Patientus Video Frontend Library

Table of content 🗒️

Description

This library is developed by the Docplanner Telemedicine team.

The main goal of this library is to provide a seamless integration of our teleconferencing solution to the Docplanner SaaS application, as well as to the software of our white label customers.
Once it is integrated, participants (doctors and patients) can join in the secure video calls based on the given appointment information. The video calls could be between a doctor and a patient, whereas it is also possible multiple participants to join in the video calls.
In addition to the standard teleconferencing features, it also provides extra capabilities such as **End-to-End Encrypted File Sharing** and **Note-Taking**.

Aiming to perform secure peer-to-peer teleconferencing, without requiring that the user install plug-ins or any other third-party software, the library utilizes Vonage Video API with its underlying WebRTC API.

The library was written with Angular CLI and supports **Angular v13+**.
There is also a custom element (web component) version built as a wrapper around Angular library and that supports JavaScript applications written with any frameworks other than Angular. The library packages are published to Patientus' private npm registry and could be installed only with provided credentials by Patientus.

How it works

With the initialization of the library, a request is sent to the Telemedicine API (or to the wrapper API created by our backend SDK) via sessionApiEndpoint specified in the library configuration to fetch Vonage session credentials. Upon the request, API checks if the requested session with the unique identifier (participant token or appointment id) was already created. In case the session exists, API returns existing session credentials. Otherwise, before returning, it will first create the session in the Vonage cloud. After retrieval of session

credentials, it attempts to connect to the session. When connection is established, it starts interacting with the session via a persistent event (or signaling) connection (WebSocket). At this point client is ready to initialize a WebRTC connection directly with other clients connected to the session. At last, upon a successful WebRTC connection between clients, they begin sending and receiving media streams.

In case of inability to connect directly to their peer, mostly because peers are located behind a firewall, media streams are transferred through **TURN servers**.
On the other hand, **IP Proxy** allows peers to hide their public IP address and not directly communicate with Vonage API by forwarding the requests through our custom server. Besides, the network restrictions scenario, while establishing peer-to-peer connections we encourage usage of TURN servers and IP Proxy in order to ensure a secure communication between participants.
Likewise, those are some requirements to obtain a data security certification for online service providers in Germany (See Datenschutz-Gütesiegel ips).

Features

Core features

- 1-to-1 and multi-participant video calls
- Pre-join screen to adjust the camera and mic
- Network and audio/video quality tests
- In-call device settings with camera preview
- Screen sharing with participants
- Peer-to-peer sessions with custom TURN servers (relayed mode)
- Routed sessions via Vonage's Media Servers (routed mode)
- IP Proxy to forward Vonage API requests

Extra features

- End-to-End Encrypted File Sharing (1:1)
- Note Taking

Upcoming features

- *Multi-language translations*
- *Custom theming*
- *E2EE File Sharing (Multi-user)*
- *Adjustable layouts*
- *In-call appointment information*
- *In-call messaging with participants*
- *Host controls for doctors*
- *Blurry and other video effects*

Installation

Package

In order to be able to download Patientus npm packages from our private npm registry, .npmrc file should be created as following:

- Create a new file in your project root folder and name it .npmrc
- Copy the contents below to the file

```
@patientus:registry=https://packages.patientus.de/repository/de.patientus.npm
/
//packages.patientus.de/repository/de.patientus.npm/:_authToken=$YOUR_AUTH_TO
KEN
```

Replace **$YOUR_AUTH_TOKEN** with your private Patientus npm token provided by us. Patientus scoped packages are now automatically installed from the Patientus registry.

Angular

Install the library package

```
npm i @patientus/video-lib-angular
```

The library utilizes Angular Material UI components, if your application doesn't include it, please install it

```
npm i @angular/material @angular/cdk
```

Installing other peer dependencies

```
npm i @opentok/client opentok-network-test-js opentok-layout-js
@ngrx/component-store @ludovicm67/lib-filetransfer
```

Custom Element

Custom element version of the library could be used in JavaScript apps build with any frameworks such as Vue or React

Install the library package

```
npm i @patientus/video-lib-wc
```

Usage

Angular

There are a handful of setup steps to go through. Generally, the steps are:

- Installing the library and peer dependencies (see above).
- Copying static library assets and serving them from the specified path
- Importing global styles and fonts
- Importing the PatientusVideoModule into your Angular project
- Including PatientusVideoComponent with the configuration input in your template (see docs below)

Copying static library assets

Use glob pattern to copy static assets to your projects output folder

```
{
  "architect": {
    "build": {
      "builder": "@angular-devkit/build-angular:browser",
      "options": {
        "assets": [
          {
            "glob": "**/*",
            "input": "./node_modules/@patientus/video-lib-angular/assets",
            "output": "./assets"
          }
        ]
      }
    }
  }
}
```

Note: In case of usage of an assets-path other than 'assets', it should be specified in the configuration as absolute path.

