencode 'grant_type=client_credentials'

# application/x-www-form-urlencoded with Authorization header
curl --request POST 'https://VIDEOCALL_URL/o/token/' --header
'Content-Type: application/x-www-form-urlencoded' --user
'CLIENT_ID:CLIENT_SECRET' --data-urlencode
'grant_type=client_credentials'
```

The token returned in the response has an expiration date according to these rules:

1. When the video call process is finished, the token expires (so it can not be used again).
2. If the video call process is not completed within 1 hour (from the token creation time), the token expires automatically.
3. [Optional] It's also possible to revoke the token manually using an endpoint.

To revoke manually an access token, this endpoint must be used:

Request:

- URL: https://*VIDEOCALL_URL*/o/revoke_token/
- Method: POST
- Content-Type: multipart/form-data (or application/x-www-form-urlencoded)
- Data:
  - 'token': *ACCESS_TOKEN_TO_REVOKE*
  - 'client_id': *CLIENT_ID*
  - 'client_secret': *CLIENT_SECRET*
- Actually client_id and client_secret parameters can be also sent in the 'Authorization' header instead of the body.

Response:

- Status: 200
- No data

Here are some examples of this request using CURL:

```
# multipart/form-data
curl --request POST 'https://VIDEOCALL_URL/o/revoke_token/' --form
'client_id=CLIENT_ID' --form 'client_secret=CLIENT_SECRET' --form
'token=ACCESS_TOKEN_TO_REVOKE'

# multipart/form-data with Authorization header
curl --request POST 'https://VIDEOCALL_URL/o/revoke_token/' --user
```

```
'CLIENT_ID:CLIENT_SECRET' --form 'token=ACCESS_TOKEN_TO_REVOKE'

# application/x-www-form-urlencoded
curl --request POST 'https://VIDEOCALL_URL/o/revoke_token/' --header
'Content-Type: application/x-www-form-urlencoded' --data-urlencode
'client_id=CLIENT_ID' --data-urlencode 'client_secret=CLIENT_SECRET'
--data-urlencode 'token=ACCESS_TOKEN_TO_REVOKE'

# application/x-www-form-urlencoded with Authorization header
curl --request POST 'https://VIDEOCALL_URL/o/revoke_token/' --header
'Content-Type: application/x-www-form-urlencoded' --user
'CLIENT_ID:CLIENT_SECRET' --data-urlencode
'token=ACCESS_TOKEN_TO_REVOKE'
```

5.3. Iframe integration

Once the access token is obtained, the iframe can be embedded into the HTML code in this way:

