**Calling Web Api:**

Calling Web Api is a web-based solution that allows its customers to communicate and collaborate through various secure communication channels including voice, video, chat, notes, file share, screen sharing and whiteboard.

Calling Web Api is the first and only collaboration service hosted in Pakistan and offers a very cost effective subscription-based model to its customers. It requires no additional investment on communication infrastructure, hardware or software.

Calling Web Api has been designed with scalable load balance infrastructure for high availability and reduced Network Latency. A private virtual space is created when you join Calling Web Api and it is accessible to authorized members only.

Calling Web Api is highly customizable product.

* Software as a Service.
* Customize your portal, as per your needs.
* Easily Integration with your existing business systems.
* Secure and Reliable Performance.
* Device and Operating System Independent.
* One tool for all your communication needs.
* Install it on-premises, cloud or hybrid.
* Use business branding for your solution.
* Use data network or Intranet, without any dependency on internet .
* Its device independent, use it in a conference room, laptop, desktop, mobile or tablet.

Calling Web Api using WebRTC Technology.

**WEBRTC**

WebRTC is a new and innovative real-time communication option that is both easy and cheap to implement. With up to 93% of communication being non-verbal, it’s essential that you have a video solution which enables employees and customers to interact in a visual way and meet face to face. For remote teams, this helps to connect offices and foster cross-office employee integration. For customers it helps to put a face to your business and it improves both the sales and support process immensely.

* Easy integration and deployment across a variety of platforms.
* Helps you improve the customer experience, thereby improving revenues.
* Enables real-time collaboration through video.
* Secure method of communication.
* Extremely cost competitive with other, similar options.
* Brings best of both worlds (hardware and software based communication tools).

**Calling Web Api Feature:**

* Video Call
* Audio Call

Calling Web Api Some Key Features.

**Client Side (Web):**

Web App

BE API

Client JS API

W3C’s webRTC JS API

WebRTC Shim

WebRTC Plugin

WebRTC Browser

Other Browser

ORTC Browser

IETF’s rtcweb

Internet

Back – End Signaling

(Back – End Media or P2P) && (Back – End ICE or Nothing)

We’ve got both client and server entities with WebRTC, and we think the above depicts the main ones. There are more as your service gets more complicated, but that’s all an issue of scaling and pure development not directly related to WebRTC.

**Web app:**

The web app is what most people think about when they think WebRTC. This is what ends up running in the browser, loaded from an HTML and its derivatives. What this means is that the language you end up with is Java Script, Node.js, PHP and any other language. We are using the combination of JavaScript, Node.js and PHP to create user friendly call API.

**TURN server**

STUN and TURN servers are also necessary. Most likely – you won’t need to do a thing about them besides compiling, configuring and running them. We are using Turn to cater calls if router is working behind firewall. And STUN caters with Synchronous data it also caters calls if router is working behind firewall.

**Media server**

Media servers come in different shapes and sizes. Kurento team update kurento every year and release new and latest versions. Current version of Kurento is 6.9 and we used 6.6 in our application.

The programming languages here depend on the media server itself. Kurento is written in C/C++ with front ends written in Java or Node.js.

Media servers are usually entities that you communicate with via REST or Websocket, so you can just use whatever language you like on the controlling side. It is a very popular choice to use Node.js (Java Script) in front of a Kurento server for example. It also brings us to the last entity. And we are using exactly same Node.js and javascript.