Importing font family

```
@import "@patientus/video-lib-angular/assets/style/fonts.css";
```

Alternatively you could import Roboto Regular 400, Regular 400 Italic, Bold 700 and Bold 700 Italic fonts from Google Fonts CDN

```
<style>
    @import url('https://fonts.googleapis.com/css2?
family=Roboto&display=swap');
</style>
```

Note: You could skip importing Roboto fonts in case your project already includes them.

Importing global styles

Global stylesheet should be imported in the host component.
Alternatively it could be imported in angular.json

```
@import "@patientus/video-lib-angular/assets/style/global.css";
```

Importing the PatientusVideoModule into your Angular project

```
import { PatientusVideoModule } from '@patientus/video-lib-angular';

@NgModule({
  imports: [PatientusVideoModule],
})
export class YourAppModule {}
```

Including PatientusVideoComponent

```
<patientus-video [config]="config"></patientus-video>
```

Important note: Changes to config are ignored after they are initially set. This is because the config is passed into the PatientusVideoComponent constructor, so they can't be changed in any case. So, make sure the config object exists before the PatientusVideoComponent is created. You'll want to create the object in ngOnInit or hide the DOM element with *ngIf until you can create the config object.

Types

Type definitions could be imported from `@patientus/video-lib-angular/types`.

Examples

Below you can see an example to start a video call with the specified configuration including appointment object

```
import { Component } from '@angular/core';
import { PatientusVideoConfig } from '@patientus/video-lib-angular';
import { Appointment, ExternalUrl, Feature, ParticipantGender,
ParticipantType, PatientusVideoConfig, TurnConfig } from '@patientus/video-
lib-angular/types';

@Component({
    selector: 'app-example-component',
    template: `
    <patientus-video *ngIf="config" [config]="config"
(disconnected)="onCallEnd()" (canceled)="onCancel()"></patientus-video>
  `,
})
export class AppExampleComponent {
    appointment: Appointment = {
        topic: 'appointment-topic',
        startDateTime: '2022-10-15 17:00:00',
        endDateTime: '2022-10-15 17:30:00',
        participant: {
            id: 'doctor-id',
            title: 'Dr.',
            firstName: 'Max',
            lastName: 'Mustermann',
            gender: ParticipantGender.MALE,
            email: 'max.mustermann@dummy.de',
            type: ParticipantType.DOCTOR,
            imageUrl: 'path-to/doctor-avatar-image-url.jpg',
            specialities: [
                {
                    id: 123,
                    itemName: 'Allergologie'
                }
            ]
        },
```

```
        otherParticipants: [
            {
                id: 'patient-id',
                firstName: 'Jane',
                lastName: 'Doe',
                gender: ParticipantGender.FEMALE,
                email: 'jane.doe@dummy.de',
                type: ParticipantType.PATIENT,
                imageUrl: 'path-to/patient-avatar-image-url.jpg',
                inviteUrl: 'path-to/invite-url'
            }
        ]
    };



    turnConfig: TurnConfig = [
        { urls: 'stun:turn.example.de:443' },
        { urls: 'turns:turn.example.de:443', credential: 'example', username:
'example' }
    ];

    feature: Feature = {
      dataPrivacy: true,
    };

    externalUrl: ExternalUrl = {
        dataProcessTerms: '/path-to/data-process-url',
        terms: '/path-to/terms-url',
        dataSecurity: '/path-to/dataSecurity-url',
    };

    config: PatientusVideoConfig = {
        sessionApiEndpoint: '/example-session-api-endpoint',
        jwt: this.exampleService.jwt,
        appointment: this.appointment,
        ipProxy: 'example-ip-proxy',
        turnConfig: this.turnConfig,
        logoUrl: '/path-to-your-app-assets/your-logo.png',
        feature: this.feature,
        externalUrl: this.externalUrl

    };

    onCallEnd() {
        console.log('user ended the call, do something');
    }

    onCancel() {
        console.log('user exits, do something');
    }
}
```