```
<iframe
 id="videocall-iframe"
 allow="camera; microphone;"
 style="width:100%;height:100%;"
src="https://VIDEOCALL_URL/call?access_token=ACCESS_TOKEN&validation_id=
VALIDATION_ID">
</iframe>
```

Note that:

- The `URL` is slightly different that the one at section 5.2 (note the "/call").
- The `ACCESS_TOKEN` is a new authentication token obtained with the same requests as in section 5.2
- The `VALIDATION_ID` is the *`validationId`* parameter provided by vali-Das during a validation process, which is the unique identification of the validation process.

There are two ways to get the VALIDATION_ID, one way is using the XpressId and the other way is vali-Das.

if using XpressID, **please check the XpressID documentation in section 4.6 Receiving data from iframe,** that VALIDATION_ID is available to be used.

Otherwise if using vali-Das, **please check the vali-Das documentation in sections 6.2 Create a new validation and 6.3 Create a new validation and upload documents,** this

VALIDATION_ID to be valid and downloaded by Boidas, the VALIDATION_ID has to be confirmed, **please check the vali-Das documentation in section 6.15 Confirm validation.**

When the client enters this website, he will see a waiting room until an agent joins the video call at boi-Das. To minimize the time an end-user waits, its validation process will take precedence in the boi-Das validations queue, so that agents always are dispatched with validations pending of video calls rather than any other validation reviews.

An event (see section 5.5.1.3) will be sent to the client when the agent joins the video call, with this event you can do a push notification for the client.

When the boi-Das agent finishes the video call, a message will be displayed to the client saying that the process has been completed successfully.

Note that since it's an iframe, there is no problem related to CORS (cross-origin resource sharing) and the VideocallManager URL can be embedded directly into the iframe itself.

This is the only needed code in order to make VideocallManager to work properly. Note that "ACCESS_TOKEN" is unique for each final user and it's going to be different every time VideocallManager is embedded.

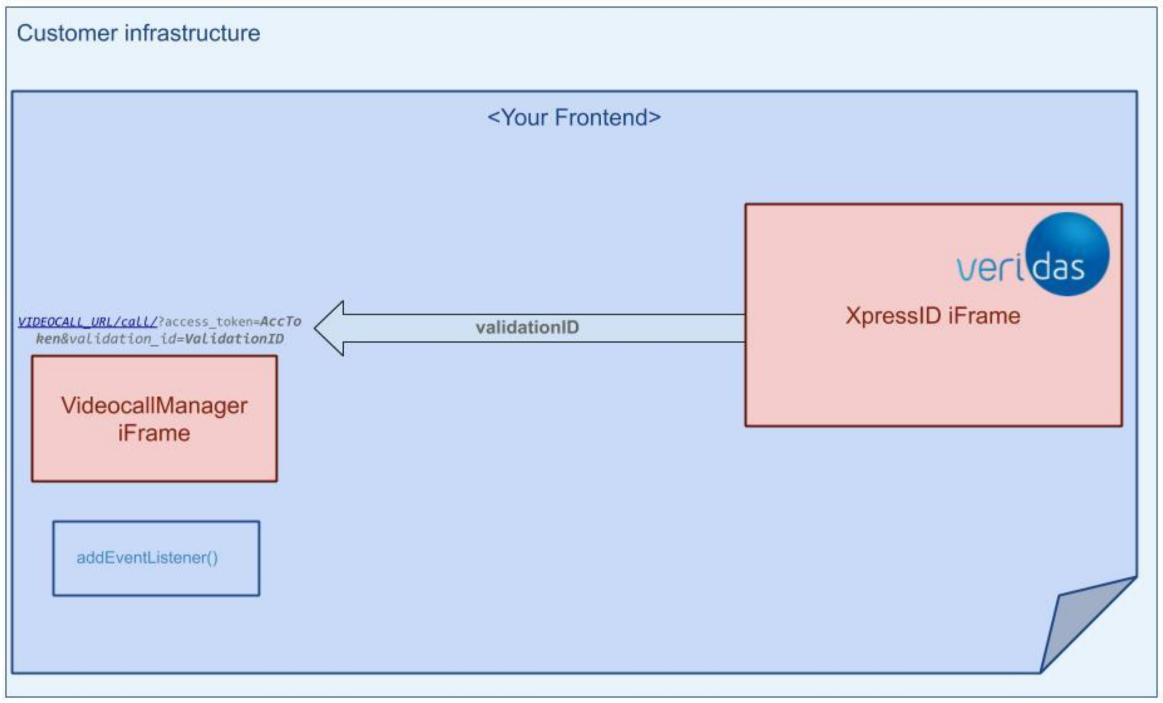
The web that embeds the iframe must be served over HTTPS, otherwise some browsers may experience problems accessing the camera and the microphone.

5.4. Architecture overview

There is an interaction with XpressID if this product is used:

1. ValidationId: This value is returned by XpressID after a validation process.

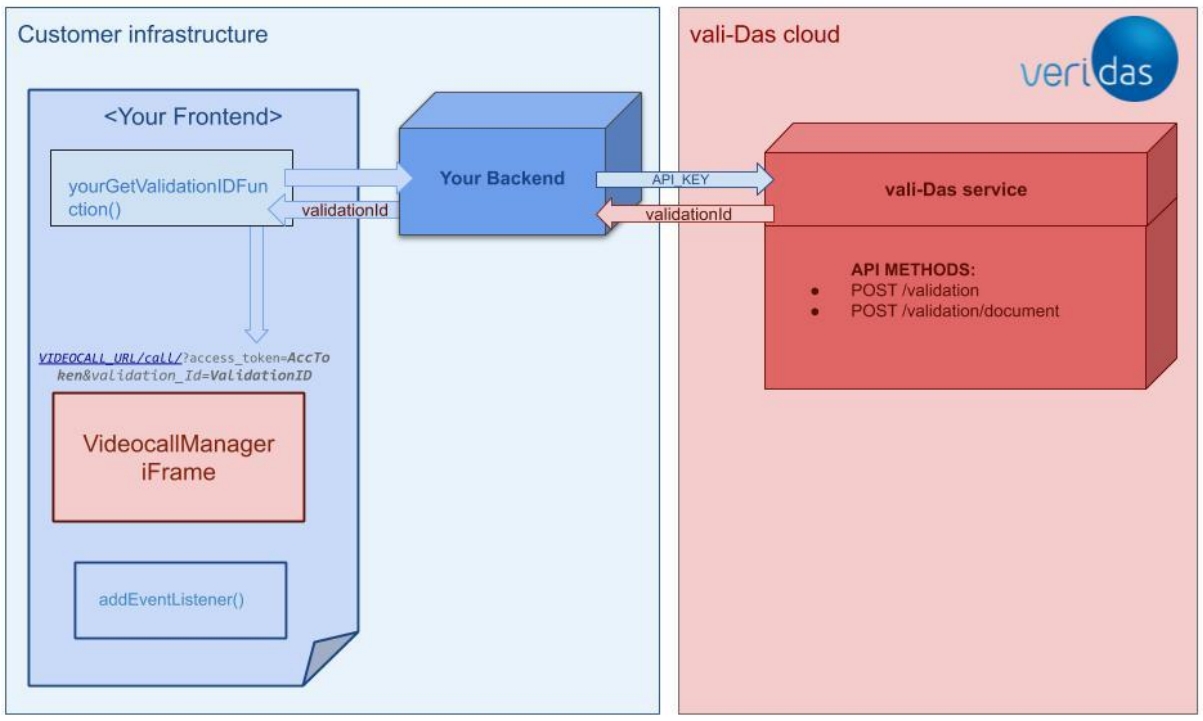
This must be the architecture of the web application using XpressID:



There is an interaction with vali-Das if this product is used:

1. ValidationId: This value is returned by the vali-Das API.

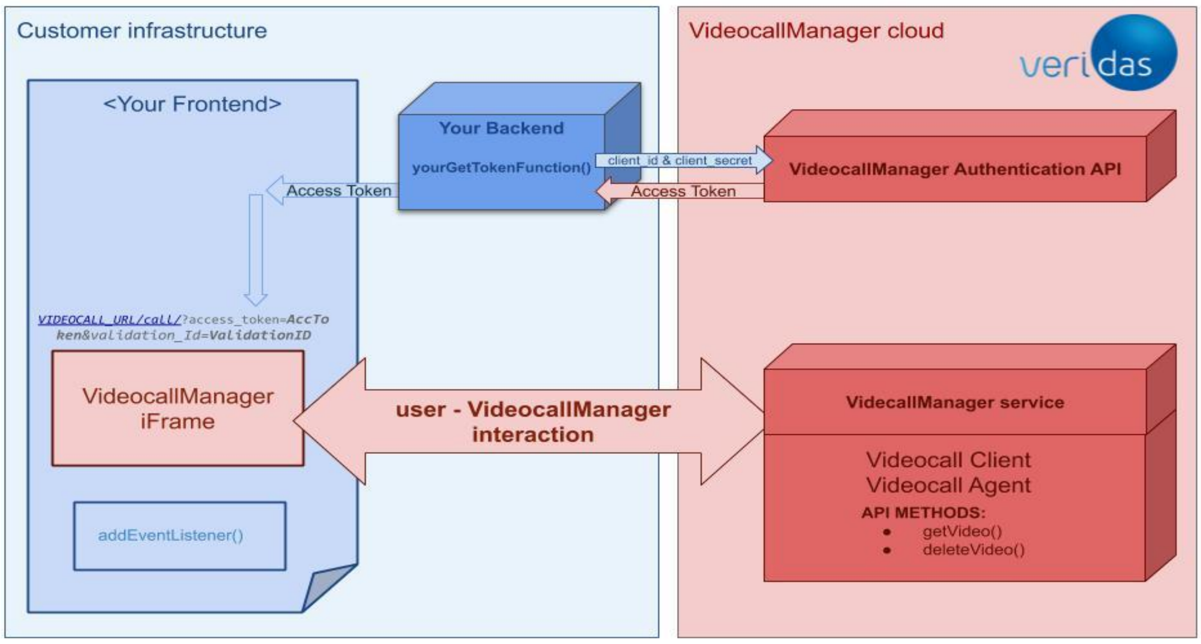
This must be the architecture of the web application using vali-Das:



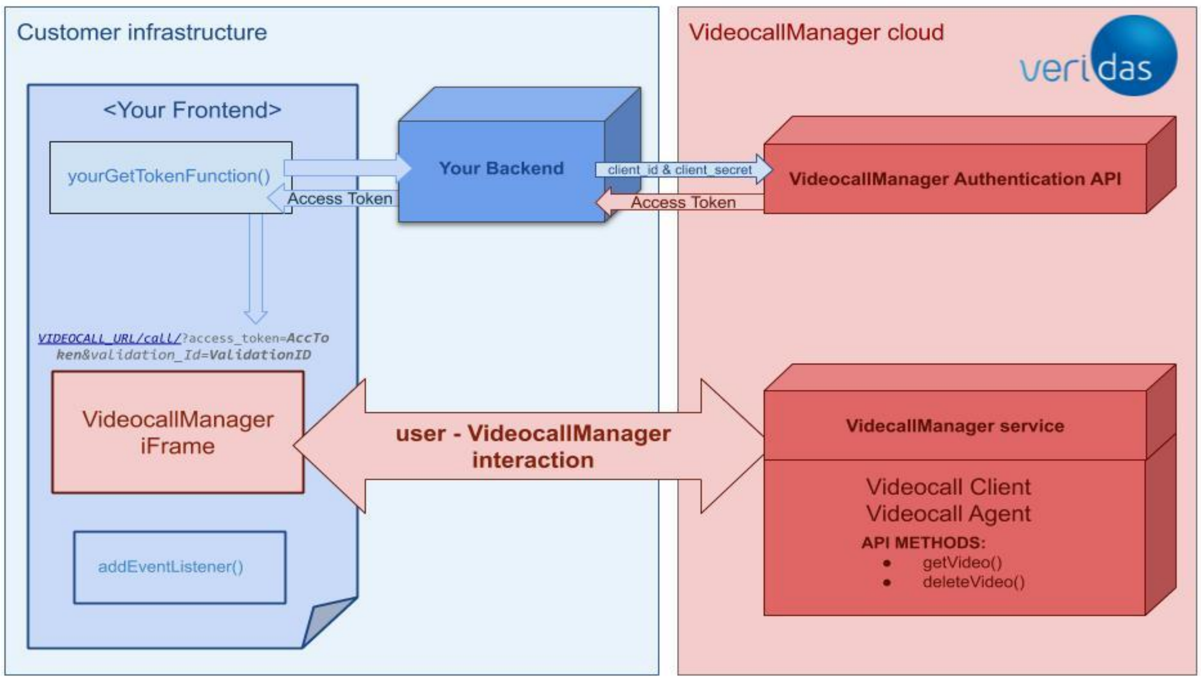
There are two interaction points with VideocallManager:

1. Authentication API: these requests must be made from the application backend. This is the only place where the variables “CLIENT_ID” and “CLIENT_SECRET” are needed.
2. Iframe: this is where VideocallManager is embedded. Since it’s HTML code, it must be used in the application frontend. It’s very important not to use the variables “CLIENT_ID” and “CLIENT_SECRET” within the frontend, because this way an end user could see them.

This must be the architecture of the web application using VideocallManager:



It is also possible to have “yourGetTokenFunction()” in the frontend, however, it must send the request to the backend and the backend must redirect the request to Veridas cloud, adding the “CLIENT_ID” and the “CLIENT_SECRET”:



5.5. Receiving data from iframe

When the process is over, a postMessage is sent by VideocallManager to the parent website where it has been embedded. If desired, it is possible to listen to that message with an event listener:

```
window.addEventListener("message", receiveMessage, false);
function receiveMessage(event) {
    if(event.origin.includes("videocall")     &&     event.data.code     ==
"VideocallCompleted") {
  var dataFromVideoCall = event.data;
  // do something
 }
}
```

This event returns a code “VideocallCompleted” that indicates that the video call process has been successfully completed.

Throughout the process, VideocallManager needs to pass some data to the parent website where it’s embedded. This data can be of two types:

- Informational data
- Error data

Note that with the “`event.origin.includes("videocall")`“ directive we ensure the event comes from VideocallManager, otherwise, the event could be coming from another part of the website.

The returned data (“`dataFromVideoCall`“ in the example), will always have the same structure: it’s a JavaScript object with these fields:

- type: event type. Only two available values: “info” (for informational data) and “error” (for error data).
- code: unique event code
- message: event text explanation

To distinguish between the different messages, it is necessary to check the “`event.data.code`“.

5.5.1. Informational data

These are the events that send informational data to the parent website where VideocallManager is embedded.

5.5.1.1. VideocallCompleted

This event occurs when the video call has just been ended. The structure of the received data is:

```
{
 type: "info",
 code: "VideocallCompleted",
 message: "The video call has been successfully completed"
}
```

5.5.1.2. VideocallInitiated

This event occurs when the video call has just been initiated. The structure of the received data is:

```
{
 type: "info",
 code: "VideocallInitiated",
 message: "The video call has just been initiated"
}
```

5.5.1.3. VideocallLoading

This event occurs when the agent joins the video call. The structure of the received data is:

```
{
 type: "info",
 code: "VideocallLoading",
 message: "The video call is loading"
}
```

5.5.2. Error data

When an error occurs, VideocallManager tries to handle it so that it is transparent to the user. However, there are certain errors which can not be handled by VideocallManager. In these cases, it will display an error message to the user, and send a message to the parent website via postMessage.

Note that although the error code is immutable.

Here below there is a list with all unhandled errors.

5.5.2.1. VideocallNotFound

This error occurs when VideocallManager tries to access to a non-existent video call. The structure of the received data is:

```
{
 type: "error",
 code: "VideocallNotFound",
 message: "Video call not found"
}
```

5.5.2.2. VideocallServerNotOperational

This error occurs when the video call server is not operational at that time. The structure of the received data is:

```
{
 type: "error",
 code: "VideocallServerNotOperational",
 message: "Video call server not operational"
}
```

6. Security

6.1. IP restriction

As mentioned above, when using the production environment, VideocallManager has a whitelist with the server IPs where it can be embedded, so production IPs must be sent to Veridas. Otherwise, the connection to the authentication API will be blocked.

6.2. Used ports

To use the video call, access to all these ports is required:

- TCP/443
- UDP/10000
- TCP/4443

This access is required not only for the agent (boi-Das user), but also for the Client user.

6.3. Recommendations

Since VideocallManager does not provide any manual login step (because the authentication step is done from the backend), it is recommended to put some security measures to avoid abusive uses.

Some of these security measures could be:

- A login step before accessing VideocallManager
- A captcha step before accessing VideocallManager
- A rate limit to block multiple requests from the same IP
- DDoS mitigation measures

7. Annex 1: Changelog History

2020Q4 (v2.0.7)

Added

- Videocall service segregated from XpressID as a separate service
- Event VideocallLoading. (see section 5.5.1.3)

2020Q3 (v1.1.0)

Added

- First productive version