Vue (Custom Element)

Copying static library assets

In your webpack configuration 'copy-webpack-plugin' plugin could be used to copy static assets.
Please note that this must be set if any path other than '/assets' is to be used.

```
const path = require('path');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const asssetsFolder = `${path.dirname(require.resolve('@patientus/video-lib-wc/package.json'))}/assets`;

module.exports = (env) => {
  'use strict';
    return {
        plugins: [
            new CopyWebpackPlugin([
                {
                    from: asssetsFolder,
                    to: 'assets'
                }
            ])
        ]
    }
}
```

Similarly, you could use 'vite-plugin-static-copy' in case you have Vite as bundler

```
import { defineConfig } from 'vite'
import { viteStaticCopy } from 'vite-plugin-static-copy'
import path from 'path'
const asssetsFolder = path.resolve(__dirname, 'node_modules/@patientus/video-lib-wc/assets')

export default defineConfig({
  plugins: [
    viteStaticCopy({
      targets: [
        {
          src: asssetsFolder,
          dest: 'assets'
        },
      ],
    }),
  ]
})
```

Importing the required Roboto font family

```
<style>
  @import url('https://fonts.googleapis.com/css2?
family=Roboto:ital,wght@0,400;0,700;1,400;1,700&display=swap');
</style>
```

Note: You could skip importing Roboto fonts in case your project already includes them.

Importing global styles

```
<style scoped>
    @import '@patientus/video-lib-wc/styles.css';
</style>
```

Importing JS bundle file

```
<script>
    import '@patientus/video-lib-wc/patientus-video-wc'
</script>
```

Including patientus-video component in your template

```
<template>
    <div v-if="config">
        <patientus-video :config="config"></patientus-video>
    </div>
</template>
```

Note-1: Custom elements could receives props only as string, therefore make sure config object is stringified before it's passed.
Note-2: Changes to config are ignored after they are initially set. This is because the config is passed into the constructor, so they can't be changed in any case. Therefore, make sure the config object exists before the component is created. You'll want to hide the DOM element with v-if until you can create the config object.

Types

Type definitions could be imported from `@patientus/video-lib-wc`.

Examples

Below you can see an example to start a video call with the specified configuration including appointment object

```
<script>
import '@patientus/video-lib-wc/patientus-video-wc'

const VIDEO_ELEMENT = 'patientus-video'
const ASSETS_PATH = '/assets'

const APPOINTMENT = {
    topic: 'appointment-topic',
    startDateTime: '2022-10-15 17:00:00',
    endDateTime: '2022-10-15 17:30:00',
    participant: {
        id: 'doctor-id',
        title: 'Dr.',
        firstName: 'Max',
        lastName: 'Mustermann',
        gender: 'm',
        email: 'max.mustermann@dummy.de',
        type: 'doctor',
        imageUrl: 'path-to/doctor-avatar-image-url.jpg',
        specialities: [
            {
                id: 123,
                itemName: 'Allergologie'
            }
        ]
    },
    otherParticipants: [
        {
            id: 'patient-id',
            firstName: 'Jane',
            lastName: 'Doe',
            gender: 'f',
            email: 'jane.doe@dummy.de',
            type: 'patient',
            imageUrl: 'path-to/patient-avatar-image-url.jpg',
            inviteUrl: 'path-to/invite-url'
        }
    ]
}

const TURN_CONFIG = [
    { urls: 'stun:turn.example.de:443' },
    { urls: 'turns:turn.example.de:443', credential: 'example', username:
```

```
'example' }
]

const FEATURE = {
    dataPrivacy: true,
}

const EXTERNAL_URL = {
    dataProcessTerms: '/path-to/data-process-url',
    terms: '/path-to/terms-url',
    dataSecurity: '/path-to/dataSecurity-url',
}

const CONFIG = {
    sessionApiEndpoint: '/example-session-api-endpoint',
    jwt: 'JWT_TOKEN',
    appointment: APPOINTMENT,
    ipProxy: 'example-ip-proxy',
    turnConfig: TURN_CONFIG,
    logoUrl: '/path-to-your-app-assets/your-logo.png',
    feature: FEATURE,
    externalUrl: EXTERNAL_URL
}

export default {
    data: () => ({
        config: null,
        videoElement: null
    })

    async mounted() {
        await this.$nextTick()
        this.videoElement = document.querySelector(VIDEO_ELEMENT)
        this.videoElement?.addEventListener('disconnected', this.onCallEnd)
        this.videoElement?.addEventListener('canceled', this.onCancel)
        this.config = JSON.stringify(CONFIG)
    },

    destroyed() {
        this.videoElement?.removeEventListener('disconnected',
this.onCallEnd)
        this.videoElement?.removeEventListener('canceled', this.onCancel)
    },

    methods: {
        onCallEnd() {
            console.log('user ended the call, do something');
        },

        onCancel() {
            console.log('user exits, do something');
        }
    }
}
```

```
</script>

<template>
    <div v-if="config">
        <patientus-video :config="config"></patientus-video>
    </div>
</template>

<style scoped>
  @import '@patientus/video-lib-wc/styles.css';
</style>
```

API

PatientusVideoComponent (selector: patientus-video)

@Input()	type	description
config	PatientusVideoConfig	configuration object for the video call component

@Output()	value	description
disconnected	undefined	Emits when user ends the video call
canceled	undefined	Emits when user clicks cancel in pre-join step

PatientusVideoConfig

property	type	required/optional	description
sessionApiEndpoint	string	required	api endpoint to fetch the session credentials
jwt	string	optional	added to the auth header of the sessionApi request to fetch protected resource
turnConfig	TurnConfig | CustomServers[]	optional	all media stream communication would be relayed via TURN servers
ipProxy	string	optional	Vonage requests would be forwarded through the IP proxy
appointment	Appointment	required	appointment information
assetsPath	string	optional	path to serve library assets **(default: '/assets')**
logoUrl	string	optional	if specified, it replaces patitentus logo
feature	Feature	optional	toggles optional features

property	type	required/optional	description
externalUrl	ExternalUrl	optional	external urls

CustomServers

property	type	required/optional	description
urls	string | string[]	required	server url(s)
username	string	optional	needed if server requires authentication
credential	string	optional	needed if server requires authentication

Appointment

property	type	required/optional	description
topic	string	optional	appointment topic
startDateTime	string	required	appointment start time (YYYY-MM-DD hh:mm:ss)
endDateTime	string	required	appointment end time (YYYY-MM-DD hh:mm:ss)
isEbm	booelan	optional	flag for ebm appointments
participant	Participant	required	participant joining the video call
otherParticipants	Participant[]	required	other participant(s) joining the video call

Participant

property	type	required/optional	description
id	string | number	optional	participant id (required in multi-party calls)
type	'doctor' | 'patient'	required	the participant could be a doctor or patient
title	string	optional	salutation of the participant
firstName	string	required	first name of the participant
lastName	string	required	last name of the participant
gender	'f' | 'm' | 'd'	required	f: female, m: male: d: diverse
email	string	optional	email address of the participant
imageUrl	string	optional	profile image url of the participant
specialities	DoctorSpeciality[]	optional	specialities for doctors
inviteUrl	string	optional	invite url of the patient app for doctors

DoctorSpeciality

property	type	required/optional	description

property	type	required/optional	description
id	number	required	id of the speciality
itemName	string	required	name of the speciality

Feature

property	type	required/optional	description
dataPrivacy	boolean	optional	toggles data privacy view

ExternalUrl

property	type	required/optional	description
dataProcessTerms	string	optional	data process terms url in dataPrivacy view
terms	string	optional	terms(AGB) url in dataPrivacy view
dataSecurity	string	optional	dataSecurity(Datenschutz) url in dataPrivacy view

Browser Support

- Google Chrome (latest release version)
- Google Chrome for Android (latest release version)
- Beta support for Google Chrome for iOS (latest release version)
- Firefox (latest release version)
- Firefox for Android (latest release version)
- Beta support for Firefox for iOS (latest release version)
- Microsoft Edge versions 79+ for Windows and macOS (Chromium-based versions of Edge)
- Safari 11+ on macOS and iOS. For information on video interoperability and other issues, see the Safari browser support page.
- Opera (latest release of desktop version only)
- Electron (latest release version)

Known Issues

Please see known issues of Vonage Video API

Release Notes

See the changelog for recent changes.

Further Help

To get more help and for any kind of information please send an email to ovs-sdk-support@docplanner.com

Versioning

This library follows semantic versioning convention for commit message guidelines.
https://www.conventionalcommits.org/en/v1.0.0/
https://github.com/angular/angular/blob/22b96b9/CONTRIBUTING.md#-commit-message-guidelines